Computational Construction Grammar based on Formal Ontologies

von Vanessa Micelli-Schmidt

zur Erlangung des Grades
Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

Vorgelegt im Fachbereich 3 (Mathematik & Informatik)
der Universität Bremen
im Dezember 2023
Kolloquium: 8. Dezember 2023
1. Gutachter Prof. Dr. Rainer Malaka
2. Gutachter Prof. Dr. John A. Bateman

Danksagung

Um diese Arbeit fertigzustellen, habe ich von vielen Seiten Unterstützung erfahren, wofür ich mich an dieser Stelle bei allen beteiligten Personen herzlich bedanken möchte. Insbesondere danke ich Herrn Prof. Dr. Rainer Malaka für die Erstellung des Erstgutachtens und Herrn Prof. Dr. John Bateman für die Übernahme des Zweitgutachtens. Ohne Ihrer beider nahezu endlosen Geduld und kompetenten Unterstützung wäre diese Arbeit nicht realisiert worden. Ich danke Dr. Robert Porzel, der mich zum Promovieren anstiftete, von dem ich in frühen Jahren viel lernen durfte und dem damaligen Team an der European Media Laboratory GmbH, darunter besonders Dr. Berenike Litz, für die Motivation und langjährige Freundschaft. Ich danke Prof. Dr. Luc Steels, der mich in den Jahren, in denen ich Forschung betrieben habe, inspiriert hat und meinem Language Team an den SONY Computer Science Laboratories in Paris. Ich danke meiner Freundin Dr. Nancy Chang für die vielen inspirierenden, motivierenden und lustigen Momente - in der Arbeit oder unserem gemeinsamen Zuhause. Ich danke der Klaus Tschira Foundation gGmbH, der European Media Laboratory GmbH und dem Bundesministerium für Bildung und Forschung für die Finanzierung meiner Forschungsarbeit.

Besonders möchte ich mich bei denen bedanken, die immer ermutigend und unterstützend an meiner Seite sind – meiner Familie: Danke, Cathrin Geiß, meine langjährige und beste Freundin, die zur Familie gehört. Danke, Petra und Severino Micelli – ohne Euch hätte nichts in meinem Leben so funktioniert, wie es funktioniert hat. Danke, Jana Micelli-Loesch – mein Vorbild in so vielen Belangen, die weltbeste Schwester.

Danke, Dirk Schmidt, unglaublich, dass Du immer an meiner Seite bist und nahezu unfassbar, was wir gemeinsam schaffen konnten (und können!) und Danke, Valentin Schmidt, dafür, dass Du bist wie Du bist; Euch beiden widme ich diese Arbeit. Danke, dass Ihr immer für mich da seid.

[Grammar is...]
[14] A gateway to liberation, a cure to the blemishes of speech, purifier of all (other) disciplines, it shines as being applied to them.
[15] Just as all thing-classes depend upon word-classes similarly, in this world, this (grammar) is the basis of all disciplines.
[16] It is the first rung on the ladder towards liberation, it is the straight Royal Road for those desirous of (reaching) that goal.
(Citation from [Pilla, 1971], the English translation of Bhartrhari's Vākyapadīya, Indian grammarian and language philosopher of the 7th century.)

Contents

Chapter 1

Introduction

Discussions about what constitutes natural languages, how they are learnt, or how they change over time have always been an omnipresent topic in linguistics and can be traced back many decades in history. At all times, the study of grammar – the art of letters – has been a major issue in language research. Antoine Furetière, for instance, member of the *Académie Française*, depicted grammar's importance and presence in the 17th century "...au royaume d'eloquence" [Furetière, 1658] – in eloquence's kingdom – where the various disciplines of a language's grammar and style are displayed as troupes that are fighting each other (see Figure 1.1).
In modern linguistics, similar discussions are still ongoing, adding not only an additional computational perspective on languages (I will talk about that later in this section) but also having grammar's relevance for a language's meaning as a topic. This means that there are grammar theories, that consider the linguistic discipline of semantics not as a separate part of a language's grammar but as a major part of it, equally important as other linguistic disciplines as for instance morphology or tense. This work focuses on such a grammar theory.

Today, there is a strong, additional computational perspective on languages. Formal grammars are needed to be machine-processable – both to understand language but also to generate it.

Formal models of different kinds of language phenomena are created (lin-

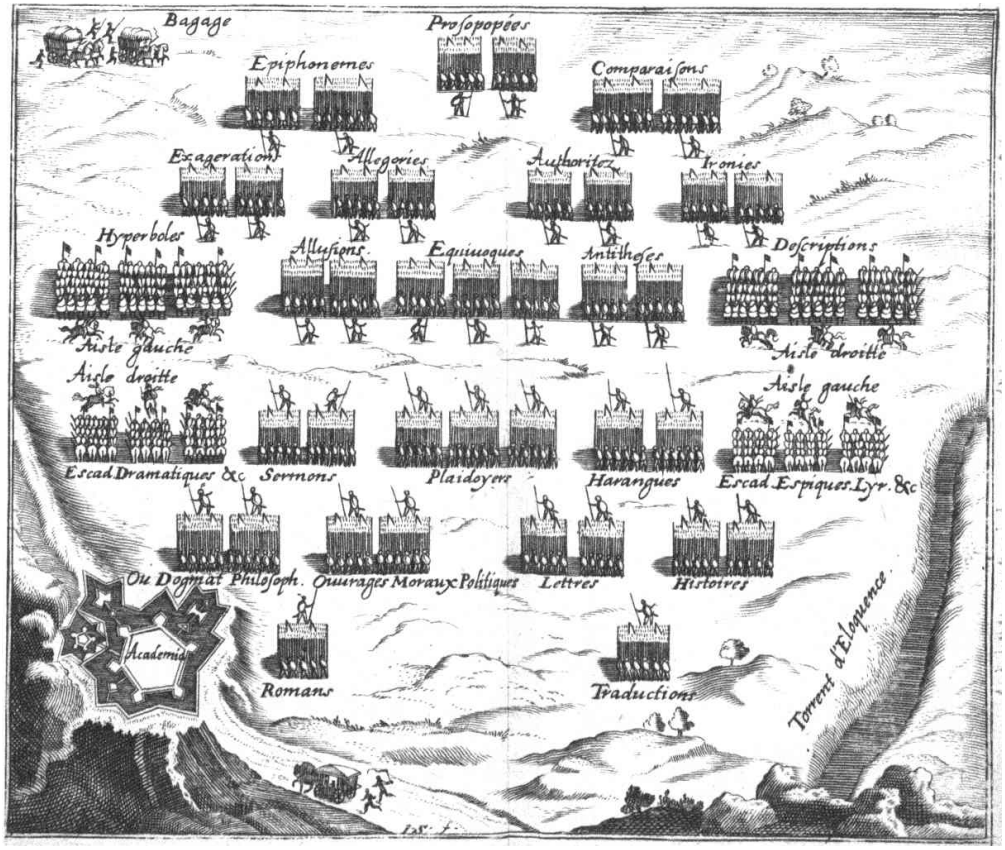


Figure 1.1: The allegory of grammar and style [Furetière, 1658].

guistic engineering) and used in areas which require natural language processing (NLP). Linguistic engineering and particularly grammar engineering, one of its sub-disciplines focusing on the development of formal, operational i.e. machine-processable grammars, still constitutes a serious bottleneck in the development of applicable natural language processing systems as for example spoken dialogue systems, automatic translation systems, chatbots, or question answering systems. And these systems are suddenly everywhere! We have advanced from rule-based chatbots like ELIZA [Weizenbaum, 1966] that started being explored in the 60ies already to corpus-based chatbots [Serban et al., 2018] or a hybrid architecture of such [Paranjape et al., 2020]. Technology is finally that advanced that we can access data anywhere at anytime. That we can process it

on devices like our phones. These advancements paved the way for us getting used to using speech assistants, for instance, in our cars, being on walks, and even in our own homes – to support us with simple tasks like navigation, calling or sending text messages, or even turning the lights on or off without having to get up from the chair. Lately, we've also started using them in education [Bahja et al., 2020] or to support us in our daily life at work with various predefined scenarios (see e.g. [Rizk, 2020]). Our children grow up with user interfaces that allow spoken language input and will be used to this convenience much more than we can be. And why should they eventually abstain from this convenience in their work lives? Gartner, a leading technological research and consulting firm, predicts that until 2025 50% of all employees will be using a digital assistant of any kind daily. We can see the pace of this trending topic comparing it to 2019, where it has still been only approximately 2% [Bradley, 2020]. Amy Webb, well-known futurist and CEO of the Future Today Institute and professor at the New York University, recently stated that a major (exciting!) advancement with respect to speech assistants is that they do not only understand what we say but also what we mean by it. But is that really true? How satisfied are we when having used a chatbot to upgrade our recent flight? How happy are we when we try to book vacation with the HR chat bot? What has happened when we start being frustrated with the system and turn to other mechanisms of resolving our requests and tasks? Speech systems, no matter how advanced, will always lag behind when it comes to understanding real spoken language. And we believe that there will always be manual, human input needed in order to cover language phenomena occurring in natural language interaction with a system.

Over the years, linguists proposed various grammar theories or linguistic formalisms some of which have been modelled from a computational perspective to be usable in NLP systems of various kinds. The grammar theory which constitutes this work's focus is part of the Construction Grammar framework. The reasons for this choice will be made clear within

the course of this work.[1]

Engineering a machine-processable Construction Grammar for use in NLP systems is still being researched and improved. At this stage of research in computational construction grammar, there are numerous open issues that leave enough room for trying out new ways of representing constructivist ideas.

There is promising progress – especially in the development of Fluid Construction Grammar exciting results have been yielded – and also a growing interest in machine-readable Construction Grammar formalisms as can for instance be tracked by the numerous publications and talks at linguistic conferences as for instance the yearly ICCG.[2] More concretely, recent advances in Fluid Construction Grammar have, for instance, demonstrated progress in solving two visual dialogue tasks [Verheyen et al., 2023], a visual question answering benchmark [Nevens et al., 2019], or mining opinions on the web [Willaert et al., 2022]. Additionally, there is a lively exchange with researchers stemming from other grammar formalisms, as e.g. Head-driven Phrase Structure Grammar (HPSG) [Pollard and Sag, 1994], Sign-based Construction Grammar (SBCG) [Sag, 2012] or cognitive linguists in general (see e.g. publications from the yearly International Cognitive Linguistics Conference (ICLC)[3] or AFLICO conferences[4] or two earlier publications [Micelli, 2009] or [van Trijp, 2009]).

Concluding the above, the main motivation for developing this work are the challenges grammar engineering in general and Construction Grammar engineering in particular (see also [van Trijp et al., 2022]) is still facing nowadays, and the fact that current systems have still not reached a

[1] The two existing computational formalisms that base on construction grammar are *Fluid Construction Grammar* and *Embodied Construction Grammar*. They both were developed for very special purposes and will be discussed in more detail in Section 2, dealing with this work's theoretical background.

[2] The International Conference on Construction Grammar (ICCG) (see e.g. *uajd.ff.cuni.cz/en/node/546*)

[3] See *https://www.cognitivelinguistics.org/en/event/detail/international-cognitive-linguistics-conferences-iclcs*.

[4] See *http://www.aflico.fr*

satisfying maturity when it comes to natural language interaction, even after these major advances in technology and seemingly endless amounts of data and computing power, grammatical formalisms like the ones discussed in this work, might help in getting a bit closer to real language interaction filling in a few of the gaps detailed further below.[5] Figure 1.2 represents the main challenges of grammar engineering alongside with its major themes. All of them will be further elaborated on in the following section.

Figure 1.2: Major challenges in grammar engineering and especially in engineering constructing grammars.

1.1 Main Motivation

Formal grammars resulting from traditional generative grammar formalization approaches usually consist of a set of terminal symbols, i.e. a lexicon, and of various rules.

When dealing with natural language, it was soon realized, that those traditional grammar accounts were able to handle a lot of natural language phenomena, but as soon as it came to issues concerning for instance agreement between the subject and the predicate of the sentence, they failed, which means in this case that they tended to over-generate. Therefore, so called *unification grammars* were developed, adding constraints to the rules of context-free grammars and feature structures to their instances. Thus, the mentioned shortcomings were eliminated as for instance over-generation could be prevented and agreement could be checked. Those

[5] See also Weissweiler et al., 2022 on investigations on constructional knowledge of large-language models and their limits on capturing constructional meaning.

grammar formalisms are extremely powerful, but still one issue of major importance is completely neglected within those frameworks which is the significance of grammar's contribution to *semantics*. This work will put semantics and a holistic grammar formalism in its center.

Amongst other things, the above mentioned deficiencies lead to the development of a formal approach of grammar called Construction Grammar [Lakoff, 1987, Fillmore and Kay, 1987, Talmy, 1988, Goldberg, 1995, Kay, 2002], to cite just a few publications on that topic. Construction Grammar presents a particular cognitively motivated grammar framework which allows for exactly one data type: a so-called *construction*. Constructions exist on every level of language and are pairings of form and function, which means that even syntactic structures – constituting the form side of a construction – can possibly contribute meaning to utterances. While the category of linguistic form of constructions covers any variety of formal grammatical components as for instance components concerning morphology, the lexicon, or syntax of a language, the category of linguistic function includes information which is in traditional linguistics generally subsumed under the terms *semantics* or *pragmatics*.

Construction Grammar was mainly developed to handle phenomena occurring in natural language which were problematic to be dealt with in other theories beforehand as, for instance, partial utterances, ellipses, coercion, i.e. the usage of verbs as being ditransitive although their conventional use is an (in-)transitive one or as well as conversational implicatures. Furthermore, it presents a theory that offers a plausible explanation of how language can possibly be acquired.[6] In addition to that, language research through centuries has proven that language is a complex adaptive system [Steels, 2000, Beckner et al., 2009] and that a constructional approach to language analysis offers the most promising linguistic framework to explore and potentially represent this flexibly.

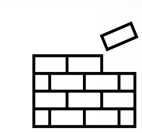
[6] See for example [Tomasello, 2003] or [Lieven and Tomasello, 2008] for a usage-based approach on language acquisition.

For exactly this reason, amongst others to be elaborated further, we adopt a constructivist position of grammar representation in this work, meaning that a grammar should ideally include every layer of language, i.e. form and meaning or function as suggested in construction grammar theory. We are aware of the fact that statistical systems are successfully used in areas that apply NLP as for instance in machine translation, human-machine interaction, or machine learning for language understanding. The most successful systems nowadays are, however, hybrid systems which combine both statistical and symbolic natural language processing and there is a growing consensus in the field that hybrids will continue to achieve the best results in, for instance, the area of machine translation [Wilks, 2009a]. Purely statistical applications have by now reached a point where only small further improvements can be expected. This fact motivates the development of symbolic grammar formalisms for natural languages to provide a base for deep natural language processing.

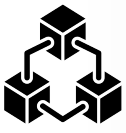
The progress towards formalizing constructions and towards a machine-processable construction grammar architecture raises a number of promising issues and challenging questions (see, for example, publications by [Kay, 2002, Bergen and Chang, 2005, Steels, 2011, Steels, 2017]).

And at this stage of research both in the field of theoretical and computational construction grammar, open issues leave lots of room for experimenting with or studying various ways of representing constructivists' ideas. This work attempts to fill in some of the open issues. It will deliver a formalization of construction grammar using the state-of-the art in knowledge representation and exemplifying it with an example sentence.

As graphically represented in Figure 1.2, various points are important in the development of computational implementations of grammars in general which will be discussed and included in the herein proposed framework. With the main focus on the implementation of a construction grammar based on ontologies, we consider the following points as being essential in its design and development and raise the following questions:



How 'easily-operable' is the grammar? Are there usable editors which can graphically display the structures? With which domain of application can it deal?



Which already existing modules can be reused in the grammar? For example, the integration of existing (linguistic) knowledge bases already covering specific linguistic knowledge as for instance Frames as represented in FrameNet into other grammars, prove to be extremely complicated or even impossible without making major changes to the grammar's format. Also, integrating different domain knowledge or already existing partial grammars constitutes a challenge.

Techniques of development should be documented for improving long-term and multi-developer maintainability of the grammar. The grammar should be easily extendable (also by non-experts), especially to different domains as we can never foresee the exact needs of any end user. Extensions, and especially large-scale extensions, of grammars mostly turn out to be problematic. Often expert knowledge is required to build further grammatical components, the model does not scale and its performance decreases the bigger the grammar gets.

The grammar should be relevant in theoretical and computational linguistics: The development of the grammar should be understandable for other research communities. It should be applicable in the field of natural language processing. In general, formalisms are necessary and useful in testing the linguistic accuracy of hypotheses to find out, for example, if they are contradicting or exact enough to be operational.

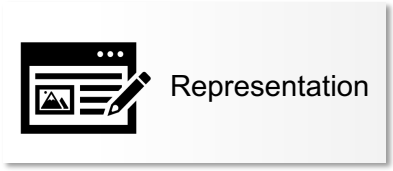
To build grammars that can be distinguished from so-called "toy grammars" which can merely deal with a handful of linguistic structures, the coverage of a grammar is another main issue that has to be tackled in the discipline of grammar engineering. So far, there do not exist many deployable, broad coverage grammars and none does based on the constructivist framework.

The main questions raised within this context are:

1. How big is the grammar, i.e. its lexicon and more complex grammatical constructions, and can it be extended (including an extension of the domain)?

2. How big is the effort to extend the grammar?

3. How powerful is the grammar (e.g. measured by the amount of grammatical phenomena that it captures)?



What is the representational format of the grammar? The grammar format should be in line with the state-of-the art in knowledge representation. This way, compatibility issues can be avoided as standard tools for representation can be used. In addition, extensibility and editability are thereby made easier and it can be ensured that the latest technology is applied.

How time-consuming, i.e. complex is it to edit the grammar? The grammar should be editable within a reasonable amount of expertise. The availability of editors for non-expert and non-technical users are of utmost importance to help in grammar engineering, extending, and editing.

It would be helpful to have a common ground to compare grammars. There should be proposals concerning evaluation methodologies and metrics which can capture the added benefits of deep linguistic analysis as well as evaluation techniques which can compare grammars across languages and linguistic theories (cf. [Marques and Beuls, 2016]). The questions raised in each of this lists' bullet points should be answered within an evaluation.

To summarize, many problems and open issues in the field of grammar engineering base on the absence of a common or shared vocabulary or agreed standard format of how linguistic knowledge of any kind should ideally be represented. With such a format for instance reusability and accessibility of the linguistic knowledge base would be increased.

Another major deficiency is the absence of intuitively-to-use editors, tools and conventionalized engineering methods defining precisely how to ideally build a reusable, accessible and scalable grammar model including large-scale coverage. All of the mentioned deficiencies additionally result in the fact that there is no common ground on which different grammars' quality, efficiency, performance, and coverage could be compared. This work aims at providing such a common ground, exemplifying the framework by a pre-population of instances and integration of various modules according to the state of the art in knowledge representation. Let's have a closer look into what exactly this work comprises.

1.2 Thesis Aim and Contribution

Two decades ago, within the Semantic Web effort,[7] machine-readable semantics has been added to content retrieved on the World Wide Web so that it could be processed by machines [Berners-Lee et al., 2001]. This content did not only consist of data encoded in html-tables but also of running, natural language texts found on websites – even back then it had been a mixture of structured and unstructured data. The challenge still remains: How to make texts machine-understandable? Natural language texts are complex, ever changing systems. This work tries to add to solving this challenge with the help of ontologies including lexical grammatical constructions that equally focus on form and meaning of each of their constituents.

[7] http://www.w3.org/standards/semanticweb/

Ontologies have been the Semantic Web's basic building blocks and have been continuously used as knowledge representation for natural language processing applications.[8] They developed into a state of the art representational method when it comes to formally describing a set of concepts and the relationships that hold between those concepts.

What we suggest in this work is a formalization of construction grammar by means of formal ontologies. The result of this undertaking is a powerful ontological model which is enriched with a cognitively motivated grammar layer to be used in natural language applications.

To account for extensibility, it will additionally be elaborated in detail how further knowledge sources can be integrated into the model since its format is conform to the state of the art in knowledge representation. Basic linguistic information for instance such as parts-of-speech, case, number and grammatical gender that are parts of basic constructions can be integrated by adding an ontological plugin which we developed and which is called LingInfo. Furthermore, domain-specific information has been integrated to our grammar framework stemming from a knowledge source being organized consistent with FrameNet's structure [Baker et al., 1998], a knowledge base including a huge dataset of semantic frames consistent with Frame Semantic theory. Construction grammar and Frame Semantics are sister theories in cognitive linguistics and many constructivist approaches to grammar claim that the constructions' semantics is represented by frames. However, how exactly this relation is made explicit is left underspecified and most analyses only focus on the skeletal meanings [Goldberg, 1995, p. 28] that underlie grammatical constructions. This gap may cause inconsistencies in the development of both theories and

[8] The successful and continuous use of ontologies for Semantic Web or natural language processing applications can for instance be traced at the yearly held *International Semantic Web Conference* http://swsa.semanticweb.org/content/international-semantic-web-conference-iswc or in numerous publications as for instance [Hitzler et al., 2009], [Davies et al., 2006], [Bateman, 2010], [Hitzler and Shimizu, 2018] or [Sharma et al., 2019].

leaves a lot of crucial issues unaddressed within cognitive linguistics.

To summarize, the herein presented effort enables easy access to or even integration of various already existing (linguistic) knowledge sources coupled with constructions. Thereby, the coverage of the grammar model can be increased incrementally and even provides interfaces to further couple it with additional knowledge sources. Each step is well documented by discussing both the ideas and the principles behind the design of the model and its implementation and engineering issues.
The present work tries to provide some missing jigsaw pieces in building a grammar model fitting into the construction grammar framework which can be deployed in natural language processing systems including the following objectives:

- It provides a concrete method of implementing a formalization of construction grammar based on ontologies. This method is well documented in order to make the ideas and analyses reusable for instance for extending the framework with additional (domain) knowledge or even for writing new grammars.

- By providing a standardized grammar format that NLP tools using ontologies can use (see e.g. [Mahesh and Nirenburg, 1995]), reusability for various NLP applications is guaranteed.

  Concrete advantages coming with the ontological format and the herein used engineering environment are listed in the next section.

- We integrate various external knowledge bases and describe the interface where this knowledge can be tied to the grammar model so that it can possibly be further extended similarly.

Finally, the model represents a rich knowledge base for different kinds of linguistic information which can benefit from the advantages that come with ontologies, especially with the use of a foundational ontology, which include the following:

- It is compatible with other ontologies based on the same foundational ontological model.
- The foundational ontology provides a setting including deep and well-defined semantics.
- It provides a reference point that enables a comparison to other ontologies.
- It enables merging with other ontologies.
- It enables the reuse of a predefined set of entities.
- Its consistency can be checked automatically by using existing tools (see for instance [Baclawski et al., 2002]).
- Editors providing visualizing tools or search functions (among other functions) are already available.
- Applying standard ontology learning mechanisms to extend the grammar including its lexicon is possible.
- In case the ontology gets too big concerning its instances, they can be externally stored for instance in *Jena*[9], providing storage of data in ontological format in a relational database.

It is not in the scope of this work to provide a solution to all currently existing grammar engineering problems. Also, we do not claim to provide a grammar without weaknesses which can be compared to those powerful formalisms developed within decades by a group of scientists as this work is not part of a broad grammar engineering project. However, the contribution of this thesis aims at contributing to formal construction grammar development as this promising grammar theory is still searching for a robust, usable, flexible and extendable computational grammar formalism to be applied in natural language processing systems. Therefore, we deep dived into existing construction grammar formalisms, identified

[9] See *http://jena.apache.org/*.

some weaknesses, picked out crucial constructivist ideas and elementary parts of the grammar engineering process and describe their modelling in detail with the help of one example sentence within a representative and widely-used ontological framework. The sentence will be modelled within that ontological framework that is intended to be reused and extended in future increasing construction grammar's visibility and attempting to iron some weaknesses out.

1.3 Organization of the Thesis

This thesis is structured as follows:

Following the general introduction given in this Chapter, Chapter 2 presents the theoretical background relevant to this work in four sections:
It starts with presenting an overview of relevant grammar theories in linguistics arguing why linguistic formalisms are needed at all and gives a short historic overview on grammar debates. It then introduces generative grammars and typed feature structure grammars. The chapter continues with discussing why computational implementations of linguistic formalisms are fundamentally important before presenting main issues in grammar engineering in computational linguistics in general and briefly presenting those grammar implementations relevant for this work in particular. The next section introduces formal ontologies, the definition, formats and engineering in general. It introduces foundational ontologies and describes how linguistic knowledge is represented in ontologies. Finally, schemas and frames used for semantic representation are introduced.

The subsequent Chapter 3 presents a detailed description of all components being important for the herein produced work. It starts with an introduction to construction grammar in general and continues with a comparison on the two existing computational construction grammars, i.e. Embodied Construction Grammar and Fluid Construction Grammar. It then discusses the motivation and merits of a further computational for-

malization of construction grammar. Additionally, the employed foundational ontology is presented and the motivation for this choice elaborated on. The chapter continues with an overview of the LingInfo model that enables the integration of basic linguistic information into an ontology and closes with a few remarks on constructional meaning.

Chapter 4 represents the heart of this work. It starts with an informal constructional analysis of the selected example sentence that is used throughout this work and specifies the implementation details of the ontological grammar model that is created (called `ECtoloG`). It describes its various components, i.e. lexical, compositional and other constructions, in great detail. The following section focuses on how linguistic information is represented in constructions and modeled in the `ECtoloG`. The upcoming part of the chapter focuses on the meaning representation of constructions, i.e. on schemas and frames. A case study is presented that deals with the integration of FrameNet frames into construction grammars. The chapter ends with a summary of the so far obtained merits.

In the subsequent Chapter 5, a proposed example language processing flow is presented where the ECtoloG is used. The application flow is first described and then stepwise processed. It starts with an automatic corpus creation, a conversion into parser-readable format, the constructional analysis of the sentence, the conversion of the output into further processable information. The chapter then describes the automatic population of the ECtoloG and the LingInfo model with all terms and the appropriate linguistic information of the created natural language corpus. The tools and scripts that are used are described.

Chapter 6 ends with a summary of the ECtoloG's contribution to the construction grammar and ontology communities and suggests various ideas on how to continue the reported efforts and explorations.

Chapter 2

Theoretical Foundations

The following chapter deals with the theoretical background of this work describing the state of the art in those fields we consider being important for the creation of our grammar model as described in further detail in Chapter 4. Additionally, it already introduces the main differences of existing grammar implementations and issues that are considered important in this work.

The first section includes basic information on grammars in theoretical linguistics. It also includes a discussion why we believe formalisms are necessary in linguistics. The subsequent section deals with those grammars applied in the field of computational linguistics, i.e. with concrete computationally applicable implementations of grammars, already introducing implementations based on construction grammars. This section, as well, starts with a short discussion, why we believe that linguistics is in need of computational models of the theories it suggests. Subsequently a section on formal ontologies follows, including their definition, their formats, various engineering approaches, a definition of foundational ontologies and their adequacy and the state of the art in representing linguistic knowledge in ontologies. The chapter concludes with a section on Frame Semantics, regarded as a complementary framework to Construction Grammar. It sketches out frames and other frame-based templates as for instance image or executive schemas. Those templates play a crucial

role in natural language understanding or reasoning.

2.1 Grammars in Linguistics

Why do we need linguistic formalisms at all? This section focuses mainly on exactly that question. In addition, it will be discussed what constitutes a grammar in theoretical linguistics followed by a brief presentation of different kinds of grammars and their major features.

2.1.1 Linguistic Formalisms

The question to be answered in this section is why linguistic formalisms are needed at all and – subsequently – as there already exist dozens of those formalisms – why we present yet another one with again a different flavor. As especially the second question will be answered along the course of this thesis, it will briefly be motivated here.
As previously mentioned, the development of linguistic formalisms started in the late 1950s (see for instance [Garvin, 1954, Tesnière, 1959]). How to model these formalisms as suggested during the past decades and adding up to dozens of ways nowadays has mainly been and still is developed to eventually make linguistics a proper science. Research fields such as mathematics, geography, or physics use formalisms that are shared in the respective communities. The formalisms give guidelines about which names to use, which representations or inference procedures per respective areas. The same need for a common language or a common denominator arises in linguistics. The answer to the maybe obvious question, why there exist so many different grammar theories in linguistics instead of agreeing on a common one, as it is done for instance in algebra, is that it is still a mystery how linguistic structures are grounded in the brain. Scientists still argue about how language is learnt. This uncertainty makes it that difficult to ground linguistic structures in reality.
When studying various linguistic formalisms it becomes clear that they have to adhere to a variety of principles which will be listed below:

- Above all, a linguistic formalism has to be clear and precise. This principle holds for all the following points listed below.

- A linguistic formalism should represent well-defined notations for structures and processes relevant in the language they aim to describe. An example for a linguistic structure is for instance a noun phrase which can possibly be composed of a determiner and a noun. As shown in Figure 2.1, a noun and a determiner are then combined into a larger unit (the noun phrase). Whole sentences could be captured in such tree-like structures, assigning parts of speech to their components (as here *noun* and *determiner*). The model presented in this work will give examples of linguistic structures which can be captured and suggests how additional ones can be added.

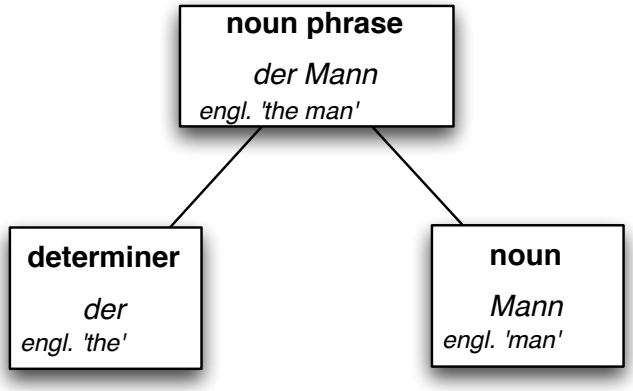


Figure 2.1: The phrase structure of a noun phrase.

In this work, semantic structures that underly complex syntactic structures are regarded as being important in particular. A famous example is for instance a caused-motion construction, in detail described in Goldberg [Goldberg, 1995]. Caused-motion constructions describe that *X causes Y to move Z*. They are composed of a subject, a direct object and an oblique object on the syntactic side. On their semantic sides, those sentence parts play different roles in a sentence. While the subject plays the role of the agent of the

caused-motion event, the direct object plays the role of a mover or a moving thing and the oblique object acts as a goal in a prepositional phrase. Goldberg's example sentence is *She sneezed the napkin off the table*. Where it is the caused motion constructions that contributes to the meanings of the parts of the sentence that it is her (the agent) sneezing that causes *the napkin* (the moving thing) to move off *the table* (the goal of the prepositional phrase).

- As languages show many regularities, as e.g. in German the conjugation of verbs, a linguistic formalism has to capture those. This aspect is also important for learning, as novel verbs might enter the speech community and immediately those rules can be applied to form the different conjugational paradigms. This leads directly to another aspect which has to be captured:

- A linguistic formalism should take into account what is necessary to successfully interpret and produce well-formed utterances. Also it has to be flexible enough to be expandable as new forms or meanings might enter the speech community. Therefore, ways to acquire those and the link between a form and its function in communication have to be defined.

To check whether those requirements are met by a linguistic formalism, implementations of those grammars are necessary. Section 2.2.1 briefly goes into this issue. Before, however, we will give an overview on grammar debates and look a bit closer at a few linguistic formalisms to get an overview of the theoretical linguistic foundations.

2.1.2 Short Historic Overview on Grammar Debates

Already for decades, there have been debates in linguistics about what language is, how it is acquired, and what exactly constitutes a language's grammar. In sciences in general, but especially in the debates on grammars the so-called 'nature-nurture'-debate is omnipresent and deals with

which parts of a language or – to be more specific – of a grammar are already given or innate and which parts have to be learned. While this discussion whether word-meaning mappings were universal or conventional can already be back-dated to the six orthodox Indian schools of thought[1] and discussions within from approximately the 7th century BCE [Raja, 1969, Arapura and Raja, 1990], Chomsky kept this discussion vital in modern linguistics advocating the generative approach to grammar (see for example [Chomsky, 1957, Chomsky, 1965] or more detailed information in Section 2.1.3).

Following those generative approaches to grammar, recently more functional (as e.g. proposed by [Dik, 1978]), usage-based and cognitive approaches to language and its grammar proposed by cognitive linguists in the late 1980s (for instance by [Fillmore and Kay, 1987, Lakoff, 1987] or [Langacker, 1987]) became more and more influential trying to explain language learning based on cognitive mechanisms as, for instance, with the help of frame-based categorization abilities.

To make the various approaches to grammar more powerful, various models of linguistic knowledge were being developed within the research field of theoretical linguistics. Noam Chomsky again was the first person proposing a formalization of generative grammars and – more specifically – proposing a hierarchy which is known today as the *Chomsky Hierarchy* [Chomsky, 1957]. Within the following sections, let's have a look at a short overview of grammars in theoretical linguistics, starting with a short summary of the mentioned *Chomsky Hierarchy* before passing on to typed feature structure grammars.

2.1.3 Generative Grammars in Theoretical Linguistics

Generally speaking, a grammar can be described as being a generative mechanism that allows to generate strings of different kinds of signs of any kind of length, such as for example phonemes, words, morphemes,

[1] For a brief overview, see *https://en.wikipedia.org/wiki/Indian_philosophy*

and so forth.
Within the mentioned *Chomsky-Hierarchy*, Noam Chomsky classified grammars into four different types which will be listed below.[2]

The most general definition of a grammar is the following where $A, B \in \Phi; \omega \in \Sigma^*$ and $\alpha, \beta, \gamma \in \Gamma^* = (\Phi \cup \Sigma)^*$:

A grammar $G = \langle \Phi, \Sigma, R, S \rangle$ is a quadruple consisting of the following four components:

1. A finite set Φ of nonterminal symbols. Typically this set contains syntactic categories as, for example, NPs (nominal phrases), PPs (prepositional phrases) or parts of speech as, for instance, V (verb) or N (noun).

2. A finite set Σ of terminal symbols ($\Phi \cap \Sigma = \oslash$). This set includes all atomic parts which are defined by the grammar whereas atomic means not decomposable.

3. A finite set $R \subseteq \Gamma^* \times \Gamma^*$ of production rules with a left and a right side where terminal or nonterminal symbols can be found

4. A start symbol $S \in \Phi$

The four types of grammars differ from each other with respect to the constraints on the set of rules R.

Type-0-Grammar/Unrestricted Grammar:
A grammar is *unrestricted* in case its rules have the following form:

$\alpha \rightarrow \beta$ where $\alpha \neq \epsilon$ and $\alpha \notin \Sigma^*$

[2]The following definitions follow those by [Klabunde, 1998] who gives a detailed introduction to the subject.

The only constraint on the set of rules R is that at least one nonterminal symbol is substituted by an arbitrary string over Γ. In other words, there has to be at least one nonterminal symbol on the left side of each rule. Besides, on both sides of the rules all elements of the complete alphabet Γ are allowed. This constraint makes languages which are generated with those grammars highly complex, therefore, they are not used in computational applications.

Type-1-Grammar/Context-sensitive Grammar:
A grammar
$G = \langle \Phi, \Sigma, R, S \rangle$ is *context-sensitive* in case all of its rules have the following form:

$\alpha A \gamma \rightarrow \alpha \beta \gamma$ where $\alpha, \beta, \gamma \in \Gamma^*, A \in \Phi, \beta \neq \epsilon$ or $S \rightarrow \epsilon$

That means that a nonterminal symbol A can only be substituted by a word β from the complete alphabet in case it occurs in the context α _ γ. Context-sensitive grammars contain rules that have strings on both sides.

Type-2-Grammar/Context-free Grammar:
A grammar $G = \langle \Phi, \Sigma, R, S \rangle$ is *context-free* in case all of its rules have the following form:

A $\rightarrow \alpha$ where $A \in \Phi$ and $\alpha \in \Gamma^* = (\Phi \cup \Sigma)^*$

Rules of context-free grammars require a single symbol on their left side. A major part of natural languages can be covered with context-free grammars (although Chomsky argued that natural languages are not context-free). However, as for instance shown by Shieber [Shieber, 1985], there are a few structural properties that go beyond their capacities.

Type-3-Grammar/Regular Grammar:
A grammar $G = \langle \Phi, \Sigma, R, S \rangle$ is *right regular* in case its rules have the following form:

$A \to \omega$ and $A \to \omega B$ where $A, B \in \Phi$ and $\omega \in \Sigma^*$

and *left regular* in case its rules have the following form:

$A \to \omega$ and $A \to B\omega$

A formal language is regular in case it can be described with the help of a regular expression.

Context-free grammars are able to capture a lot of natural language phenomena. Phenomena such as the segmentation of complex expressions into parts (being possibly complex, as well) can for example be described. However, there is no way to describe generalizations in a satisfying way with context-free grammars' nonterminal symbols [Kolb, 2004]. For example, a grammar that generates singular and plural noun phrases should use different nonterminal symbols for each phrase as, for example, NP_{sg} or NP_{pl}. From a formal perspective those variables are completely different. Their appearance, however, suggests a certain structural similarity. One possible solution to this problem is to use feature structures to model complex categories. Those structures play a crucial role in so-called *Unification Grammars*, often also called *constraint-based formalisms* or *typed feature structure grammars*. Those grammars and their basic building blocks – the *feature structures* – will be the topic of the following section.

2.1.4 Typed Feature Structure Grammars

Basically, typed feature structure grammars are grammars in which the non-terminal symbols are replaced by feature structures and which use feature structure unification as an operation. This chapter defines the notions of *feature structure* and of *unification*. Amongst others, the logic of typed feature structures is described in [Pereira and Shieber, 1984, Shieber, 1986] and in [Carpenter, 1992] and an overview on Unification Grammars can be found in [Francez and Wintner, 2011].
Feature structures offer a flexible way to represent (linguistic) information. They are mainly used to present complex objects that have different characterizing features or attributes. They basically are lists of features – or of attributes – where each feature can be assigned to a value, being either atomic or as well complex, i.e. another feature structure. Therefore, another frequently used term instead of the term *feature structure* is the term *attribute-value pair*.
Additionally, feature structures constitute the lexicon entries of unification grammars, while context-free grammars list atomic categories. Since the information which is encoded in feature structures is highly rich, the lexicon has to be organized in a clearly structured way. This is normally done via inheritance.[3]
The following formal definitions follow [Carpenter, 1992] and Copestake's shorter versions [Copestake, 2000] who adhered to Carpenter, as well.

Definition Typed Feature Structure Grammar: A typed feature structure grammar $G = \langle T, \subseteq, F, \Theta \rangle$ is a tuple which consists of the following components:

1. $\langle T, \subseteq \rangle$: A type hierarchy which is a finite bounded complete partial order

2. F: A set of feature symbols

[3] There are unification grammars that do not use the traditional inheritance mechanisms as for instance Fluid Construction Grammar (see Section 2.2.3 and for instance [De Beule and Steels, 2005, Steels, 2017]).

3. Θ: A set of typed feature structures

Definition Typed Feature Structure: A typed feature structure $\theta \in \Theta$ over T and F is a rooted, directed, labeled graph. It is a tuple $\langle Q, r, \delta, \theta \rangle$, where

1. Q is a finite set of nodes
2. $r \in Q$ (r is the root node)
3. $\theta : Q \rightarrow T$ is a partial typing function
4. $\delta : Q \times F \rightarrow Q$ is a partial feature value function

In general, feature structures are displayed in a frame-like *attribute-value matrix* (see [Carpenter, 1992]). Figure 2.2 displays a simple attribute-value matrix describing a person. The feature structure is of type `person` and lists the features `name`, `surname`, `birthday`, `father` and `mother`. The first mentioned features are assigned to atomic values. The feature `father` is assigned to another feature structure again of type `person`. The same is the case for the feature `mother`. This displays the way of how recursion can be modeled with the help of feature structures.

person		
NAME	*Mary*	
SURNAME	*Doe*	
BIRTHDAY	*1.1.1980*	
FATHER	*person*	
	NAME	*John*
	SURNAME	*Doe*
	BIRTHDAY	*2.2.1950*
	FATHER	...
MOTHER	...	

Figure 2.2: An attribute-value matrix describing a person.

Unification is a kind of pattern matcher. It enables merging information content of two feature structures and also, in case their content is incompatible, it rejects the merger. The following presents the formal definition.

Definition Unification: The unification $F \bigcap F'$ of two feature structures F and F' is the greatest lower bound of F and F' in the collection of feature structures ordered by subsumption [Copestake, 2000][4].

Today, the most widely used unification grammars are Lexical Functional Grammar (LFG) [Bresnan, 2001], Functional Unification Grammar (FUG) [Kay, 1984], Categorial Unification Grammar (CUG) [Kartunnen, 1989], Tree Adjunction Grammar (TAG) [Joshi and Schabes, 1997], Head-Driven Phrase-Structure Grammar (HPSG) [Pollard and Sag, 1994] and Construction Grammar (CxG) [Goldberg, 1995]. For all of those frameworks computationally operational formalisms were developed and used in various applications concerning linguistic processing or grammar development, more or less successfully. The research field dealing with the development of operational grammars is computational linguistics. Grammars are approached from such a perspective in the following section.

2.2 Grammars in Computational Linguistics

The interdisciplinary research field of computational linguistics is interested in providing computational models of written or spoken natural language to automatically process natural language in, for instance, parsing, language production, or stemming (see, for instance, a standard reference by [Jurafsky and Martin, 2008]). The field of computational linguistics is tightly connected to the field of Artificial Intelligence as no language can ever be modelled completely without including learning paradigms to at least cover phenomena like language change. Language understanding or translation rely heavily on included machine learning paradigms to become efficient and scalable.

[4]Subsumption describes the ordering of feature structures. A feature structure F subsumes F' if its type is more general. Any feature f defined in F is also defined in F'. That means that the feature f in F subsumes the feature f' in F'.

Part of this linguistic engineering effort concerns itself with developing machine-readable and operational implementations of grammars. Grammars are especially needed to enable natural language processing in various application areas such as for instance in strongly Artificial Intelligence-related fields as machine translation, text understanding, or speech recognition. Therefore, the grammars have to be made explicit and then machine-readable, which means they have to be formalized and then implemented.
The following sections describe the aspects of grammar engineering which are, as already mentioned, considered as being major challenges in that field in general and mentions several existing grammar formalizations, which are considered important for this work.

2.2.1 Computational Implementations of Linguistic Formalisms

Why is there a need to computationally implement linguistic formalisms at all? Computational implementations of linguistic formalisms present in our opinion a crucial step relevant for any kind of linguistic theory to be able to validate the hypotheses and assumptions it makes. In order for a linguistic formalism to be fully operational, i.e. to be made machine-processable, to be used in parsing or producing natural language phrases or sentences or other linguistic constructions, its hypotheses need to be validated. They have to be accurate and non-contradicting, i.e. clear and precise. They need to be reproducible and unique. The issues raised here will be further discussed in more detail in the course of this thesis and especially be raised in the first sections of Chapter 3 with a specific focus of why this work is valuable to research in language processing. Section 3.3 covers the main issues of the contribution of this work and argues for a new implementation of a specific linguistic theory, namely Construction Grammar.
Let's have a closer look at different aspects that should be accounted for when implementing grammars followed by a short overview of several

grammar implementations which are fundamental for this work.

2.2.2 Implementing Grammars in General

The technical term for the complete process of developing a machine-interpretable grammar is *grammar engineering*. The discipline of grammar engineering deals with the development of grammars which can be used in natural language processing systems such as for example in spoken dialogue systems or question-answering systems. It includes the application of conventionalized methods, tools and techniques in order to come to the result – the grammar implementation. The development of large-scale grammars mostly takes over several years and is conducted in general by a number of scientists. Typically, the result of those projects is a powerful grammar formalism which is usable by those applications it was designed for.

As already mentioned in the introduction of this work, there are various issues which are of major importance within the field of grammar development (see Figure 1.2)), which are repeated in the following list and further elaborated in relation to this work:

- Applicability: The grammar has to be applicable to the domain and in the system it was designed for. Once this goal has been achieved, the question is raised how applicable it is in another domain or even in another system.

- Reusability: As grammar engineering always starts with thoroughly investigating the language in question, several steps might be obsolete if already existing sources can be incorporated, including for instance partial grammars or different domain-specific knowledge.

- Extendability/Maintainability: While engineering a model, documentation should be created to enable researchers to continue the work on the grammar or extend it - either manually or automati-

cally. This is especially important as languages change as we speak and new constructions enter the speech community continuously.

- Coverage: In case a grammar is supposed to be employed in 'real-world' applications, its coverage has to be as broad as possible. Nowadays, statistical grammar implementations dominate the market but work on symbolic grammar formalisms is promising and up to now, hybrid systems yield the best results compared to strictly statistical ones (see for instance [Van Eecke et al., 2022]).

- Relevance: Both in the field of linguistic theory and especially in the field of computational implementation, construction grammar still presents a relatively young theory. This fact leaves many issues still unresolved and open to further research. Through its formal, computational implementation, assumptions have to be made explicit and might add valuable insights to those theories. The formalism gains further visibility through additional case studies and might potentially be advanced and adopted further.

- Evaluation: Evaluation of grammar implementations is a complex endeavour and one example is to in general take into account what can be parsed and what eventually cannot. This allows a comparison of grammar implementations of different grammar theories based on their precision or coverage in parsing (and also in production in case the grammar captures the relevant rules). What should not be neglected, however, is the amount of invested man-power in designing and engineering them. With smaller grammar implementations different evaluation features can be of higher importance. A grammar formalisms crafted by a few researchers cannot be compared in coverage with a formalism engineered by a community.

- Editability: People who are not used to implement on computers might be hesitant or even 'pushed away' if there is too much implementation involved in creating a grammar. Therefore, editors

and other visual interfaces are of great help for any grammar engineer to lower the barrier and improve the user experience eventually increasing adoption.

- Representation: The format of the grammar is of major importance in case it is supposed to be employed in natural language processing systems of different kinds. Therefore, it is necessary to take into account the state of the art in knowledge representation. This will increase adoption, ease evaluation, and even coverage as useful tools might exist for e.g. editing or extending a formalism.

2.2.3 Grammar Implementations

Unification Grammar is one of the most commonly used grammatical formalisms. Within the last decades, powerful computational implementations have been engineered for several unification based formalisms.

Some examples are:

- Functional Unification Grammar (FUG) [Kay, 1984]
- Categorial Unification Grammar (CUG) [Kartunnen, 1989]
- Head-driven Phrase Structure Grammar (HPSG) [Pollard and Sag, 1987] or [Pollard and Sag, 1994].[5]
- Lexical Functional Grammar (LFG) [Bresnan, 2001, Borjars et al., 2019]
- Tree Adjoining Grammar (TAG) [Joshi and Schabes, 1997]
- Sign-Based Construction Grammar (SBCG) [Sag, 2012]
- Construction Grammar and its computational implementations Embodied Construction Grammar (ECG) [Bergen et al., 2001] and Fluid

[5] You can find a list of key publications on HPSG at [Müller, 2021]

Construction Grammar (FCG) [Steels, 2005, Steels, 2011, Steels, 2012, Steels, 2017].[6]

All of these formalisms have implemented feature structures in one way or the other and use unification (see Section 2.1.4 for definitions).
One of the main challenges of grammar engineering is that up to now no standardized methods for efficient grammar engineering exist including, for example, for the development of frameworks for grammar engineering [Uszkoreit and Zaenen, 1997, Bender et al., 2002]. Instead, it is an extremely labor-intensive mostly manual process.

The following sections shortly present the main features of three of the mentioned unification grammar implementations: *Head-Driven Phrase Structure Grammar* (HPSG), *Fluid Construction Grammar* (FCG) and *Embodied Construction Grammar* (ECG). The reasons for this choice are the following:

We chose HPSG since it is widely used and plays a major role in computational linguistics today. After almost 30 years of research, HPSG represents a formalism standing on a solid foundation and possessing wide-coverage grammars for various languages like French, English, Korean, Japanese, Spanish or German.[7]
Two of the existing construction grammar implementations are examined here since Construction Grammar per se presents a promising, relatively young grammar theory still leaving many issues open both regarding its theoretical foundation and the existing implementations of it. Yet, there is lots of room for experimenting and improving existing implementations or extend them to further applications. Additionally, since the model described in this work deals with construction grammar, too, the two already existing implementations should be examined. Their rather specific

[6] Section 3.2 presents and compares the two main computational implementations of construction grammar.

[7] See for instance [Müller, 2008] or the proceedings of the yearly HPSG conference at *http://web.stanford.edu/group/cslipublications/cslipublications/HPSG/* .

domains of application will be described in more detail in the following sections. The grammar examples in the sections on Fluid Construction Grammar and Embodied Construction Grammar have been implemented by the author.

Head-driven Phrase Structure Grammar *Head-driven Phrase Structure Grammar* [Pollard and Sag, 1987], [Pollard and Sag, 1994] can look back on more than 30 year of research. Meanwhile, it is one of the mainly used grammar formalisms in computational linguistics and various implementations exist covering different languages as for instance English, German, Chinese and Maltese.[8] HPSG is a lexicalized grammar theory, i.e. that almost all grammatical information is part of the lexicon. While the lexicon presents a richly-structured hierarchy instead of a list of entries, only a few grammatical rules are needed to take care of the constraints for processing the lexicon. It is head-driven meaning that the nucleus or head of a phrase is obligatory and determines relevant features of that phrase. As an example is the verb, for instance, the head of a verbal phrase – and not eventual objects – and determines information like number or person of the phrase. HPSG is a monostratal theory which means it has only one level of representation: the sign. A sign takes into account the function and the form of its components, as defined by [de Saussure, 1985]. HPSG uses – as all of the below described implementations of typed feature structure grammars – an implementation of typed feature structures to represent both lexicon entries and phrases.
Figure 2.3 shows an attribute-value matrix of the lexical item soccer player in HPSG notation.
All types of feature structures in HPSG are of type `sign` combining the two attributes `PHON` and `SYNSEM`. That means that each feature structure of this type contains information of these two attributes. The `PHON` attribute contains all phonological information of the feature structure, `SYNSEM` the specification of its syntactic and semantic properties. `SYNSEM`'s value is

[8] The grammar implementations for German, Chinese and Maltese can be inspected at *http://hpsg.fu-berlin.de/ stefan/Pub/hpsg-lehrbuch-grammatiken.html*

again a feature structure of type synsem containing the two attributes LOCAL and NON-LOCAL. LOCAL contains the attributes CAT (for category), containing syntactic information and CONT (for content) contains semantic information. Figure 2.3 specifies under CAT that *soccer player* is a *noun* and that its CASE is determined by the HEAD of the phrase. Case congruency between an accompanying determiner and the noun is ensured with the help of a variable (here 1) that is attached to both CASE attributes in the CAT attribute. The CONT attribute links *soccer player* to an INSTANCE X wich can be another feature structure, an ontological instance, or some other entity containing further semantic information on the item.[9]

$$\begin{bmatrix} \text{PHON} & \textit{soccer player} \\ \text{SYNSEM|LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CASE} \boxed{1} \\ \textit{noun} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{DET} \left[\text{CASE} \boxed{1} \right] \right\rangle \\ \textit{cat} \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INST X} \\ \textit{soccer player} \end{bmatrix} \\ \textit{loc} \end{bmatrix} \\ \textit{word} \end{bmatrix}$$

Figure 2.3: An attribute value matrix for the lexical item *soccer player* in HPSG notation.

Embodied Construction Grammar *Embodied Construction Grammar* (ECG) represents a formal and explicit model of construction grammar which was developed within the Neural Theory of Language project (NTL), an interdisciplinary research effort oriented towards investigating cognitive phenomena, seeking answers to the question: *How do humans learn and use language?*[10] and the EDU project (EDU).[11] It has mainly

[9] For further and fine-grained information and introductory details we refer to the well-established HPSG literature, mainly to Pollard and Sag, 1987, Pollard and Sag, 1994 and Müller, 1999 and to their website at *http://hpsg.stanford.edu/*

[10] See http://www.icsi.berkeley.edu/NTL/ and Feldman, 2006 for a detailed description of the Neural Theory of Language.

[11] http://www.eml-development.de/english/research/edu/index.php

been developed to be applied in a simulation-based language understanding system (see for instance [Chang et al., 2002, Bergen and Chang, 2005, Chang, 2008]) which – to our knowledge – unfortunately has never completely been put together. Several high-value components were developed but they never were integrated into one single system.
Embodied Construction Grammar grew out of the cognitive linguistics-based construction grammar framework. Within the framework of Lakoff's *Neural Bridging Theory* [Lakoff and Johnson, 1999], ECG represents an extension which gets more and more the status of a separate approach in the area of construction grammatical research.

While other approaches consider language as completely independent from the organism which uses it, it is displayed in ECG that several characteristics of the language user's sensorimotor system can influence his or her language. The needed dynamic and inferential semantics in ECG is represented by so-called *embodied schemas*. These schemas are comparable to and include some *image schemas* known from traditional cognitive semantics. They constitute schematic recurring patterns of sensorimotor experience [Johnson, 1987, Lakoff, 1987].
As proposed by traditional construction grammar theory, the main building blocks are constructions ranging from lexical constructions to highly complex structures.[12]
We've implemented an example ECG grammar and Figure 2.4 shows an example lexical construction for the term *soccer player*.[13] The format of ECG reflects the influence of constraint programming languages and inheritance-based ontologies: special keywords (in boldface) are used to indicate internal structure and express inheritance relations and other constraints, and dotted slot chains are used to refer to non-local structures. On the surface, ECG closely resembles a data structure.

[12] Of course it is the grammar engineer's choice how fine-grained his grammar should be and smaller constructions than lexical ones like for instance morpheme constructions are possible.

[13] More detailed information on implementation details of ECG is given in Section 3.2. For even further information we refer to the ECG literature.

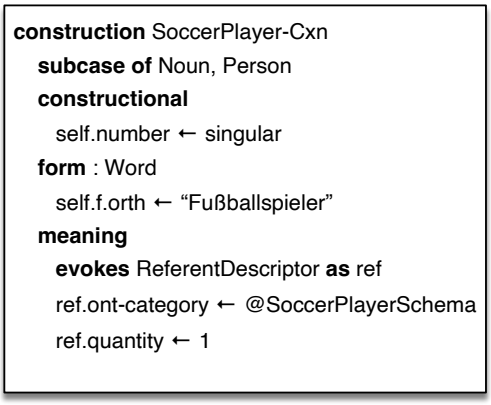


Figure 2.4: A construction for the lexical item *soccer player* in ECG notation.

The high-level structure of the construction includes three blocks, where keywords (shown in boldface) are used to indicate special terms and structures: the constructional block contains information relevant to the construction as a whole, while the form and meaning blocks (or poles) contain information relevant to each of those domains.
In fact, the SoccerPlayer-Cxn construction is defined within a larger network of structures, including other constructions to which it is related and schemas in each of the form and meaning domains. These structures will be described in more detail later, but we highlight here a few of the notations that express constructional constraints.

- The **subcase of** declaration defines a multiple inheritance hierarchy over constructions, here identifying the SoccerPlayer-Cxn as a specific kind of Noun and Person. Inheritance makes all features of a parent construction automatically available in the subcase.

- Various notations allow flexible reference to different accessible structures. The *self* keyword allows self-reference, i.e. reference to the construction currently being defined. The special names self.f and self.m refer to the construction's form and meaning poles, respectively. Slot chains of the form x.y allow reference to a non-local feature (or role) y accessible through a local structure x.

- ECG includes a fixed set of constraint types for expressing category constraints and role-filler binding. Category constraints are indicated with a colon (i.e., the form pole must be a Word), and role-filler constraints of the form x y indicate that role (or feature) x is filled by the (atomic) value y (i.e., the form pole is associated with the orthographic string shown).

- The meaning pole illustrates some additional notations specific to ECG. The **evokes** ReferentDescriptor as ref declaration indicates that there is a structure of category ReferentDescriptor present, referred to by the local name ref and discussed more below. Here we note that it has a feature ont-category that is constrained to be filled by @SoccerPlayerSchema. (The @ symbol is used to allow reference to an external conceptual ontology.)

Fluid Construction Grammar Around 2001 a congruent and parallel development has led to *Fluid Construction Grammar* (FCG), a computational implementation of construction grammar's features. Compared to other unification-based formalisms such as HPSG [Pollard and Sag, 1994] it provides a number of mechanisms that allow a more fine-grained control over the interaction between constructions. It is arguably one of the most advanced computational construction grammar formalisms and is the only one that can handle both parsing and production using the same set of constructions rather than using separate generation and parsing procedures.[14] So far, FCG has mainly been applied in research on the emergence and evolution of grammatical phenomena [Steels, 2001, Steels, 2004, van Trijp, 2008, Steels, 2017]. Current investigations include studies on how construction grammars can be learnt from semantically annotated corpora [Doumen et al., 2023] or through communicative interactions (see [Nevens et al., 2022, Beuls and Van Eecke, 2023] and also [Beuls and Van Eecke, 2024]), and how they can be used in an approach to capture narrative-based language understanding [Van Eecke et al., 2023].

[14] Please find a more detailed discussion of the importance and noteworthiness of reversible grammars in [van Trijp, 2010].

At first, the focus was on the lexical domain, which then moved further to the level of grammar [Steels, 2005, van Trijp, 2008, van Trijp, 2017]. Recent advances demonstrate FCG's application in reasoning intensive visual dialogue tasks [Verheyen et al., 2023], or visual question answering [Nevens et al., 2019].

FCG combines various ideas of different constructivists' approaches. It represents, for instance, a usage-based model of language [Langacker, 2000], and the formalism is unification-based [Kay and Fillmore, 1999]. Additionally, aspects of Radical Construction Grammar approach [Croft, 2005] are implemented, which are, for example, realized in the fact that the set of linguistic categories, semantic roles, or syntactic features are all open.

Language is seen as a complex and adaptive system that evolves over time through verbal interaction within a community [Steels, 2003]. The main goal of Steels and his team has not been to cover all existing natural language phenomena as they occur in natural languages as they are but to build models of the processes that go on in languages concerning the emergence and adaptation of new forms or meanings, i.e. constructions [Steels and De Beule, 2006]. However, several efforts in FCG have led to a number of so-called mini-grammars that focus on the modeling of various complex language phenomena in different languages as for instance in German [van Trijp, 2011, Micelli, 2012], Hungarian [Beuls, 2011] or Spanish [Beuls, 2012] showing that this can be covered with FCG, as well. Figure 2.5 shows an example lexical entry in FCG notation.

Regarding its format, it reflects the influence of the LISP programming language: internal structure is indicated with parenthesized lists employing prefix-list notation, and variable names are marked with a leading question mark. FCG structures look more like a program than a data structure.

The structure in Figure 2.5 makes use of the basic FCG def-fcg-cxn predicate for defining constructions followed by the construction's name and a list of its semantic (sem-cat) and syntactic categories (syn-cat). Leading question marks as in ?soccer-player-unit or ?ref indicate that these are

```
(def-fcg-cxn soccer-player
             ((?soccer-player-unit
               (sem-cat (referent-descriptor ?ref)
                        (ont-category soccer-player)
                        (quantity 1))
               (syn-cat  (lex-cat noun)
                         (number singular)
                         (gender masculine)))
            <-
              (?soccer-player-unit
               (HASH meaning ((find-entity ?ref [soccer-player])))
               --
               (HASH form ((sequence "Fußballspieler" ?start ?end))))))
```

Figure 2.5: The lexical construction for the German term *Fußballspieler*, i.e. *soccer player* in English, in FCG notation.

variable names. Units specify the constraints and categorizations relevant to each domain. The variable ?ref (corresponding to the Referent structure connected to the word *Fußballspieler* in the informal analysis) is of ontological category [soccer-player] (where square brackets denote reference to an ontology item) and can also be categorized as a referent-descriptor, in addition to its quantity being 1. The list of syntactic categories denotes that this construction is of category noun, its number value is singular, and its grammatical gender masculine. The construction lists its form and meaning features following the arrow (<-) preceded by the HASH operator. Detailed information on FCG's syntax and semantics is described in [Van Eecke, 2018] or can be found in the Babel Wiki.[15]

A fully operational version of FCG which is implemented in a substrate

[15] *https://emergent-languages.org/wiki/docs/recipes/fcg/syntax-and-semantics/*

of Common LISP can be downloaded for testing or conducting own experiments.[16] For further information on comparing FCG and ECG, please consult Section 3.2.

2.3 Formal Ontologies

This section gives an overview on formal ontologies. First, we introduce the concept and the various existing formats of ontologies. Furthermore, methodologies used in generally engineering ontologies are described. Section 2.3.4 gives a short overview of popular existing foundational ontologies and Section 2.3.5 concludes this Section with a short description on the representation of linguistic information in ontologies.

2.3.1 Definition of Ontology

Originally, ontology describes a philosophical discipline describing the study of being or existence. The term is used in computer science where an ontology defines the terminology, the concepts, and their relationships towards each other in a specific domain of application. Following Gruber [Gruber, 1993] "an ontology is an explicit specification of a conceptualization". It is a knowledge base where concepts and the relations among them are agreed upon and formally defined. Unlike simple taxonomies, where the single relation among the contained elements is inheritance (so-called *is a-relations*), an ontology represents a network-like structure with logical relations between its elements. The most basic relation between concepts is inheritance. Often there are axioms added that express more fine-grained relationships and constrain their interpretation (see [Guarino, 1998]).
Guarino [Guarino, 1998] defines three different kinds of ontologies (see also Figure 2.6):

- ***Top-level ontologies*:** Top-level ontologies are often called *foundational* or *upper ontologies*. In this work we will adopt the term

[16] *www.fcg-net.org*

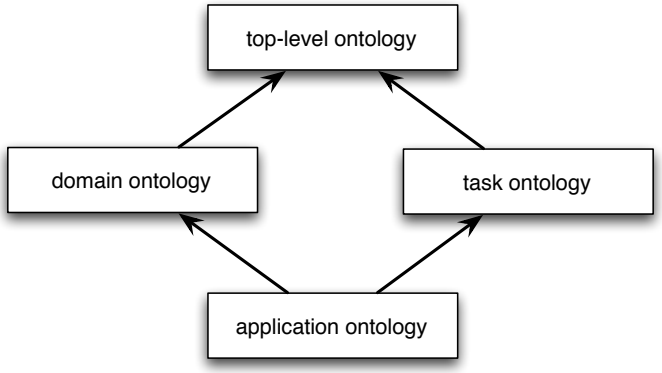


Figure 2.6: Hierarchy and types of ontologies as defined by [Guarino, 1997].

foundational ontology. They represent general, domain-independent knowledge. There exist just about a dozen freely available foundational ontologies, as, for instance, the Cyc-ontology [Lenat, 1995], SUMO[17] or DOLCE[18]. [Oberle, 2006] gives a detailed overview on foundational ontologies. Section 2.3.4 briefly sketches out the most popular ones.

- ***Domain* or *Task ontologies*:** Task ontologies define the terminology of a specific domain or application area, whereas different experts have to agree upon a common understanding of the terminology and its meaning. An example is the Financial Industry Business Ontology [FIBO, 2022] or the Ontology for Strongly Sustainable Business Models [Upward and Jones, 2016].

- ***Application ontologies*:** Application ontologies model a limited part of the real world. They can be compared to traditional data models.

[17] http://ontology.teknowledge.com/, last checked May 20, 2022
[18] http://www.loa-cnr.it/DOLCE.html, last checked May 20, 2022

2.3.2 Ontology Formats

RDF/RDFS The RDF (short for *Resource Description Framework*) data model[19] has been developed by the W3C to be used to make statements about resources in the Semantic Web. Those statements are usually subject-predicate-object expressions. It enables the description of properties of resources of the world wide web in a machine readable and processable way. It is the basis of OWL which will be described briefly in the following paragraph.
RDFS (short for *RDF schema*[20]) allows to describe complex relations between RDF resources. An `rdf:property` is a relation between a subject resource and one or more object resources. The two instances of `rdf:property` of highest interest for this work are `rdfs:domain` and `rdfs:range`. `rdfs:domain` states that any resource with a given property is an instance of one or more specified classes, and `rdfs:range` is used stating that the values of a property are instances of one or more specified classes. Examples can be found in Section 4.4.

OWL OWL (short for *Web Ontology Language*)[21] is another W3C standard which is technically based on RDF. The main difference of OWL compared to other ontology languages as for instance DAML+OIL or RDF is that it includes more powerful operators as for instance *and*, *or* or *negation*. It is based on a different logical model which allows not only to define concepts but also to describe them. Additionally, the use of reasoners (as e.g. *Pellet* [Pellet, 2022]) is possible which checks the ontology's consistency. One can distinguish between three different kinds of OWL versions:

- **OWL-Lite**:[22] This version is syntactically the simplest one. In general, it is used in cases where simple class hierarchies and constraints

[19] http://www.w3.org/RDF/, last checked May 20, 2022
[20] https://www.w3.org/TR/rdf-schema/, last checked May 20, 2022
[21] http://www.w3.org/TR/owl-features/, last checked May 20, 2022
[22] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3, last checked May 20, 2022

are needed.

- **OWL-DL:**[23] DL stands for Description Logic which constitutes OWL-DL's basis. It is more expressive than the Lite version and allows the use of ontology reasoners to verify the consistency of the ontologies. Description Logics enable knowledge representation in a well-formed and structured way. Description Logics is equivalent to First Order Logic.

- **OWL-Full:**[24] OWL-Full is the most expressive version of the different OWL languages allowing predicate logics on a higher level. In return the price to pay is that using reasoners is not possible anymore since ontologies being in OWL-Full become partially undecidable, i.e. not everything can be calculated anymore.

2.3.3 Ontology Engineering in General

Engineering an ontology is a cumbersome and time-consuming task, especially, when done manually. Therefore, there is quite some literature suggesting supportive guidelines (see e.g. [Noy and McGuinness, 2001, Antoniou and van Harmelen, 2004, Falbo, 2014]). The subsequent steps are suggested by [Noy and McGuinness, 2001]:

1. **Determine the domain and scope of the ontology**: The decisions made in this step determine exactly which domain the ontology will cover and for which applications the ontology will be useful. To determine the scope it is suggested to ask so-called competency questions [Grüninger and Fox, 1995], which are questions a system using the ontology should be able to answer.

2. **Consider reusing existing ontologies**

[23] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s4, last checked May 20, 2022

[24] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s4, last checked May 20, 2022

3. **Enumerate important terms in the ontology**

4. **Define the classes and the class hierarchies**

5. **Define the properties of classes**

6. **Define the facets of the slots**: Here, constraints need to be determined as, for instance, the allowed values of a slot, the classes' cardinality and which relations are allowed between which classes.

7. **Create instances**

2.3.4 Foundational Ontologies

To get an overview on several freely available foundational ontologies, we refer the interested reader to, for instance, [Oberle, 2006]. Examples for popular foundational ontologies are DOLCE [Masolo et al., 2005, Borgo and Masolo, 2009], the Object-Centered High-Level Reference Ontology (OCHRE) [Schneider, 2003], OpenCyc (*https://cyc.com/*), and Suggested Upper Merged Ontology (SUMO) [Niles and Pease, 2001].
As a more detailed description of each of those foundational ontologies is not in the scope of this work, we continue with a short description of DOLCE, which is the foundational ontology that is chosen as a base of our model. Please refer to the mentioned literature for more information on the other ontologies.

DOLCE DOLCE - which is the first module of the WonderWeb library of foundational ontologies [Masolo et al., 2005, Borgo and Masolo, 2009] - is based on first order logic (OWL-DL) and has a strong cognitive bias. That means, that it was designed based on the underlying ontological concepts of natural language and human commonsense. Therefore, it is a so-called *descriptive* ontology as opposed to *revisionary* ontologies which may impose that only entities extended in space and time exist (see [Masolo et al., 2003] for a more detailed discussion). DOLCE has already been applied in different domains such as law [Gangemi et al., 2003b],

biomedicine [Battaglia, 2004] or has been explored to model the variety of engineering features [Sanfilippo and Borgo, 2015].

The fundamental distinction that is made within DOLCE is that between `Perdurants` and `Endurants`. While `Perdurants` describe processes or events, `Endurants` define objects or substances. The main relation between `Perdurants` and `Endurants` is that of participation, which means that an `Endurant` participates in a `Perdurant` as, for example, a person (i.e. an `Endurant`) participates in his or her life (which is a `Perdurant`).

Furthermore, DOLCE defines `Qualities` and `Abstracts`. `Qualities` include basic entities that can be measured or perceived as, for example, size, shape, color, sound or smells. The class of `Abstracts` includes Regions as, for example, time or space. DOLCE's basic concept hierarchy is partly depicted in Figure 2.7.

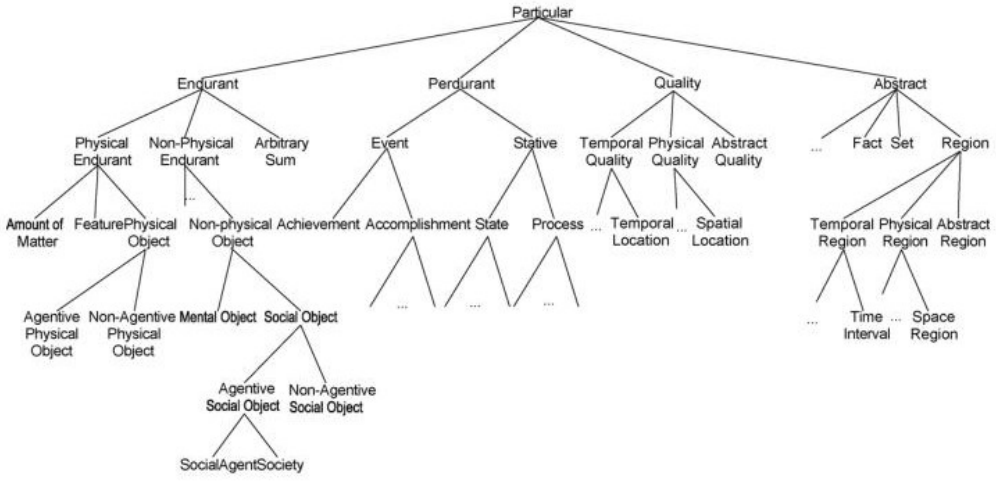


Figure 2.7: DOLCE's basic concept hierarchy [Masolo et al., 2005].

2.3.5 Representation of Linguistic Knowledge in Ontologies

Currently, linguistic information of ontology classes and properties as listed below is mostly missing or represented in impoverished ways in ontologies, leaving the semantic information in it without a grounding to the human cognitive and linguistic domain.
The following list gives examples of useful linguistic information of ontology classes and their properties:

- **Part-of-speech:** Representation of the part of speech of the head of the term like *noun*, *verb* or *adjective*.
- **Language-ID:** ISO-based unique identifier for the language of each term
- **Morphological and syntactic decomposition:** Representation of the morphological and syntactic structure (segments, head, modifiers) of a term.
- **Part-of-speech-specific information:** Representation of for instance the case of a noun, the person or number of a conjugated verb and so forth.

The subsequent paragraphs briefly sketch out various existing approaches tackling those issues and are published in a research paper by the author et al. [Buitelaar et al., 2006].

Simple Knowledge Organization Systems (SKOS) One approach to express linguistic information in an ontology is the SKOS format.[25] It enables the formalized representation of different types of controlled vocabularies as for instance taxonomies or thesauri. However, there is a technical and conceptual reason why SKOS does not provide the linguistic information we wish our model to have as previously listed above:

[25] *http://www.w3.org/TR/swbp-skos-core-guide/*

On the technical side, SKOS uses sub-properties - `skos:prefLabel` and `skos:altLabel` - of `rdfs:label` together with `xml:lang` to attach multilingual terms to concepts. The RDFS specification defines the range of `rdfs:label` to be `rdfs:Literal`. In the definition of `rdfs:subPropertyOf` it is determined that the range of `skos:prefLabel` and `skos:altLabel` is also (or a specialization of) `rdfs:Literal`. As we wish to attach more linguistic information to ontology classes than simply attaching multilingual strings, we consider this approach as insufficient for the current scenario. The conceptual problem we see with SKOS for the use in our scenario is that it mixes linguistic and semantic knowledge. SKOS uses `skos:broader` and `skos:narrower` to express "semantic" relations without clearly stating the semantics of these relations intentionally, and defines the subproperties `skos:broaderGeneric/narrowerGeneric` to have class subsumption semantics (i.e., they inherit the `rdfs:subClassOf` semantics from RDFS).

Wordnets and OntoWordNet Alternative lexicon models concentrate on the definition of a top ontology for lexicons instead of linguistic features for domain ontology classes and properties, see e.g. [Bateman et al., 1995] and [Alexa et al., 2002]. Similarly, the proposed OntoWordNet model [Gangemi et al., 2003a] aims at merging the foundational ontology DOLCE [Gangemi et al., 2002] with WordNet to provide the latter with a formal semantics.

Lexical Markup Framework Some initiatives of the ISO TC37/SC4[26] working group on the management of language resources continue the work from previous standardization initiatives, like EAGLES (Expert Advisory Group on Language Engineering Standards) for morphological and syntactic annotation and ISLE (International Standards for Language Engineering) for the representation of lexicon entries. In the various initiatives of ISO TC37/SC4 the focus is on the more abstract level of meta-annotation and of frameworks supporting the creation and the exchange of anno-

[26] *https://isotc.iso.org/livelink/livelink?func=llworkspace* (checked May 28, 2022)

tations, data structures and resources. An important part of this work consists of the definition of procedures for the creation and maintenance of data categories for various annotation frameworks. Data categories are formalized representations of the most relevant linguistic concepts, such as part of speech, or lemma and so forth.

None of the briefly introduced methods of displaying linguistic information of ontology classes or properties serve our purpose completely. Wordnets, for instance, lack the linguistic features that are needed and SKOS mixes linguistic and semantic knowledge. Therefore, the decision has been made to use a model that the author and others designed within the SmartWeb project [Wahlster, 2007] for exactly the purpose that is needed in this work. The model is called LingInfo [Buitelaar et al., 2006] and will be presented in detail in Section 3.5.

2.4 Frames and Schemas

Having had a brief look at what constitutes a construction, let's in this section focus on that side of a construction that constitutes its meaning. Let's have a look at what can possibly fill the meaning pole of a construction and what is used later in the engineering portion of this thesis.

The section starts with a short description of two different types of semantic schemas. Then Frame Semantics is briefly introduced together with its online application, i.e. the FrameNet database [Baker et al., 1998] The section ends with a short overview of the *kicktionary*, a semantic database based on the FrameNet project and on the work of Seelbach [Seelbach, 2001] and Gross [Gross, 2002].

2.4.1 Schemas

Schemas are (mental) structures listing most important features of an aspect of the world. In contemporary cognitive linguistics, they are consid-

ered crucial structures of experience and we believe they can be valuable being used in natural language processing and reasoning. Two instances of schemas will be described in further detail below: Executing schemas and image schemas.

Executing Schemas

Research in language processing or robotics and AI have shown that active models of actions and events are needed to act quickly in an uncertain and dynamic world. To model these kinds of actions, executive schemas, (or x-schemas) have been defined [Feldman et al., 1996, Narayanan, 1997] that allow a system to draw inferences about actions or events quickly as they are representations tightly coupling action, execution monitoring, error-correction or failure recovery. To perform a simulation, it executes an internal model of the action or event in question. In the context of language understanding, x-schemas are mainly represented as Petri nets [Murata, 1989] which are models of discrete, distributed systems used in all kinds of areas of research that need modelling of e.g. business processes or theoretical biology. Basic Petri nets are weighted, bipartite graphs consisting of connected places containing tokens that represent a predicate of the world and transitions. The connections are input and output arcs. Tokens are moved from input to output places when a transition fires.

Image Schemas

Image Schemas constitute schematic recurring patterns of sensorimotor experience [Johnson, 1987, Langacker, 1987], i.e. they enable the mapping of spatial onto conceptual structure. Since 1980, the notion of *image schema* has been the touchstone notion of cognitive linguistics (see, for instance, [Lakoff and Johnson, 1980]).

Thorough investigations gave evidence for common mechanisms for action and perception (so-called mirror neurons) in the F5 area of the human brain [Rizzolatti et al., 1996]. What can be claimed based on these

findings is that natural language understanding in humans is active simulation. Schematic knowledge, then, might provide a framework whereby the schematic roles assigned to various values can serve as simulation parameters and help natural language systems improve their performance. That image schemas are actually present in our speech can, for example, be seen in their use in spatial utterances or prepositions [Kuhn, 2005].
Typically, when being depicted, a schema consists of its name and a list of roles. Example image schemas are for instance the `Containment schema`, the `Near-Far schema` or the `Source-Path-Goal schema` as depicted in Figure 2.8.

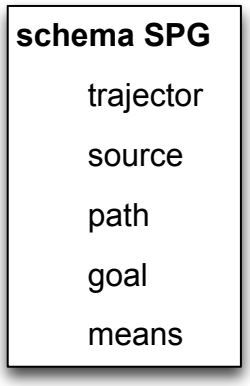


Figure 2.8: The Source-Path-Goal Schema

The `Source-Path-Goal` is called **schema SPG** followed by the list of its roles `trajectory`, `source`, `path`, `goal` and `means`.

2.4.2 Frames

According to standard Frame Semantic theory, frames are abstract 'scenes' or 'situations' needed for understanding the semantic structure of a word [Fillmore, 1982, p. 115]. They are encyclopedic cognitive structuring devices, parts of which are indexed by words associated with it and used in the service of understanding [Fillmore, 1985a].
Frames are the basic building blocks of Frame Semantics [Fillmore, 1982] and designate concepts (comparable to *schemas* or *scripts*). The structure

is based on Fillmore's claim that the meaning of words should be described against the background of connected knowledge.
Frame Semantics and Construction Grammar are complementary frameworks of contemporary linguistics. Although their relation has already been studied extensively [Fillmore, 1985b, Matsumoto, 1989, Fujii, 1993, Goldberg, 1995, Petruck, 1996, Östman and Fried, 2005, Goldberg, 2006a, Boas, 2008, Verhagen, 2009], it still lacks a precise operationalization. A full operationalization should ideally include the integration of Frame Semantics with Construction Grammar in a computational fashion, i.e. in the form of an automatic parser/generator of sentences into and from their frame semantic meanings.
One of the computational implementations of construction grammar, ECG, has specifically been designed to support a frame-based semantic representation. Section 3.2 will go into further details of this topic.

FrameNet

Within this spirit, the Berkeley FrameNet project [Baker et al., 1998] has built an online database documenting the range of semantic and syntactic combinatory possibilities (valences) of each word or lexical unit in each of its senses.
The FrameNet database contains more than ten thousand English lexical units and, more recently, similar efforts for several other languages such as German, Spanish and Japanese are made. All lexical units are annotated with their semantic frames based on example sentences in which the respective frames and frame elements are marked. The corpus-oriented approach of the FrameNet project makes that it is tightly connected to empirical observations. It offers linguists a very valuable tool for exploring theoretical issues and challenges in applied linguistics that require text understanding or production facilities and in principle, this provides the basis for automatic information retrieval, language generation and intelligent machine translation [Boas, 2002, Wilks, 2009b].[27]

[27] For more information see *http://framenet.icsi.berkeley.edu/*.

The Kicktionary – a multilingual electronic dictionary containing soccer-specific language – has been created based on FrameNet and is shortly described in the section below.

The Kicktionary: Description and Motivation for its Usage

The Kicktionary is a multilingual online dictionary containing soccer-specific language.[28] The Kicktionary is, however, much more than an ordinary electronic lexicon: It is a semantic database based on Frame Semantics, using the methodology of the FrameNet project [Baker et al., 1998] and the work of Seelbach [Seelbach, 2001] and Gross [Gross, 2002].
All in all, the dictionary contains about 1900 lexical units, which are grouped into about 100 frames. Those frames can in turn be grouped into sixteen different scenarios. An extensive list of all of the Kicktionary's frames and scenarios can be found on the website. Figure 2.9 lists the sixteen scenarios and gives one example frame per scenario.
The reasons why this semantic database is of interest for this work can be summarized as follows:

1. It contains the most important words, phrases, and other constructions regarding soccer language in the three different languages German, English and French (initially we are only interested in the German subpart of the data).

2. Its structure is not like the structure of any "regular" electronic dictionary but it is similar to FrameNet's structure.

How we actually make use of the Kicktionary and its particular structure in the ontological framework that is going to be engineered, will be described in detail in Section 4.7.
Let's now dive into the following chapter where the decisions that were taken are being discussed regarding:

[28] You can find the Kicktionary and all information related to it on its website at *http://www.kicktionary.de*.

Figure 2.9: Scenarios of the Kicktionary with example Frames.

- Grammar theory: What has been picked for the grammar engineering effort described in this work?
- Components: What are the specific components of the grammar engineering effort?
- Advantages and disadvantages: What are the pros and cons of the decision making process?

Chapter 3

Taking our Pick

This chapter presents the details on the decision-making process on which formalism to chose as a linguistically theoretical foundation for the model that is being built later on. It describes the motivation behind the reason why construction grammar has been selected, starting with a detailed description of construction grammar's current trends and detailing out the main characteristics of Vanilla Construction Grammar.
Subsequently, it presents a detailed comparison of the two existing representational formalisms FCG and ECG, highlighting their representational choices with the help of a simple example. The section that follows that comparison provides the main motivation and merits of a new formalization instead of adapting or improving one of the previously presented formalisms.
Then, the foundational ontology is described that has been chosen as the ontological basis of the grammar model. The subsequent section describes the LingInfo model, that we have designed to present linguistic information of ontology classes and properties, while the chapter ends with a description of further modeling decisions, which are assumed as being known in the subsequent chapters.

3.1 Searching for the Right Grammar Framework: Construction Grammar

Chapter 2 outlined the various approaches to grammars both in theoretical and in computational linguistics. As mentioned, a constraint-based formalism is of advantage when dealing with natural language data. Of the existing unification-based grammars we consider construction grammar (see for instance [Fillmore and Kay, 1987], [Langacker, 1987], [Lakoff, 1987], [Goldberg, 1995], [Croft, 2001], [Goldberg, 2006b]) as being the right grammar framework for applicable NLP systems since it comprises several advantageous features which will be detailed in this section.
Construction grammar originates from earlier insights in functional and usage-based models of language mainly proposed by cognitive linguists in the late 1980s (see [Lakoff, 1987], [Fillmore and Kay, 1987], [Talmy, 1988], [Kay, 2002]). Over the last decades, the construction grammar framework has been gaining more and more attention. This is mostly due to the fact that language phenomena that could not be dealt with earlier in other grammar frameworks (as in for instance traditional generative grammar [Pinker, 1989]) suddenly can be described in a simpler and more plausible way.
Another reason is that the usage-based approach construction grammar offers, represents a way to describe early language development in a satisfactory way [Tomasello and Brooks, 1999], [Tomasello, 2003], [Tomasello, 2014].
The term *construction grammar* still does not designate a completely defined lingustic theory but stands for a variety of approaches of different flavours of a grammar framework. Still it remains unclear which of those approaches will develop into a well defined theory or which will converge. However, exchange and discussion of the approaches is very dynamic.

We see following three main streams in construction grammar to be distinguished (see [Fischer and Stefanowitsch, 2006]):

1. The first approach is strongly based on Frame Semantics when it

comes to the representation of the semantic pole of a construction [Fillmore, 1982] (see Section 2.4) and is mainly represented by works of Charles Fillmore and Paul Kay [Fillmore and Kay, 1987, Fillmore, 1988, Kay, 1997]. Concerning syntactic aspects this approach is getting more and more similar to Pollard's and Sag's *Head-driven Phrase Structure Grammar* (HPSG) [Pollard and Sag, 1994] (see Section 2.2.3).

2. The second main approach has a strong cognitive linguistic bias and is mainly advocated, amongst others, by George Lakoff [Lakoff, 1987] and Adele E. Goldberg [Goldberg, 1995, Goldberg, 2006b]. This approach, as well, tends to be based on Frame Semantics. However, it additionally includes ideas of Generative Semantics [Lakoff, 1987], Cognitive Grammar [Langacker, 1987, Langacker, 1991], Natural Semantic Metalanguage [Wierzbicka, 1996], Lexical Functional Grammar [Bresnan, 2001] and Conceptual Semantics [Jackendoff, 1983]. *Embodied Construction Grammar* originates from this approach.

3. Croft's *Radical Construction Grammar* [Croft, 2001] comprises the third approach. It is typologically motivated. Croft basically claims that all formal grammatical structures are language or construction-specific and that there is no such thing as, for instance, universal syntactic categories.

As different as they are, all of those different approaches feature exactly one formal data type which is a so-called *construction* and do not employ rules as e.g. tranformational rules. Therefore, all variations of construction grammar are non-derivational and monostratal. Monostratal means that there is one single description level (i.e. one *stratum*). Furthermore, construction grammars are not modular since constructions constrain both form and meaning.

This work uses Vanilla Construction Grammar as its basis. In the following, the main characteristics of Vanilla Construction Grammar will be

introduced, featuring all characteristics that the previously mentioned different constructivist approaches have in common[1] [Croft, 2005] and at the same time what distinguishes them from other grammar theories as for instance from projection-based theories like HPSG (see Section 2.1.4) or LFG [Bresnan, 2001].

In Vanilla Construction Grammar every level of a language, i.e. syntax, semantics, pragmatics, discourse, morphology, phonology, and prosody, are considered equal and can be mapped to corresponding constructions. This fact yields one main difference between the two streams called West Coast Grammar [Langacker, 1987, Lakoff, 1987] and East Coast Grammar [Chomsky, 1965, Katz, 1972], i.e. generative approaches: construction grammar offers a vertical - not a horizontal - organisation of any knowledge concerning a language's grammar. That is, that generative grammars split form from function: form is constituted by formal components of the grammar as for instance syntax, morphology or a lexicon, and the conventional function is defined by semantics. Constructions, in contrast, can unify several of these levels as displayed in Figure 3.1.

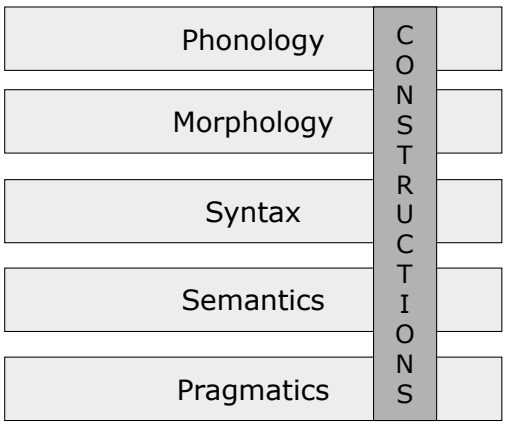


Figure 3.1: Construction grammar's position in a language's grammar. Constructions can exist on every level of a language.

In Langacker's terms all constructions of a language form "a structured inventory of the speaker's knowledge of the conventions of their languages"

[1] A very clear and detailed overview of the similarities and main differences of the constructivists' theories (in German) is given in [Fischer and Stefanowitsch, 2006].

[Langacker, 1987, p.73-76] [Goldberg, 1995, p.67]. This inventory is structured as a network, i.e. there are at least taxonomic links among the constructions [Diessel, 2004] which represents a relevant difference in contrast to generative grammars.

There are various definitions of the term *construction*. We adhere to the one by Goldberg [Goldberg, 1995, p.4] where it is stated that

- "C is a construction iff_{def} C is a form-meaning pair $< F_i, S_i >$ such that some aspect of F_i or some aspect of S_i is not strictly predictable from C's component parts or from other previously established constructions."

The constructions' form (F_i) contains its formal components, e.g. syntactical or morphological information, while S_i contains the meaning components. Instead of e.g. Lakoff's definition of a construction (see [Lakoff, 1987, p.467]) the additional condition of non-compositionality of a construction is included in Goldberg's definition. See the example for the need of a caused-motion construction to fully define the meaning of the example sentence in (1).
To define constructions in simple words it can be said that constructions, in general, are form and meaning pairings, whereby a form can be any linguistic unit, from phonemes, morphemes to clauses, and its meaning is represented by a conceptual schema, consistent with the definition of schemas in cognitive semantics. There is the discussion if *meaning* is the right term to describe the whole range of what this part of any construction can express. Some construction grammarians preferably use the term *function* instead (see also [Goldberg, 2006b, p.214]). However, Croft [Croft, 2001, p.19] uses the term *meaning* but states that it is "intended to represent all of the conventionalized aspects of a construction's function...". Basically, the definition depends on the definition of *meaning*. We follow Croft's suggestion and use *meaning* for all possible functions of a construction including semantic and pragmatic aspects.

Construction grammars have originally been devised to handle actually occurring natural language, notoriously containing non-literal, metaphorical, elliptic, context-dependent or underspecified linguistic expressions. This fact constitutes one of the major reasons why we consider it to be the right grammar formalism for use in natural language processing systems. There are theories that consider linguistic rules as algebraic forms which help in combining morphemes or words, not providing the newly formed structure with any additional meaning. Construction grammar, however, considers every construction itself on every level of language analysis as a meaningful linguistic symbol. Each construction presents a template for how linguistic signs can be used during communication. To give one concrete more complex example, the Passive-Construction is briefly described following Tomasello. Tomasello cites this specific construction since the subject of a passive sentence is not the agent, which typically is the role the subject plays, but the patient of the ongoing process [Tomasello, 2003, p.5f.]. Therefore, it can be claimed that the meaning of a sentence is not the meaning of its component words but of the used constructions, i.e. again, there is no explicit separation between syntax and semantics.
This claim, that grammatical phenomena contribute to the semantics of a sentence, as well, can be evidenced further with one of Goldberg's famous example sentences [Goldberg, 1995, p.29] as briefly described in Section 2.1.1 and summarized here:

(1) *He sneezed the napkin off the table.*

The whole meaning of this sentence cannot be gathered from the distinct meanings of the discrete words. For example, the direct object *the napkin* is not postulated by the intransitive verb *to sneeze*. Grammar theories adhering to the principle of lexical projection, however, would postulate a new meaning of the verb *sneeze*, determining that it even needed three arguments: `X causes Y to move Z by sneezing`. In an analog way, new meanings for each verb used in another environment as it normally is expected in would be posited, resulting in massive verb polysemy. The construction account as described by Goldberg, for instance, therefore, avoids

implausible verb senses (as just mentioned for the example verb *sneeze*). According to Goldberg the additional meaning of caused motion which is added to the conventional meaning of the verb *sneeze* is offered by the respective caused-motion construction. Additionally, she states that the argument structure of sentences, therefore, cannot be determined solely by the verb of a sentence - as it is claimed in grammar frameworks as HPSG, LFG or Role and Reference Grammar [Van Valin and LaPolla, 1997] but is determined by higher-level constructions.
A case study illustrating a field topology approach within the context of Fluid Construction Grammar focuses in particular on the double object construction in ditransitive sentences. The completely operational solution shows, how, for instance, German constituent ordering determines information structure of a sentence [Micelli, 2012].
The third characteristic that Croft states as being one of vanilla construction grammar[2] is that syntactic constructions are different to their schematicity in contrast to being, for instance, substantive, specific or lexical. "A more schematic construction describes a complex structure [...]" [Croft, 2005, p.2]. This complex structure is a construction in which other constructions are combined into one unit, as, for example, the caused-motion construction mentioned by Goldberg or the transitive argument linking construction, which is "a maximally schematic syntactic unit" [Croft, 2005, p.2].
What follows is that individual constructions can differ along three dimensions [Langacker, 2003] which are sketched out below:

- **Generality:** describes the extent to which the constructional schema is schematic rather than specific, e.g. highly specific constructions are lexical expression as found, for example, in early child language or idioms.

- **Productivity:** describes the extent to which a constructional schema

[2] *Vanilla* as in the sense of something without any additional and optional extras, i.e. something basic, the simplest version of something. See also *https://www.merriam-webster.com/dictionary/vanilla*.

is accessible for sanctioning new instances, e.g. so-called extensions via analogy or reanalyses.

- **Compositionality:** describes the extent to which the meaning and form of the whole are predictable from those of its parts in accordance with corresponding sanctioning schemas.

Especially in the light of language variation and change - and, therefore, robust and scalable natural language understanding - it is important to note that constructions can change their location over time or text in this three dimensional space in any direction.
A nice natural language example is the phrase *be going to*. The phrase changed through the process of reanalysis from a purposive directional construction with a non-finite complement as, e.g., in *Jane is going to marry Bill* to the auxiliary *be going to* which involves then the change of aspect from progressive to future, since there already is an inference of futurity from purposives [Hopper and Traugott, 2003]. The *be going to* construction is nowadays more general than it has been before the reanalysis process.
The following section compares the two existing computational construction grammar frameworks FCG and ECG with the help of a simple example phrase, thereby highlighting their main features, similarities and differences.

3.2 Comparing ECG and FCG: A Case Study

This section compares the two computational frameworks Fluid Construction Grammar (FCG) (see, among others, [Steels, 2005], [Steels, 2011], [Steels, 2012]) and Embodied Construction Grammar (ECG) (see among others [Chang et al., 2002], [Bergen and Chang, 2005], [Chang, 2008]).[3] Both of these representational formalisms are quite similar in terms of their expressive capabilities and basic operators and are rooted in the construction

[3]Parts of this section have been published by the author et al. here [Chang et al., 2012].

grammar tradition, sharing basic assumptions about the nature of linguistic units and the crucial role played by contextual factors. Nonetheless, they have arisen from different perspectives and with different goals: FCG was designed to support computational language game experiments that address the evolution of communication in populations of (robotic) agents, as well as for modeling language change and can be employed both for language understanding and generation, while the main focus of ECG has been on language understanding and supporting cognitive modeling of human language acquisition and use, with a focus on natural language interpretation and not generation.

Each formalism is also the centrepiece of a broader scientific framework tackling the fundamental issues above, albeit from different perspectives and with different goals. While these endeavors are theoretically compatible, they nonetheless have somewhat different orientations that have in turn given rise to some differences in their respective formalizations. Some of these differences may be described as superficial notational variations in formal devices, while others reflect more substantial divergences in emphasis, implementation or interpretation. It will be presented how these differing emphases motivated different design choices in the two formalisms and illustrate the linguistic and computational consequences of these choices through a small case study. Results of this comparison may sharpen issues relevant to computational construction grammar in general and may hold lessons for broader computational investigations into linguistic phenomena. They also might offer open research issues supporting the modeling project that will be presented in Chapter 4. Section 3.3 addresses the motivation and merits of a new representational formalism of construction grammar in detail.

Shared theoretical and methodological commitments Before turning to the case study, some basic theoretical commitments shared by the two research frameworks under consideration are briefly summarized. Broadly speaking, both are identified with constructional, cognitive and usage-based approaches to language. Constructions are taken to be the

basic units of language, and meanings correspond to particular ways of conceptualizing or construing a situation. Language is also assumed to be inherently embodied, grounded and communicative: language users have sensorimotor capacities that shape their conceptual categories, and they are grounded in particular environments with specific communicative goals.

Most relevantly, both formalisms were designed to support working systems that actually instantiate structures and processes that are elsewhere typically described only discursively. This commitment to supporting language use means that it is not sufficient merely to represent linguistic knowledge in formal notation; rather, the processes that interact with that knowledge must also be specified, and considerations related to processing (e.g., search space, storage, efficiency) must guide representational choices at the level of both individual constructions and the formal notation itself. Linguistic representations in both frameworks are also assumed to interact closely with structures in other domains, including in particular embodied, situational and world knowledge. The two frameworks differ in the details of how such interactions are modeled, and even in how terms like embodiment are used. This will be briefly discussed in the paragraph below.

For the purposes of this section, however, we mainly focus on the specifically linguistic knowledge expressed by the two grammatical formalisms and their mechanisms of use. Both frameworks take these to be conceptually distinguishable from the details of sensorimotor representations; world (ontological) knowledge; general mechanisms of inference and belief update; and specific mechanisms of contextual grounding and reference resolution. We will also refrain from addressing in detail how language learning is modeled in each framework, though we will highlight some connections to these topics where relevant.

Embodiment The term *embodiment* describes the interaction of the body and the mind. During the 1980s mainly Rodney Brooks (see e.g. [Brooks, 1991]) brought the *embodiment theory* into the research area of

Artificial Intelligence. Questions were raised as, for example, if simple movements could influence decision-making and studies were conducted that confirmed that assumption. Especially in Cognitive Science the emphasis on the body increased.

Nuñez defines full embodiment as "an embodied-oriented approach that has an explicit commitment to all of cognition, not just to low-level aspects of cognition such as sensory-motor activity or locomotion (lower levels of commitment)" [Nunez, 1999, p.42]. Dourish [Dourish, 2001] discusses *embodied interaction* – rejecting the strict separation between body and mind[4] – a novel approach to the design of user interfaces and interactive experiences of computation where, for instance, user interfaces are moved into the real world and become tangible.

In this work, we follow his definition – similar to ECG's and FCG's understanding of embodiment – and belief that language is inherently embodied, situated and communicative: language users have sensorimotor capacities that shape their conceptual categories, and they are grounded in particular environments with specific communicative goals. Both formalisms are understood to provide a narrow interface to richer embodied representations and ontological world knowledge.

Acquisition One of the stronger beliefs of any constructivist approach concerning language acquisition is that language emerges through the speaker's both physical and social experience. Furthermore, it additionally develops through language contact. This is what is meant when it is stated that construction grammar is a usage-based model of grammar. The dynamic character of constructions and schemas and also their different levels of abstraction resemble that feature in both formalisms.

[4](Mind-body dualism as discussed in early philosophy and strongly defended by René Descartes.

3.2.1 Informal Example Constructional Analysis

The German nominal phrase "der Fußballspieler" (engl. the soccer player) will serve as an example phrase which will be informally analyzed in the following. A traditional analysis might identify a determiner (der; engl. the), a noun (Fußballspieler, engl. soccer player) and a noun phrase (NP) which combines the determiner and the noun in their appropriate order. For a construction-based approach, it is crucial to consider the utterance's meaning as well: a speaker uttering "der Fußballspieler" is engaging in an act of reference, picking out an individual soccer player uniquely identifiable to the hearer in the current discourse context.
A straightforward constructional analysis might have the structure shown in Figure 3.2, with three constructions:

- DER: The word form *der* constrains the referent to be uniquely identifiable to the hearer in context; other determiners may restrict features like number (*einige Fußballspieler*, engl. *some soccer players*) or case (*des Fußballspielers*, engl. *of the soccer player*).

- SOCCERPLAYER: The word form *Fußballspieler* constrains the referent's ontological category to a soccer player and its grammatical case to not be genitive in singular; other nouns might specify a different case (*Fußballspielers*) or otherwise constrain qualities of the referent (for instance, semantic role, gender, animacy, or countability).

- DETERMINEDNP: This phrasal construction has two constituents, corresponding to the two constructions above. It imposes a word order constraint (*der* must precede *Fußballspieler*), and its meaning is a referent in context—in fact the same referent constrained by the two constituents. Here, the relevant constraints do not conflict; in general, such compatibility or agreement in features must be enforced between determiners and nouns (hence, e.g., **der Fußballspielers*).

The constructions in the middle of Figure 3.2 reflect the phrase's constituent structure, mirroring that of a traditional syntactic parse tree

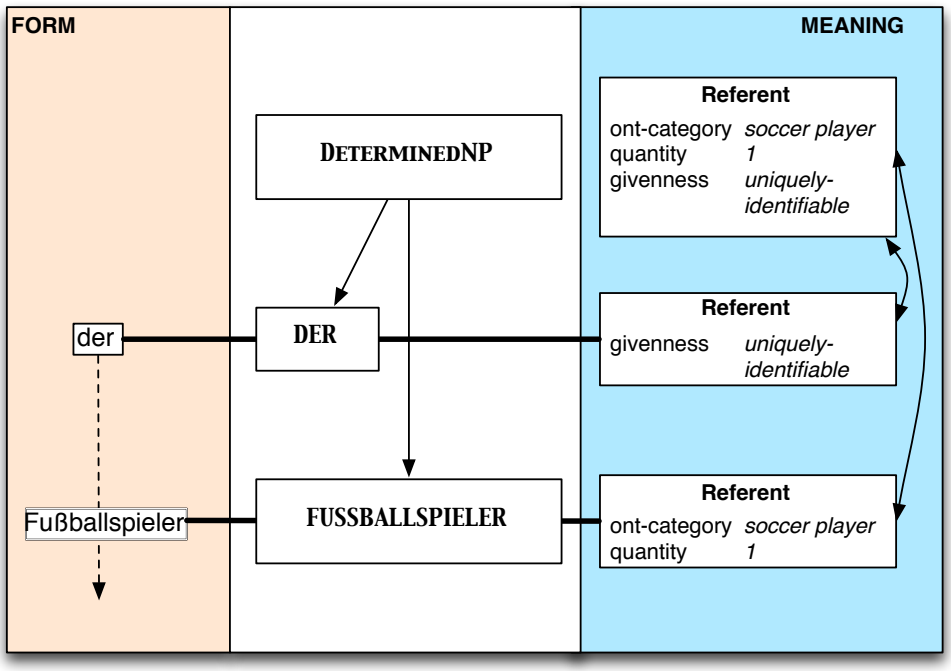


Figure 3.2: Graphical depiction of an informal analysis of an example noun phrase. Constructions (in the center) link the domains of form (left) and meaning (right). Each of the constructions shown here (the DETERMINEDNP construction and its two constituents, the DER and FUSSBALLSPIELER constructions) contributes to and constrains the particular referent specified by the phrase.

(based on a phrase structure analysis). However, since these are not just syntactic symbols but constructions, each construction also has a link (shown by horizontal bars) to form (on the left) and meaning (on the right). The form domain contains the relevant word forms, where the dotted arrow indicates the time line (and therefore word order).

The meaning domain contains several structures labeled Referent, each listing features constrained to particular values (where ont-category is an abbreviation for ontological category). Essentially, this structure summarizes any information that is important for determining the actual referent of an expression in the current context. The double-headed arrows between the Referents indicate that their values are shared (with values that

originate non-locally, i.e. through such a binding, shown in italics). The dashed border of the two Referent structures contributed by the lexical constructions indicates a slightly different relationship than that between the DETERMINEDNP construction and its Referent; we return to this point below.

As should be apparent, even a noun phrase as simple as *der Fußballspieler* involves many representational choices, with reasonable alternatives varying in both the complexity of the structures defined and the generality of the phenomena they account for. Our goal here is not to argue for the particular analysis adopted here as the best or most general one possible; rather, we focus on the basic representational toolkit involved for expressing a variety of concepts and relations and compare those available in the ECG and FCG formalisms.

3.2.2 Formalizing Constructions

Constructions both in ECG and in FCG are – following traditional construction grammar approaches – (mostly) composed of a form and the meaning pole. We say "mostly", as in FCG it is possible to have purely syntactic or purely semantic constructions. To explain those specific constructions is, however, beyond the scope of this work and we refer to the FCG literature for an explanation. The form pole in both notational frameworks can be constituted by any linguistic symbol, be it a morpheme, a lexeme or a whole phrase and the meaning pole can be assigned to the construction's function.

3.2.3 Lexical Constructions

The class of lexical constructions contains all linguistic units that have a separate entry in a dictionary, i.e. so-called *headwords*. It also includes idioms as, for instance, *to kick the bucket* or compounds as the example lexical item *Fußballspieler*. Both example constructions for *Fußballspieler* are shown in Figure 3.3. On the left hand side, the construction is shown in FCG and on the right hand side in ECG notation, respectively.

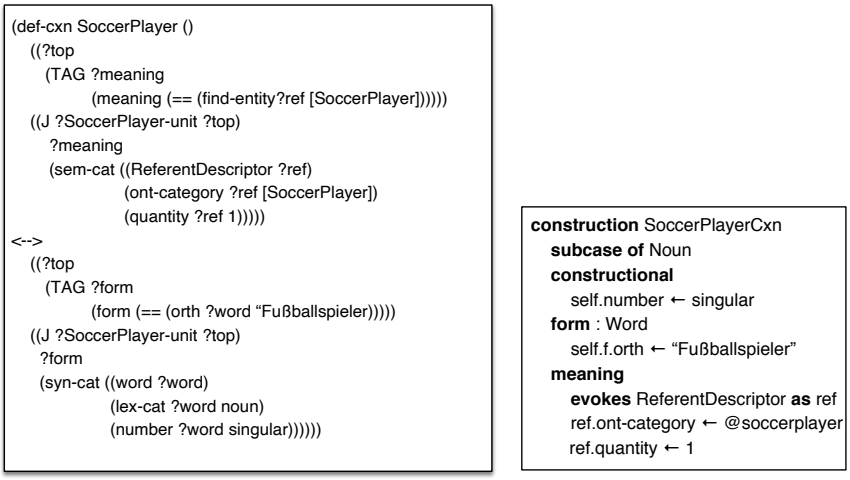


Figure 3.3: The lexical construction for the German term *Fußballspieler* (*soccer player* in English) both in FCG (left) and ECG (right) notation.

Both structures capture a relatively straightforward pairing of form (the orthographic string "Fußballspieler") and meaning (the soccer player ontological category associated with a referent, whose quantity is additionally specified as 1). They also include grammatical information (i.e., that *Fußballspieler* is a singular noun), though they differ in precisely how this information is expressed.

A few differences in basic format are apparent even at first glance. Roughly speaking, the format of FCG reflects the influence of the LISP programming language: internal structure is indicated with parenthesized lists employing prefix-list notation, and variable names are marked with a leading question mark. In contrast, the format of ECG reflects the influence of constraint programming languages and inheritance-based ontologies: special keywords (in boldface) are used to indicate internal structure and express inheritance relations and other constraints, and dotted slot chains are used to refer to non-local structures. In a nutshell, on the surface, FCG looks more like a program, and ECG more closely resembles a data structure.

The sections below take a closer look at the two constructions in 3.3. To ease comparison, we focus on how each construction captures the informal

linguistic analysis described in the previous section.

Nominal Constructions in FCG: The FCG structure in Figure 3.3 makes use of the basic FCG def-cxn predicate for defining constructions.[5] which consists of two main sections (or poles) separated by a double-headed arrow (<–>), corresponding to the meaning (or semantic) and form (or syntactic) domains. Each of these poles includes two units, one named ?top and one named ?SoccerPlayer-unit (where the leading question mark indicates that these are variable names); this latter unit is a J-unit (as indicated by the operator J). Regular units (i.e., non-J-units, in this case labeled ?top) and J-units together specify the constraints and categorizations relevant to each domain, though they differ in the kinds of information they typically contain. Broadly speaking, constraints in regular units of lexical constructions tend to be based on perceptual or cognitive categorizations, while those in the J-units correspond instead to specifically linguistic features and categorizations, typically expressed using semantic and syntactic categories (specified in sem-cat and syn-cat lists, respectively).[6]

Concretely, then, the definition organizes the various elements of the informal analysis in terms of which domain they belong with (meaning or form) and whether they are specifically linguistic. In the meaning domain, the variable ?ref (corresponding to the Referent structure connected to the word *Fußballspieler* in the informal analysis) plays a central role: the ?top-unit specifies that ?ref is of ontological category [SoccerPlayer] (where square brackets denote reference to an ontology item). In the J-unit, the predicates in the sem-cat list specify additional constraints on ?ref, namely that it can also be categorized as a ReferentDescriptor and

[5] Please note that the notation described in this comparison uses a dated version of FCG notation. Current grammars are written in an adapted syntax. For more information we refer to Van Eecke, 2018 and the Babel wiki: *https://emergent-languages.org/wiki/docs/recipes/fcg/syntax-and-semantics/*.

[6] Regular units and J-units also behave differently during language processing. We refer the interested reader to Section 6.1 in Chang et al., 2012 where the special role of the J- operator and other notations (such as the TAG operator) during language processing is discussed in more detail.

that its quantity is 1. As in the ECG description above, the Referent-Descriptor is intended to bundle all constraints on a referent imposed by various constructions, corresponding to the information shown in the Referent structures in the informal analysis.[7]
In the form domain below <–>, the analogous central variable is ?word; the ?top-unit includes the constraint (orth ?word "Fußballspieler"), which specifies the orthographic form of ?word, while the constraints listed in the J-unit's syn-cat feature express its additional syntactic categorizations: as a word whose lexical category is noun and whose number value is singular.

Nominal Constructions in ECG: We now turn to the right side of 3.3, which shows a simple ECG construction for *Fußballspieler*. The high-level structure of the construction includes three blocks, where keywords (shown in boldface) indicate special terms and structures. The constructional block contains information relevant to the construction as a whole, while the form and meaning blocks (or `poles`) contain information relevant to each of those domains. These poles are themselves structured, and can be referenced and constrained within the construction. The `self` keyword allows self-reference (i.e., reference to the construction being defined), while the special names `self.f` and `self.m` refer to the construction's form and meaning poles, respectively.
The SoccerPlayerCxn construction is defined as a `subcase`, that is a more specific instance, of the Noun construction. In fact, ECG constructions are all defined in a multiple inheritance lattice, and a separate lattice defines represent schematic form and meaning categories, or SCHEMAS. Accessible structures can be constrained to instantiate particular schema categories.
Each block contains various constraints that apply to the relevant domain, drawn from a fixed set of possibilities. We highlight a few of the notations that express these constraints:

- The constraint that the form of any construction should be of a cer-

[7] The term's name is chosen to avoid confusion with the actual referent, i.e., a concrete entity in context that satisfies those constraints.

tain type is expressed with the help of the : operator. This operator enables the assignment of a special type of schema or construction to a certain role. For instance, the schema WORD is assigned to the form pole of all lexical constructions. The only role the schema WORD lists is the `orth` role. The orthographic form of the lexeme, idiom or compound in quotation marks is assigned to it with the help of the assignment operator ←. This operator assigns atomic values to specific roles. In this example, the string "Fußballspieler" is assigned to the ORTH role of the SOCCERPLAYERCXN.

- Category constraints are indicated with a colon (i.e., the form pole must be a WORD), and role-filler constraints of the form `x` ← `y` indicate that role (or feature) `x` is filled by the (atomic) value `y` (i.e, the form pole is associated with the `orth`ographic string shown).

- The `evokes ReferentDescriptor as ref` declaration indicates that there is an instance of category `ReferentDescriptor` present, accessible using its local name `ref`. Subsequent constraints specify that its feature `ont-category` be filled by `@soccerPlayer` and its `quantity` set to 1. (Like the square brackets in the FCG definition, the `@` symbol indicates reference to an external conceptual ontology.)

In short, the construction indicates that the SOCCERPLAYERCXN is a kind of NOUN; asserts constraints on the constructional (or grammatical) number feature and the particular orthographic form; and evokes a `ReferentDescriptor` of a specified ontological category and quantity.

Determiners in ECG and FCG: Basic lexical constructions for determiners constrain a referent, just like the earlier presented SOCCERPLAYERCXN construction, but instead of specifying its ontological category, they typically constrain features like number, gender and proximity, and, as they do in German, case. As a detailed description of the modeling of determiners both in ECG and FCG would go beyond the scope of this work, we refer to related literature: For an FCG approach to determination in

German see [van Trijp, 2012], for an example how ECG could possibly deal with determination in German please see [Micelli, 2004].

3.2.4 A First Comparison

Given the shared theoretical and methodological commitments mentioned in the previous paragraph, is not entirely surprising, that the two formalisms have much in common: each represents the basic forms and meanings involved, as well as additional grammatical information associated with reference, as expressed by common nouns and determiners. But these examples also exhibit some striking differences. Perhaps the most important distinction between the formalisms is the treatment of categories and inheritance. ECG structures are all defined within inheritance lattices specifying constructional and other relationships. Many ECG notations thus allow reference to other existing structures, for example to inherit features and values, or to assert values or bindings on connected structures. FCG constructions, on the other hand, rely on category lists associated with each domains; each construction is thus relatively standalone, and defined independently of other structures that may contain similar information. In the sections below, we discuss several representational consequences of this fundamental difference.

Inheritance and Categorization

Categories play a prominent role in most linguistic theories: they capture generalizations about shared structure and behavior across different linguistic units. Part of speech categories, semantic (or thematic) roles, lexical subcategorization types, speech act types, and phonological categories are all well-established ways of carving up various linguistic domains into groups exhibiting similar properties. Both ECG and FCG allow such categories to be expressed, but they differ in the approaches taken, as well as the degree to which the relationships among categories is made explicit. The ECG approach to categories is based on inheritance networks, where shared properties are expressed at the highest level of generalization pos-

sible and inherited by subsidiary categories and instances. That is, ECG constructions are defined (using the `subcase of` relation) within a multiple inheritance hierarchy. Structures and constraints defined in any *base* (or *parent*) constructions are inherited by and accessible to their subcases, and thus need not be explicitly specified. The subcase can impose additional constraints, or refine existing constraints.
Categories are a fundamental notion in FCG, expressed as predicates in the `sem-cat` and `syn-cat` lists. Constructions that have such predicates in common implicitly form a category. Inheritance networks have not yet been much explored in FCG: there is no explicit notion of inheritance for constructions, meaning or form components, or semantic and syntactic categories. A few developments, however – such as the use of templates [Steels, 2011] and distinctive feature matrices [van Trijp, 2011] – can be seen as moving in this direction. Templates, for example, provide a means of capturing similarities across constructions, allowing a more concise, uniform declaration of constructions. Note, however, that templates currently serve mainly as an abbreviation: they do not specify inheritance relationships. That is, there is no mechanism for allowing one construction to refer to or inherit from another, and more general lexical categories like Noun are not themselves defined as structures that can inherit features. Of course, the use of templates in FCG is relatively recent and their precise form is still under development. Thus it may be possible to extend the template approach to exploit the benefits of inheritance and type hierarchies. These benefits become especially important as grammars grow larger: keeping track of the various dependencies between constructions for any non-trivial language phenomenon is a tedious undertaking. Adopting approaches based on inheritance would enable more concise grammars that reduce errors.

Form and Meaning Representations

The lexical examples we have seen also illustrate different approaches to representing the domains of form and meaning. Organizationally, FCG

distinguishes specifically linguistic categorizations (as listed in `sem-cat` and `syn-cat`) from the concrete forms and meanings taken to be based on perceptual or cognitive categorizations (associated with the `meaning` and `form` parameters. ECG constructions do not explicitly represent this difference in the notation itself (except for the use of `@` to denote ontological categories). A more important difference lies in how these categories are represented. As noted before, all categorizations (linguistically specific or not) in FCG are expressed in predicate-argument format, and are independently defined as part of each relevant construction. The particular style of semantic representation can vary; though the examples shown in this section have a declarative flavor (following the informal analysis presented before), other studies show how it is also possible to adopt procedural semantics [Spranger and Loetzsch, 2011] or frame semantics.

ECG, by contrast, was designed specifically to support frame-based semantic representations; it thus includes as part of the formalism notational tools for defining schematic structures, called *embodied schemas* or just *schemas*. Those schemas represent the needed dynamic and inferential semantics in ECG. They are schematic, role-based conceptual and semantic structures and describe an embodied, parameter-based representation of language. Special semantic expressiveness is achieved with the help of various semantic operators (as for instance the **evokes** operator). The EVOKES operator has parallels in Frame Semantics [Fillmore, 1982] and Cognitive Grammar [Langacker, 1987]. It presents one of the main differences between ECG and other unification-based grammars. This operator allows that some roles which are essential for the understanding process can be imported, which means they are made accessible for that specific construction or schema. Langacker uses the notions of *profile* and *base* to explain this notion, whereby *base* describes the background which exactly contains or *evokes* the cognitive domains that are necessary in order to completely understand the meaning of a word, phrase or sentence. The *profile* can only exist in relation to the *base* [Langacker, 1987, p.183ff.]. Langacker's hypotenuse example can elucidate the relation between the two notions and is depicted in Figure 3.4:

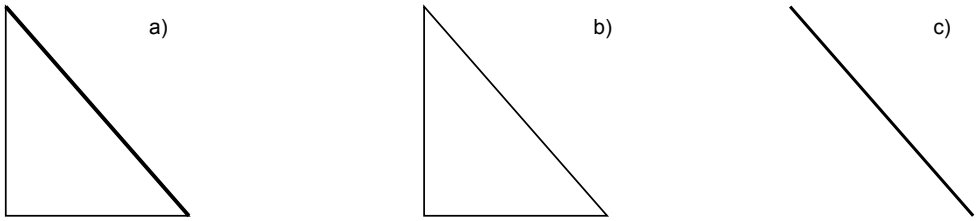


Figure 3.4: Langacker's hypotenuse example (see [Langacker, 1987, p.183ff.]).

A hypotenuse only exists while having the picture of a complete right-angled triangle in mind. In Figure 3.4 a) you see the *base* connected with the *profile*. Figure 3.4 b) shows the isolated *base*. If you do not consider the hypotenuse within the domain of a right triangle, it loses its meaning and displays an ordinary straight line. The display of the isolated hypotenuse can be seen in Figure 3.4 c).

They are again defined within an inheritance lattice. ECG schemas resemble depictions of semantic frames and image schemas in the literature, and are similarly used to bring together a set of associated and interdefined roles or features comprising a complex concept. The roles defined in a schema can be referred to and constrained by other schemas and constructions. Hence, both lexical constructions assert form constraints on the `orth` role of the WORD SCHEMA, as well as meaning constraints on the roles of the `ReferentDescriptor`.

As with constructions, we see that ECG emphasizes the interdefined nature of constructions and their associated forms and meanings. Separately defined schemas capture various linguistic generalizations and expectations, allowing brevity in definitions and enforcing some consistency across constructions. FCG constructions, meanwhile, each independently define their relevant predicates, which are therefore less constrained. This tradeoff – between explicit representation of generalizations on the one hand, and freedom of expression on the other – will manifest itself in several other ways to be discussed.

Constructional Features and Grammatical Categories

The two formalisms differ, finally, in how certain kinds of grammatical information are treated. Specifically, while all categories and constraints in FCG must be in either the meaning or form pole, ECG allows some information to be expressed in the constructional lattice:

- Constructional inheritance: Lexical and grammatical categories (like noun and verb) can themselves be represented as constructions and associated with specific roles and values. Thus, the SoccerPlayerCxn construction can be defined as a subcase of the Noun construction, inheriting relevant properties it may share with other nouns.

- Constructional features: The constructional domain itself can be defined as having particular features, often inherited from ancestral types. It is, for instance, possible, to define schemas in the constructional domain that list various grammatical features like for instance agreement features. These are not strictly about either the form or meaning domain; rather, they are associated with the constructional connection between the two.

In both cases above, the equivalent information can be expressed in FCG but must be explicitly included in every constructional definition. In each formalism, it remains largely at the discretion of the grammar writer how to decide precisely which features ought to be defined and what domain they belong in.

3.2.5 Compositional Constructions

Compositional constructions are constructions which are on a higher level of abstraction than lexical ones. This means, that constructions exist which combine different, smaller constructions into one bigger unit, i.e. into a compositional construction. Hereby, those grammatical constructions exhibit hierarchical constituent structure, which is one of their defining feature. Like lexical constructions, such constructions can impose

constraints in both the form and meaning domains, such as word order (form) or role-filler bindings (meaning). They may also enforce compatibilities, or agreement, across constituents. Both ECG and FCG have ways of introducing constituents, specifying relational constraints and enforcing agreement. The DETNOUN construction will serve as an example. It combines a determiner and a noun into a larger phrasal unit.

Determined NPs in ECG The DETNOUN construction in Figure 3.5 shows an example of how a determined NP with constituent structure might be defined in ECG. The intuition behind the analysis is that such phrases draw on both determiners and nouns to provide crucial information for constraining an act of reference, resulting in a single larger unit (as in the informal analysis from before).

Again, the **subcase of** operator is used to mark inheritance. A construction can inherit form and meaning information from other constructions. That means, that roles of the higher constructions they inherit from are accessible and can be modified locally. The DETNOUN construction combines a determiner construction and a noun construction into one unit, i.e. into a so-called *referring expression*. That means that the resulting unit is a SUBCASE OF the REFEXPR construction– which is the class of constructions describing referring expressions. Referring expressions include for instance pronouns, entities or proper names, i.e. every kind of construction that could be used for making reference to some object.

Phrasal constructions, as well as lexical ones, consist of a form and a meaning pole. In addition, they have a so-called *constructional block*, in which several constructional constituents, here one constituent of type determiner and one of type a common noun, and additional constraints are specified. The determiner and the noun are given local names, here d and c respectively, that allow simple access to their respective form and meaning poles within this construction.

In the constructional block, agreement between the determiner and the respective noun can be checked. This is done with the help of the $\leftrightarrow$ operator - the so-called *identification* operator (see Bergen and Chang, 2005).

This operator constrains that the fillers of the two roles - here, for instance, the determiner's grammatical gender (d.gender) and the grammatical gender of the noun (c.gender) - are identical. In this example construction, agreement is between gender and number of both the DETERMINER and the COMMONNOUN construction. The DETNOUN would not fire if gender and number of the determiner and the noun would not match, i.e. if the roles GENDER and NUMBER of both constructions were not filled with identical values.[8]

As previously mentioned, the **self** keyword allows self-reference of the construction itself. Therefore, what happened here is that the case of the unit built by this construction is identical to the case of the determiner that is included in the larger unit. In the example grammar built for this work, we decided to encode case in determiner constructions and not in noun constructions. Similarly, the value of the number feature of the noun construction is shared by the resulting referring expression.

Turning to the form domain of the construction, the form poles of the two constituents are specified as coming in a particular order: the form of the determiner has to come *before* the form of the noun. The two components of this construction do not immediately have to follow each other. Instead, modifiers like for instance one or several adjectives are allowed in between them.[9]

In the meaning pole of the DETNOUN construction it is determined, that the meaning of the common noun (referred to as c.m in Figure 3.5), used in this respective construction, is assigned to the meaning of the resulting compositional construction (see `self.m` ↔ `c.m`).

The needed dynamic and inferential semantics in ECG - what fills the so-

[8] In spoken language, and especially in dialects, sometimes the gender of nouns diverges from the standard (e.g. German: *die Butter* (engl. *the butter*, gender: female), Swabian *der Butter* (gender: male). This fuzziness is not accounted for in the grammar description. However, this could be learnt and either be stored in the grammar itself (e.g. that *Butter* can either be female or male) or the parser could allow both combinations.

[9] This understanding of *before* corresponds to Allen's definition of his interval relations [Allen, 1983] and states that the determiner construction has to be in front of the noun construction. The `meets` operator, however, forces constituents to follow each other immediately, i.e. no modifiers are allowed in between the two constituents.

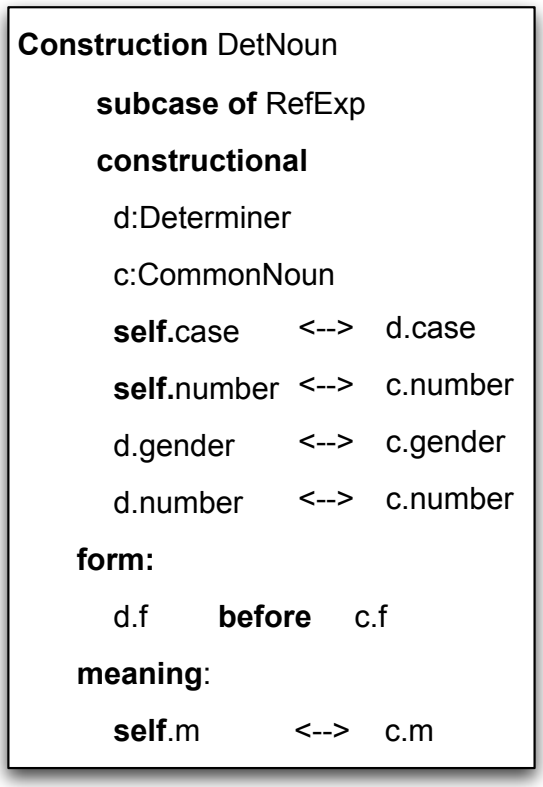


Figure 3.5: The DetNoun construction: A compositional construction for the resulting referring expression combined of a determiner and a common noun in ECG notation.

called meaning pole - is represented by embodied schemas. More detailed information on the different schemas and their usage in ECG is given in a later paragraph. Before we turn to an example how determined NPs might be realized in FCG.

Determined NPs in FCG Grammatical constructions express constraints on aspects of syntactic and semantic structure that link their meanings into a larger whole. Figure 3.6 shows how a determined NP construction for German might be realized in FCG notation. Similarly to the informal analysis described above and to the DetNPCxn in ECG, it combines smaller units (a determiner and a noun) into one single larger unit. It is

a coupled feature structure that consists of the two poles: the meaning (part above the double-headed arrow of the construction) and the form pole (part below the double-headed arrow of the construction).[10]
In the meaning pole, the two units that will be combined are semantically constrained. Different feature-value pairs constrain, for instance, that both the referent of the determiner (called `?determiner-unit`) and the referent of the noun (called `?entity-unit`) will be identical: Note that the construction uses the same logic variable (`?e`) for the referent the determiner is referring to and the referent the noun is referring to, to ensure that they are the same.
The new unit being built after the construction has found appropriate units to combine into a determined NP will as well refer to that referent and will get further semantic categories:

```
(determined +)      ;; it will be determined
(definite +)        ;; it will be a definite NP
(indefinite -)      ;; it won't be an indefinite NP
referent            ;; it is a referent, that other higher-level
                       constructions can refer to
```

Turning to the form pole of the construction, similarly, both the determiner and the noun are constrained, however, in their grammatical features like gender, case or number (to name just a few). Additionally, word order is constrained with the help of the MEETS operator (or keyword), analogous to how word order is handled in ECG.

Comparing Grammatical Constructions The two approaches to representing complex constructions demonstrated in the preceding sections are both capable of expressing constituency, word order and agreement. They differ, however, in several key respects.

[10] As ECG is only used in parsing, FCG examples that talk about application of FCG constructions will focus as well only on parsing. However, please note that all constructions that are shown here, were tested for both parsing and production and work perfectly well in both directions.

```
(def-phrasal determined-noun-phrase ()
      ((?top (sem-subunits (== ?determiner-unit ?entity-unit))
           (footprint (==0 determined-noun-phrase-entry)))
       (?determiner-unit
        (referent ?e)
        (sem-cat (==1 (definite ?def)
                      (indefinite ?indef))))
       (?entity-unit
        (referent ?e)
        (sem-cat (==1 entity)))
       ((J ?np-unit ?top (?determiner-unit ?entity-unit))
        (referent ?e)
        (sem-cat (==1 (determined +)
                      (definite ?def)
                      (indefinite ?indef)
                      referent)) ;; referent-feature is added to meaning of entity unit
        (footprint (== determined-noun-phrase-entry))))
      <-->
      ((?top (syn-subunits (== ?determiner-unit ?entity-unit))
           (TAG ?form
               (form (== (meets ?determiner-unit ?entity-unit))))
           (footprint (==0 determined-noun-phrase-entry)))
       (?determiner-unit
        (syn-cat (==1 (pos determiner)
                      (number (== ?number)) ;; aggreement checks
                      (gender ?gender)
                      (case ?case)
                      (definite ?def)
                      (indefinite ?indef))))
       (?entity-unit
        (syn-cat (==1 (number (== ?number)) ;; aggreement checks
                      (gender ?gender)
                      (case ?case)
                      (function nominal))))
       ((J ?np-unit ?top (?determiner-unit ?entity-unit))
        ?form
        (syn-cat (==1 (pos (== ref-expression))
                      (case ?case)
                      (number (== ?number))
                      (gender ?gender)
                      (determined +)
                      (definite ?def)
                      (indefinite ?indef)))
        (footprint (== determined-noun-phrase-entry)))))
```

Figure 3.6: The DetNoun construction: A compositional construction for the resulting referring expression combined of a determiner and a common noun in FCG notation.

First, as elsewhere, the ECG formalism avails itself of type lattices for both constructions and schemas. Thus, various constraints require that relevant features are defined and accessible for a given structure (i.e., a

slot chain like `det.number` implies that `det` is defined as having inherited a `number` role). This stands in sharp contrast with FCG, which does not require typing of this kind: previously unspecified features are added during processing if not already defined, and only if it is explicitly indicated that this should be the case. The less type-constrained approach of FCG may be seen as a double-edged sword: while it leaves more freedom of choice, it also requires that the grammar writer maintain the soundness of his or her grammars and ensure that the relevant semantic and syntactic categories of constituent units are percolated properly to newly created units. For instance, although the `?DeterminedNP` in the DeterminedNP construction unit is specified as an instance of the ReferentDescriptor schema, neither its `givenness` or `ont-category` values are specified. These could be inferred from its constituents, and the template could perhaps be changed to do this automatically, but again, doing so would be far from trivial. In contrast, ECG makes some structural assumptions that allow certain constraints to be succinctly stated, though possibly at the cost of flexibility. Thus the identification of the various `ReferentDescriptors` allows all their roles to be bound with one constraint, both across constituents and with the overall resulting construction. Second, the two formalisms allow somewhat different options with respect to how particular kinds of features are expressed. As noted earlier, grammatical features and categories are typically expressed in the constructional domain in ECG (though as demonstrated above, agreement can also be enforced just in the form or meaning domain). As with the lexical constructions, FCG tends to express such grammatical information by including it as a syntactic category. This difference may not ultimately affect expressive power, but it does reflect different theoretical views of particular linguistic concepts.

We refrain in this section from going into detail of how processing is handled in both formalisms and refer the interested reader to read the complete paper by the author et al. [Chang et al., 2012], where a comparison of the two different models is elaborated in further detail.

The above presented comparison shows that it is a highly complex undertaking to compare two different computational models that are on the

one hand two computational models of "the same" grammar model, which are, however, used in two completely different systems, being designed for completely different reasons, by completely different people in again different contexts, thus, being completely different in many of their design choices. This challenge and finding also undermines why an evaluation of a grammar formalism is highly complex and can in our eyes almost never be holistic and comparable.
The following section draws on what we have learnt so far from the comparison of the two existing computational construction grammar frameworks and tries to motivate why it might be of value to investigate other ways of representing construction grammar's core notions.

3.3 Main Motivation for and Merits of a New Formalization

Besides the fact that we believe that statistical natural language processing systems have reached a point where only small further improvements can be made and we advocate symbolic grammar formalisms at least to support those systems, we believe that construction grammar formalisms might aid in achieving better deep natural language processing systems being able to process language at a level that understands meanings, disambiguates constructions depending on a certain context, grasps implied meanings, or makes inferences.
The main motivation for proposing another constructional grammar model and not using one of the existing frameworks presented in the previous section is that at this early stage of research in the field of computational models of construction grammar, we consider it being worth investigating another way of representing construction grammar's main features experimenting with the formalism offering complementary advantages.
Both ECG and FCG are parts of highly impressive systems and especially FCG has recently been improved impressively with regard to applying it to deep natural language processing (e.g. [Verheyen et al., 2023]). How-

ever, the treatment of categories and inheritance made me choose ECG as a basis for engineering a version of construction grammar based on ontologies. ECG structures are all defined within inheritance lattices specifying constructional and other relationships. Many ECG notations thus allow reference to other existing structures, for example, to inherit features and values. FCG constructions, on the other hand, rely on category lists associated with each domains; each construction is thus relatively standalone, and defined independently of other structures that may contain similar information. The closer look at ECG and FCG in the previous section might have already illustrated a few open issues that might be in need of improvement or further investigation. Especially, since the original domains of application of both of the two frameworks are rather specific and have not been aimed at being used in conversational systems - at least initially.

FCG, at present, is developed within an active network of researchers and gains more and more visibility in various communities (see, e.g., [Steels, 2005], [Steels, 2011], [Steels, 2012], [Steels, 2017] or *https://www.fcg-net.org/*). The current FCG implementation shows great promise for being used for deep natural language understanding[11] and studies are continuously being carried out for various natural language phenomena in different languages (see, for instance, [Nevens et al., 2019], [Verheyen et al., 2023], [Willaert et al., 2022]) allowing FCG engineering in an innovative, new environment [van Trijp et al., 2022].

However, several issues mentioned in the section on grammar engineering in general (see Section 2.2.2) are still to be further discussed and investigated in both FCG and ECG as for instance reusability, grammar extension, the addition of richer semantics, evaluation of grammars, comparison among grammars in the same or even another formal notation and the addition of already existing knowledge bases.

As with the link to ontologies, it has been defined in both of them, however, the ontologies used in most experiments are far from being as elaborate as ontologies nowadays can possibly be. Working with ontologies in a

[11] See recent advances with FCG on visual question answering [Nevens et al., 2019].

natural language understanding system and on building digital assistants for the enterprise including past and current research studies dealing with building ontologies to be used in NLP further pushed the idea of including a constructional grammar layer in such an ontology to find out if that is actually possible and eventually beneficial for NLP systems.[12] Additionally, the network-like structure of both constructions and schemas – also mirrored in ECG's lattice structure – supports the decision to formalizing construction grammar within a formal ontological model.
The main goal of this undertaking is not to replace any of the existing formalisms, but rather to give new impulses to further improvement and maybe show another direction that might better future computational construction grammar research and give the grammar theory further visibility in further communities.

Formal ontologies represent the state of the art in knowledge representation. Section 2.3 introduced formal ontologies and their formats, defined the term, listed the main steps to be taken in ontology engineering, introduced the main foundational ontologies and discussed several different possibilities of how to represent linguistic knowledge in ontologies.
Ontologies offer various new and semantically rich possibilities how constructions and schemas can be related to each other: The lattice that can be built both among constructions and among schemas within an ontology can be much more semantically fine-grained, as relations in an ontology go beyond simple inheritance relations, which basically come for free in an ontology. Existing editors, as for example protégé[13] facilitate accessibility, readability and extensions of ontologies in various ontological formats.[14]
The format which results from building the ontological grammar model is one of its major advantages. As previously discussed, grammar engineering is often done in a non-efficient way, resulting in models that are difficult to reuse and difficult to adapt to new application areas or even

[12] We refer the reader to literature on that topic as, e.g., Wahlster, 2007, Alatrish et al., 2014, Kersloot et al., 2020.
[13] http://protege.stanford.edu/
[14] See Section 2.3.2 for a short description of different ontology formats.

impossible to be used in different processing models. Since ontologies are used in a variety of applications and especially since natural language processing with ontologies is getting more and more popular, we believe that our model can be of benefit in various existing applications that can already handle ontological formats.

The ontological framework described in detail in the next chapter combines two ontological modeling frameworks endowed with a construction grammar layer. Next to the support via dedicated editors and inference engines, one of the central advantages of the ensuing ontological model lies in its compatibility with other ground ontologies. To enable this compatibility, the choice of the foundational ontology has to be considered carefully. We decided to use the *Descriptive Ontology for Linguistic and Cognitive Engineering* (abbrev: DOLCE) [Masolo et al., 2003, Gangemi and Mika, 2003] including two additional ontological frameworks: *Descriptions & Situations* (abbrev.: DnS) [Gangemi and Mika, 2003] and the *Ontology of Information Objects* (abbrev.: OIO) [Guarino, 2006], which are both extensions of the foundational ontology. The following section presents a detailed description of all mentioned ontological frameworks and motivates the reason why they have been selected.

3.4 Which Foundational Ontology is Used in this Work and Why?

The foundational ontology that is used as centerpiece of the ontological framework is the *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE) [Masolo et al., 2003, Gangemi and Mika, 2003].[15] Main reason for this decision has been the fact that DOLCE is based on first order logic and that it is cognitively motivated. That means that it was designed based on the underlying ontological concepts of natural

[15] See also Section 2.3.4 on a discussion of DOLCE's ontological choices and a brief description.

language and human commonsense. Therefore, it is a so-called *descriptive* ontology.[16] Descriptiveness is considered important for our undertaking since the concepts and relations that are being modeled are artifacts of human common sense.
We decided to integrate construction grammar's main building blocks into DOLCE's extensional module *Descriptions and Situations* (DnS), which again contains an extensional ontological module which is the *Ontology of Information Objects* (OIO). Therefore, the structure of both modules will be described in more detail in the following section.

Descriptions and Situations The *Descriptions and Situations* ontology is an extension of the DOLCE ontology [Gangemi and Mika, 2003, Gangemi et al., 2004]. It is an ontology which is plugged into DOLCE and which offers a way to talk about non-physical entities as for instance plans, mental contents, or roles, and – most important here – language or linguistic items. This fact made us consider it the perfect fit to represent constructions embedded within this ontological framework. How exactly the features of construction grammar are engineered in tune with the used ontological framework will be described in Chapter 4.
Descriptions: According to the definition from the DnS ontology, the class of `Descriptions` represents mental objects, states, or conceptualizations. That means that we can consider `Descriptions` are the ontological equivalents of meaning.
Situations: A `Situation`, on the other hand, is a non-agentive social object representing a state of affairs, a relationship or e.g. a fact. It only exists in case it `satisfies` a `Description` and in case it is the `Setting` for at least one entity from the ground ontology [Gangemi et al., 2004].

For integrating the DnS ontology into the foundational ontology DOLCE, the class `Situation` is introduced as a (new) top category. Additionally, the class `Description` is inserted as a subclass of the non-physical class `Endurant`. Some examples of `Descriptions` and of `Situations` are

[16] For a more detailed discussion see [Masolo et al., 2003].

listed in Table 3.1 below (see Gangemi et al., 2004 for more details and for more examples of `Descriptions` and `Situations`).

Description	**Situation**
Theory	Model
Plan	Plan execution
Play	Performance
Rules of game	Play a game

Table 3.1: Examples of classes of `Descriptions` and corresponding classes of `Situations` Gangemi et al., 2004.

Ontology of Information Objects (OIO) `Information Objects` provide the formal representation of `Descriptions`. This is ensured with the help of the `expresses` property. In other words, the `expresses` property describes the relation between signs (i.e. `Information objects`) and their meaning, which is ontologically reified in `Descriptions`. That means that, for instance, iconic or linguistic objects are `Information objects`. The class of `Information objects` constitutes a class of the D&S ontology, and more precisely, a subclass of the class `Social object`.
`Information objects` are `ordered by` a `Code`, which can be any system of information encoding as, for instance, the German language. They can be `about` any entity. Unlike the property `expresses`, `about` requires a situation to be about something. `Information objects` are `realized by entities`, to be more concrete, for example, by strings.
The basic design pattern of the *Ontology of Information Objects* is displayed in Figure 3.7.
In the center you find the class `InformationObject`. The relations the class has to other ontology classes which were previously mentioned are displayed by directed arrows which are labeled with the name of the respective property and which point to the classes `InformationObject` is related to (the property's so-called *range*).
For our work, the two properties of major importance are the `expresses` and the `realizes` property and their two respective inverse properties:

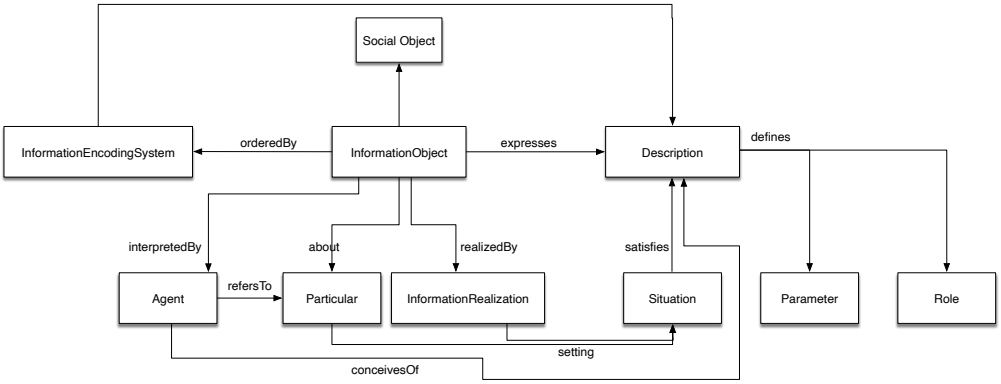


Figure 3.7: Structure of the *Descriptions & Situations* Ontology and the *Ontology of Information Objects*.

`expresses-by` and `realized-by`. How we make use of them will be described in the sections on the concrete modeling of constructions and schemas in the following Chapter.

Figure 3.8 provides a concrete example of an instance in the D&S and OIO framework. It shows the ontological representation of the former German soccer player *Michael Ballack*:

There exists a `social object`, which is `Michael Ballack` in this concrete case. This `Information Object` is `ordered by` or expressed according to the German language, which is an instance of the class `Information Encoding System`. It `expresses` the concept or the meaning (which is the `Description`) of Michael Ballack. It is `realized by` the actual string `Michael Ballack`. This `Information Object` is about a `Particular`, which is Ballack in a specific context. The `Situation` that Ballack has played in the German national team is the `setting` for the `Particular` and at the same time `satisfies` the `Description` which again is the conceptualization of Michael Ballack.

There are several reasons why the combination of ontological modules just presented provides the ideal setting for integrating a construction grammar

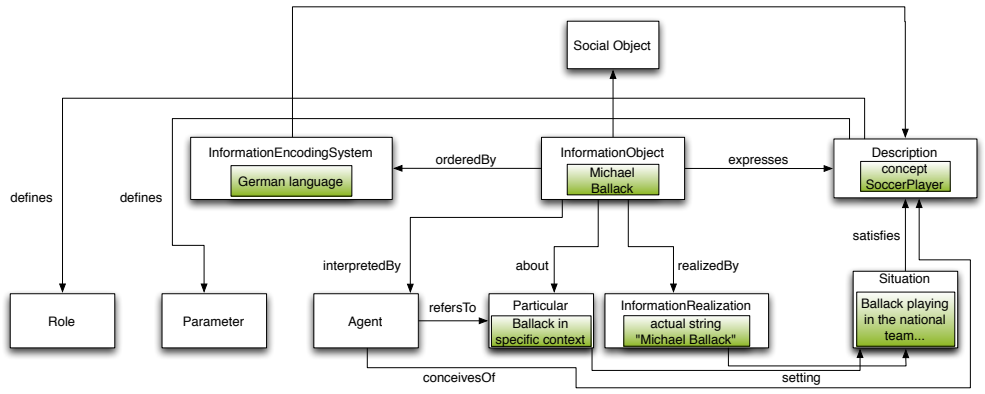


Figure 3.8: Concrete example representation of a former German soccer player called Michael Ballack.

layer. The following summarizes the main reasons:

- DOLCE is cognitively motivated. Since construction grammar is a grammar formalism which is heavily based on cognitive grammar, this feature has to be given.

- `Descriptions` are the ontological representations of meaning. They will serve as the meaning pole of constructions, capturing meaning representations as image schemas or frames. Additionally, the link to other knowledge resources can be established here. The following section goes into further details on what exactly will be included in the class of Descriptions.

- `Information objects` realize `Descriptions`. According to the definition of the class of `information objects`, linguistic items, as, for instance, constructions, can be subsumed under this superclass. Therefore, both the form side of constructions – that is their orthographic form – and the constructions as a whole unit fit perfectly in this ontology class.

The following section introduces the LingInfo Model, a model designed to add linguistic information as parts of speech or grammatical gender to ontology classes and their instances.

3.5 The LingInfo Model

This section introduces the ontological LingInfo model allowing to add rich, linguistic information to ontology classes and properties.[17] We created this model since no existing approach displaying linguistic information in ontologies fulfilled the needs of the project back then (see Section 2.3.5 for a short presentation of those approaches) and allows now even further elaboration when building a construction grammar based on ontologies. We do need exactly that: An ontological model that can include rich and fine-grained linguistic features (and that is easily extensible) keeping syntactic and semantic, ontology-based knowledge representations separate. Available resources like Wordnets, for instance, lack the linguistic features that are needed (see e.g. Bateman et al., 1995, Alexa et al., 2002) and SKOS[18] mixes linguistic and semantic knowledge. Compared to those and the approaches introduced in Section 2.3.5, the LingInfo model clearly keeps the linguistic and semantic, ontology-based knowledge representations separate: The ontology is represented using the semantic relations defined in RDFS or OWL-Full with linguistic knowledge attached to classes and properties.

The main advantage: LingInfo constitutes an ontological model in RDF/RDFS that can provide ontologies with linguistic information, assuming they are in the same ontological format. It contains linguistic information for different languages, momentarily for English, French, and German, and can easily be extended to other languages. Main objective of this ontology is to provide a mapping between ontological concepts and lexical items, i.e. to provide a grounding to the human linguistic domain. That means, that the possibility is offered to assign linguistic information as, for instance, the orthographic term, its grammatical gender, its part-of-speech, stem etc. to the respective classes and properties of an ontology.

[17] The model has originally been developed within the SmartWeb project [Wahlster, 2007] by the author et al. See [Buitelaar et al., 2006].

[18] *http://www.w3.org/TR/swbp-skos-core-guide/*

Applications which need to have a link between text and concepts of an ontology (or vice versa) are in need of a model like LingInfo or a similar one, assigning that kind of information to its ontology classes and instances.

Figure 3.9 shows the integration of the simplified LingInfo lexicon model into an ontology in need of linguistic information for its concepts. This is done via the definition of a `meta-class` and a `meta-property` (turquoise, dashed boxes top part of Figure 3.9). In the `classes` portion of that figure, you can see an example domain ontology class `o:DomainOntologyClass` (green box). This class inherits the property `lio:PropertyWithLingInfo` from its parent class `lio:ClassWithLingInfo` and, therefore, can be linked by it to its instance `lio:LingInfo` (box in orange in the bottom part of Figure 3.9) comprising the linguistic features like its language-ID (i.e. an ISO-based unique identifier for the language of each term), its orthographic form, and the morpho-syntactic decomposition of the term. The arrows linking the boxes are properties linking classes and instances.

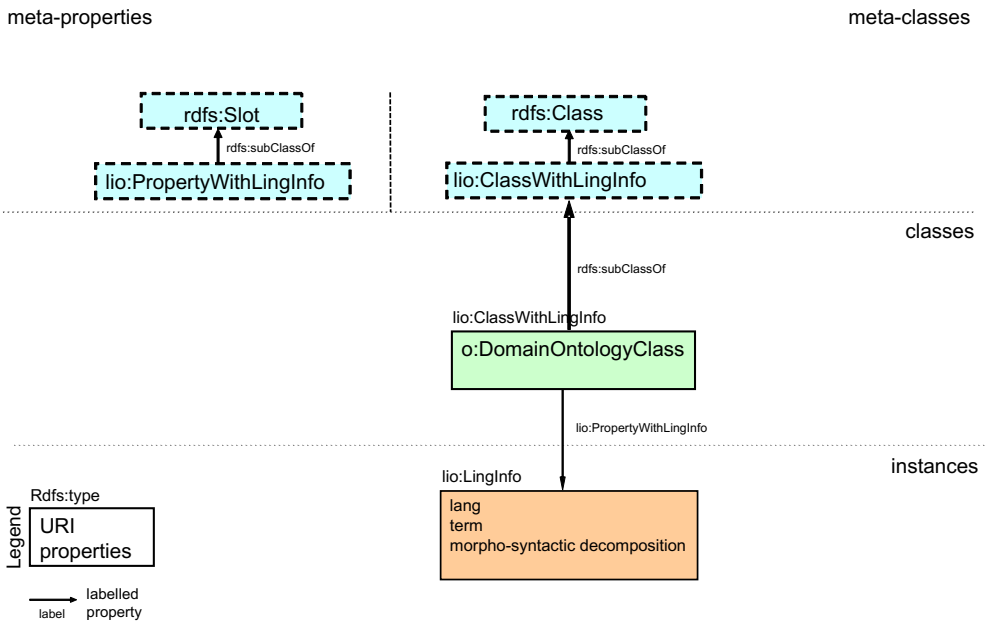


Figure 3.9: A simplified version of the LingInfo model with an example ontology class and associated LingInfo instance linked by properties.

Figure 3.10 shows the same model enriched with a concrete example.

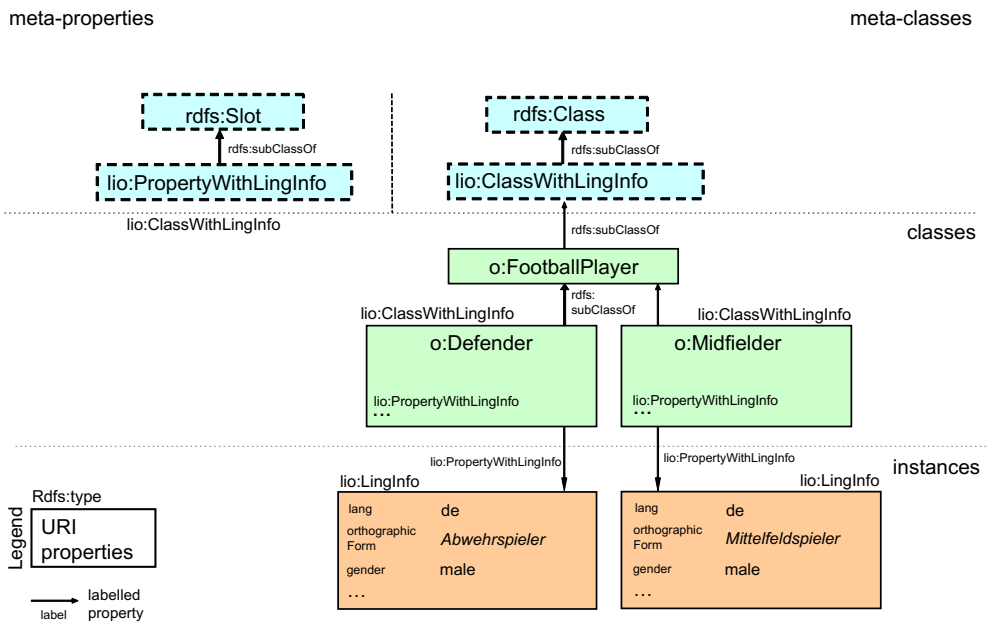


Figure 3.10: A simplified version of the LingInfo model with example ontology classes and associated LingInfo instances.

The ontology classes (example here: `o:FootballPlayer`) is defined as a subclass of the meta-class `lio:ClassWithLingInfo`. All subclasses of `lio:ClassWithLingInfo` can define LingInfo instances (orange boxes at the bottom of Figure 3.10). Those instances represent the linguistic features (`feat:linginfo`) of a term of the respective class. Table 3.2 lists the complete morpho-syntactic information of a term in the LingInfo ontology, a few of them represented in the orange boxes of Figure 3.10.

The domain ontology classes are the classes `o:FootballPlayer` with the respective subclasses `o:Defender` and `o:Midfielder`. Those classes are instances of the meta-class `lio:ClassWithLingInfo`. With the help of the meta-property `lio:PropertyWithLingInfo` the mentioned domain ontology classes are linked to LingInfo instances. Those instances contain then the orthographic form of the term (like e.g. "Abwehrspieler" - the German term for English *defender*), the language ID ("de" for German) and

Slot Name	Domain Class	Allowed Values
case	`WordForm`	Symbol: nom, gen, dat or acc
orthographicForm	`WordForm` and `Morpheme`	String
gender	`WordForm`	Symbol: male, female or neuter
part of speech	`WordForm`	Symbol: noun, verb, adj, det, conj, prep, adv, auxVerb
number	`WordForm`	Symbol: singular or plural
root	`WordForm`	Instances of class `Root`
inflection	`InflectedWordForm`	Instances of class `Affix`

Table 3.2: Morpho-syntactic information as encoded in the LingInfo model.

morphosyntactic information as listed in Table 3.2.[19]
LingInfo is one of the essential puzzle pieces we need to engineer the ontological construction grammar. Let's see which further puzzle pieces are needed.

3.6 Further Picks

This section briefly summarizes further choices that have been made, starting with the one on constructional meaning representation.

3.6.1 Constructional Meaning

Construction grammar regards the meaning of any construction equally important as any other linguistic discipline. The question is now how to formally model meaning in a formal construction grammar. As previously discussed, most of the FCG grammar engineers opted for first order logic to represent constructional meaning. As the grammar formalism seems, however, to be agnostic towards which meaning representation is used,

[19] A diagram of the complete LingInfo model can be found in the appendix.

there are a few studies that go beyond first order logic and use procedural semantics instead (see for instance [Spranger and Loetzsch, 2011]). Then again, there have been attempts to integrate FrameNet frames into FCG constructions (see also [Micelli et al., 2009]).
ECG, however, presents meaning by using so-called embodied schemas. Section 2.4.1 introduced frames and schemas which present the meaning representations that form the basis of ECG's embodied schemas. The previous comparison of FCG and ECG discussed both meaning representations in more detail.
While integrating the construction grammar layer into the foundational ontology the need for more specific information to enable deep language processing has quickly become obvious. We decided to build a hierarchy out of the most important image schemas and to include this hierarchy into the ontological framework. This image schema hierarchy can then be the foundation for further schemas to be connected even deeper than for instance the schema lattice that has been created in ECG grammars. The subsequent paragraph briefly presents the motivation behind building the hierarchy, while Section 4.6 goes into the technical details of engineering it.

An Image Schema Hierarchy As already mentioned in Section 2.4.1, image schemas gained much popularity in cognitive sciences in the 1980s. This popularity is presently still increasing in various domains, leaving different groups work on image schema hierarchies or using a schema representation as meaning representations in their natural language understanding systems.[20]

The image schema hierarchy built in this work presents a dense network of schematic, role-based knowledge. It contains the basic image schemas that we consider as most important and that we believe are necessary for

[20]See for instance [Kuhn, 2005], [Kuhn, 2007] for the use of image schemas in the geospatial domain or [Gromann and Macbeth, 2019] for how to crowd source image schemas to be further used in language processing.

our task. How we came to our decision which schemas to include in the hierarchy and also how the hierarchy looks like in detail will be described later in Section 4.6. In order to avoid misunderstandings please note that it is not claimed that this hierarchy is complete in any way or that it might present the only acceptable version. It rather presents a starting point to be elaborated or modified when need arises.

The following Chapter 4 presents a detailed description of the implementation details of the resulting ontological framework (which has been called ECtoloG), employing all modules having been presented in this chapter. It details all layers of the framework and every step in constructing it while pointing out the main challenges and advantages of the approach.

Chapter 4

ECtoloG - Engineering of a Computational Construction Grammar

This chapter describes the implementation details of a formalization of (Vanilla) Construction Grammar including important ECG-flavored features – hence, the resulting model is called *ECtoloG*. It is described how construction grammar's building blocks are integrated into the earlier introduced ontological modeling framework DOLCE and its extensional modules Descriptions and Situations and Ontology of Information Objects. Additionally, the chapter provides detailed descriptions of the various sources that have been integrated into the ontological framework and of the respective integration processes.

The chapter is structured as follows: The first section deals with setting up the ontological modeling framework and describing how the ontological extensions are integrated. It is followed by an informal constructional analysis of the sentence, showing which constructions might be identified contributing to the sentence's meaning. What follows are several sections on the concrete engineering of the grammar features, starting with constructions – fully instantiated and abstract constructions. The sections

are divided into general modeling descriptions and into parts describing concrete natural language examples. The running example sentence of this and the following chapters is the sentence:

(1) Perrotta liegt am Boden.
Perrotta is lying on the ground.

After a detailed description of the needed constructions – lexical and compositional ones – it is described how additional linguistic information as, for instance, parts-of-speech or morphological information can be modeled in the ECtoloG. Furthermore, a section describing the creation of a basic image schema hierarchy and its integration into the ontology follows. Additionally, it is discussed in the chapter's pre-last section how the model can be enriched with frames and scenarios. The chapter concludes with a discussion of the lessons the engineering effort holds and a summary of the gains of the engineering effort.

4.1 Setting up the Framework

To set up the ontological modeling framework including the foundational ontology and its extensions described in 3.4, the following files have to be loaded into an ontology editor:[1]

- DOLCE-Lite.owl (representing DOLCE (see 2.3.4)
- ExtDns.owl (representing the Descriptions and Situations ontology (DnS) (see Section 3.4))
- Information.owl (representing the Ontology of Information Objects (OIO) (see Section 3.4))[2]

[1] In this work Protégé version 3.3.1 (see *http://protege.stanford.edu/*) is used.
[2] In this work version 397 of the mentioned ontologies is used. Additionally, the lite version of DOLCE is used in this work; see *http://www.loa-cnr.it/DOLCE.html* for detailed comments on the lite version of DOLCE and the other mentioned modules.

This is done by creating a new owl project – which we call ECtoloG – and importing the Information.owl ontology. This ontology in turn imports the DnS ontology, which in turn imports the DOLCE ontology. This way of modeling incrementally ensures that all changes and additions to the ontology are directly saved in the ECtoloG.owl file and that none of the predefined modules are accidentally changed.
The resulting module hierarchy in the ECtoloG is displayed in Figure 4.1.

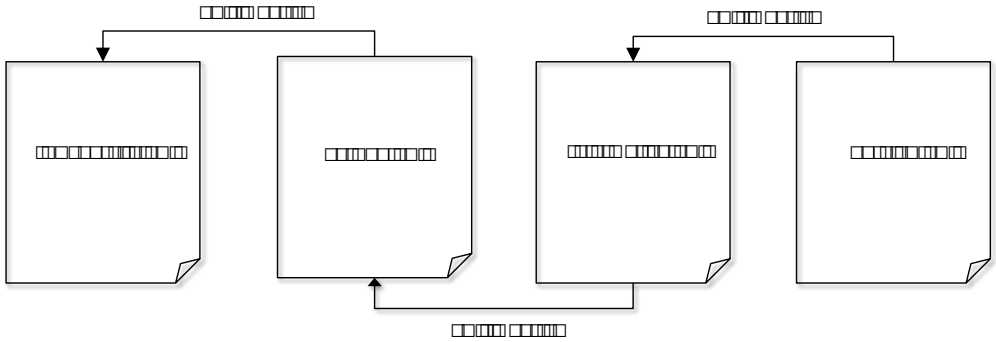


Figure 4.1: The module hierarchy in the ECtoloG.

It employs the ontology-import construct `owl:imports`. This construct is defined in the OWL vocabulary as instance of the OWL built-in class `owl:OntologyProperty`. Each of those instances has to identify the class `owl:Ontology` both as their domain and as their range.[3]

As already discussed in Section 2.3.4, we consider the use of a foundational ontology - in this special case of DOLCE - as highly necessary for our undertaking. The main reasons for this decision are again briefly summarized in the following list:

1. A foundational ontology provides a modeling basis, i.e. a pre-defined set of entities that can be reused.

2. A comparison among other ontological approaches is made possible since the foundational ontology provides a reference point.

[3]See *http://www.w3.org/TR/owl-ref/* for more information on those constructs or more details on any owl related issues.

3. The integration of existing domain ontologies which were built in tune with the hierarchy of the foundational ontology used in the model is made easier.

4. Within a foundational ontology design patterns for re-occurring modeling needs are defined.

After having set up the ontological framework, the actual modeling of constructions can begin. This engineering effort will be described in the following sections. Before, however, the example sentence mentioned above is informally analyzed, to find out which constructions might be needed for its successful constructional analysis.

4.2 Informal Example Constructional Analysis

Let's now have a closer look at the example sentence sentence *Perrotta liegt am Boden* (engl. *Perrotta is lying/lies on the ground*). It will be informally linguistically analyzed below – an analysis necessary for any construction grammatical flavour.
A traditional analysis might identify at first a `name`, a `verb`, a `preposition` and a `noun`. Then the analysis might include several compositional constructions capturing phrases and finally the complete sentence like, for instance, a NoArgClauseConstruction, a NomArgClauseConstruction and a ClausePlusPathConstruction. Part of those constructions' task is to put all of the sentence's components into their appropriate order.
As we are looking at a constructional analysis, it is important to consider the sentence's meaning, as well. The speaker finds himself in a current discourse context in which the hearer has to be able to find out what and who the speaker is referring to, i.e. the referent has to be uniquely identifiable and the act of reference has to be obvious.

A straightforward constructional analysis could identify the following constructions:

- PERROTTA: The word form *Perrotta* constrains the referent to be a kind of entity. Linguistic information like its grammatical case (here: nominative) and gender (here: masculine) might be included in this construction or in a parent construction of that construction which hands down its properties to the construction. Other qualities of the referent like animacy, countability or semantic role might be identified directly here, as well.

- LIEGT: The word form *liegt* constrains the speech act to be of a special kind of self motion. Grammatical properties might be included as for instance its part of speech (here: verb), that it is inflected (here: 3rd person singular) and its grammatical tense (here: simple present).

- AM: The word form *am* is identified as a preposition which constrains its referent to be of a specific grammatical case (here: dative). Its meaning might include a version of the trajector-landmark schema.

- BODEN: The word form *Boden* constrains its referent to be of grammatical categories singular, masculine and nominative, dative or accusative (here: dative). It is a common noun whose referent is constrained to be a kind of uniquely identifiable entity.

- NOARGCLAUSECONSTRUCTION: This clause construction has an inflected verb as its only constituent. Its meaning is the same as the meaning of the verb it captures, i.e. the meaning of the verb percolates to the new unit that is being built by this construction.

- NOMARGCLAUSECONSTRUCTION: This clause construction can have several constituents: A referring expression which is constrained to have the grammatical case nominative and a clause construction of any kind (as for instance a sole inflected verb (here called a

NoArgClause) or a more complex clause like a prepositional clause (here called ClausePlusPath)). It imposes a word order on its constituents (the referring expression precedes the clause). The construction's meaning might be a predication schema.

- CasePrepPathPhraseConstruction: This construction combines a referring expression with a preposition. Compatibility in its grammatical features as e.g. case has to be enforced.

- ClausePlusPathConstruction: This clause construction combines any kind of clause with any kind of path specifier. Its meaning might be a trajector-landmark schema. Compatibility in features as for instance grammatical case and gender must be enforced.

Figure 4.2 graphically depicts the informal analysis of the example sentence. Constructions in the center link the domains of form (left) and meaning (right). Each of the constructions shown here contributes to and constrains the meaning of the whole sentence.
The constructions are displayed in the middle of the figure. Complex constructions are depicted in grey, lexical ones in white. Each construction has a link (shown by horizontal bars) to form (on the left) and meaning (on the right). The form domain contains the relevant word forms, where the dotted arrow indicates the time link (and therefore word order).
The meaning domain contains several structures. Each structure lists features constrained to particular values. Essentially, this structure summarizes any information that is important for determining the meaning of the sentence in the current context. Values between the structures are shared which is ensured by using identical value names.

A sentence as simple as our example sentence shows apparently many representational choices with reasonable alternatives varying in both the complexity of the structures defined and the generality of the phenomena they account for. The goal here is not to argue for the particular analysis adopted here as the most general or even the best one; rather, it focuses on expressing a variety of concepts and relations available in the ECtoloG.

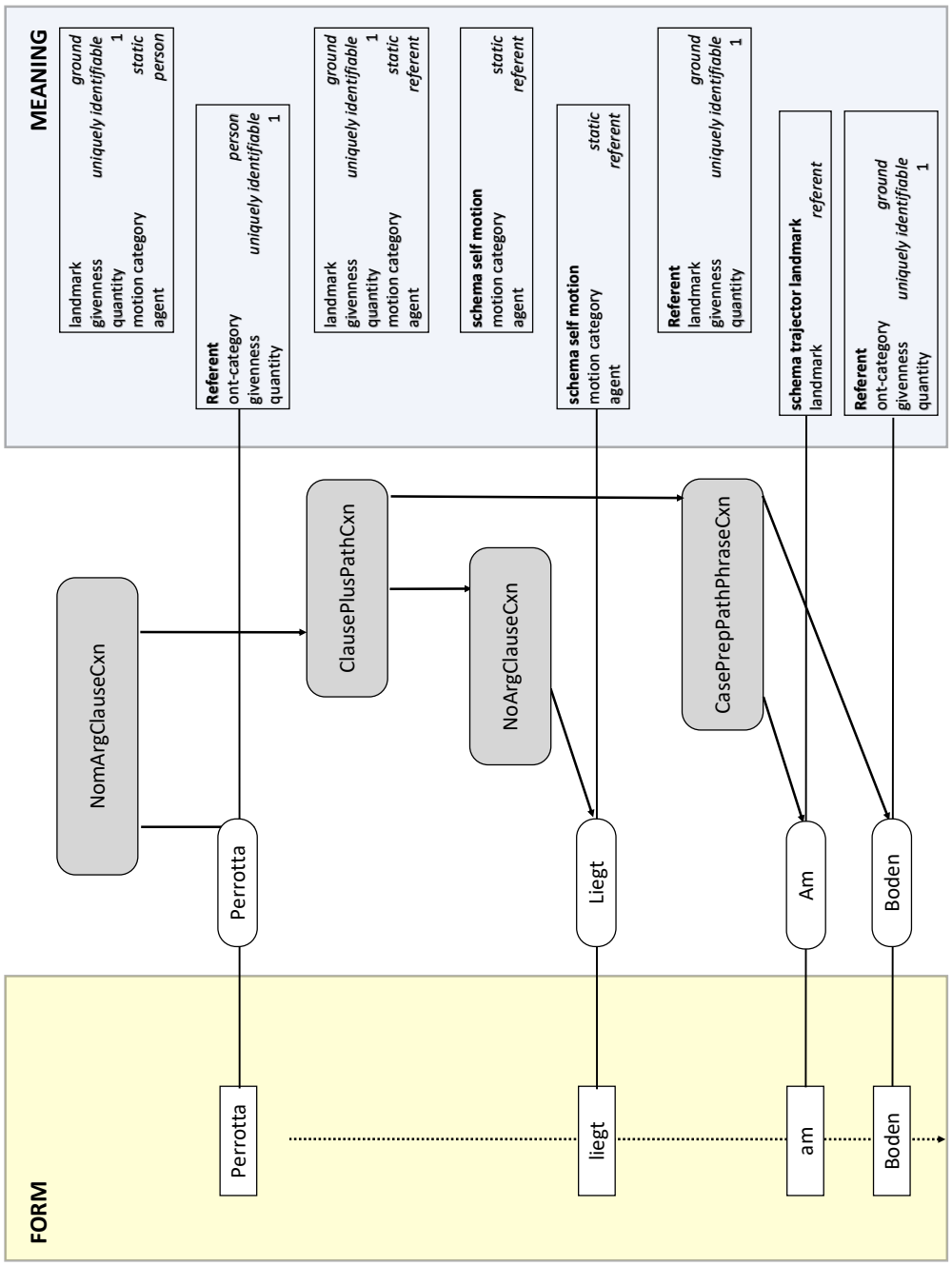


Figure 4.2: Graphical depiction of the informal constructional analysis of the German example sentence *Perrotta liegt am Boden* (engl. *Perrotta is lying on the ground*).

4.3 Constructions in the ECtoloG

A construction in the ECtoloG is – in tune with Goldberg's definition [Goldberg, 1995, p.4] (see also Section 3.1) – a linguistic unit that combines a certain kind of form with a certain kind of meaning of that unit. Constructions in the ECtoloG can exist on various levels of abstraction, i.e. can be as fine-grained or as complex as the engineer wishes them to be. Looking at fine-grained, linguistic information, quite some morphological information is provided through the included LingInfo model as previously described in Section 3.5. The details on the exact integration of the LingInfo model and what information it exactly contains will be further described in Section 4.5. How morphological constructions could be integrated into the model will be briefly discussed in the final discussion of this work.

Comparable to how constructions and schemas are ordered in ECG, they are ordered in an inheritance network in the ECtoloG. While in ECG the `subcase of` operator is used to mark inheritance, in the ECtoloG an `is-a` relation holds between subclasses and their parent or superclasses. This effects that subclasses inherit all of the properties of their respective superclasses. A construction can have more than one superclass and multiple inheritance is, therefore, possible.

In the ECtoloG, constructions are defined as `edns:information-objects`.[4] Their meaning pole is constituted by a schema while their form pole is constituted by instances of the `inf:writing` and `inf:word`[5] classes. A detailed description of their structure and how they are modeled will be described in Section 4.4.

The following presents a more formal definition of constructions in the

[4] `edns` is the namespace of the class `information-object`. It uniquely identifies the class' affiliation to the DnS ontology.

[5] `inf` is the namespace of the classes `writing` and `word`. It uniquely identifies their affiliation to the Ontology of Information Objects.

ECtoloG using Goldberg's definition [Goldberg, 1995, p.4] cited in Section 3.1 as a foundation and adding the second paragraph needed for our purposes. The extension is needed to formally define how exactly a formal construction is built within the ECtoloG, leveraging the ontological foundation provided by embedding it into the DOLCE framework.

Definition Construction in the ECtoloG:
Definition following A. Goldberg: "C is a construction iff_{def} C is a form-meaning pair $< F_i, S_i >$ such that some aspect of F_i or some aspect of S_i is not strictly predictable from C's component parts or from other previously established constructions." [Goldberg, 1995, p.4].

Definition extension for ECtoloG:
C is a subclass of an `edns:information-object` with the following restrictions on its inherited properties:

1. $\exists$ `edns:expresses` some `ECtoloG:schema` and

2. $\exists$ `edns:realized-by` `inf:writing`

This definition ensures that each construction in the ECtoloG is a subclass of the class `edns:information-object` inheriting its properties. These properties in turn are restricted in a way that they only allow exactly one instance of the class of `ECtoloG:schema` to be the filler of the `edns:expresses` property and the class `inf:writing` to be the filler of the `edns:realized-by` property.

More details and how exactly those restrictions and the ECtoloG constructions are modeled in the ontological framework and what they precisely mean will be the topic of the following section.

4.4 Modeling of Constructions in the EC-toloG

The complete, complex network of constructions can be divided into two basic categories of constructions: lexical constructions and compositional constructions. The following sections 4.4.1 and Section 4.4.2, respectively, present a definition of those terms. Section 4.4.3 describes the few additionally needed constructions that do not exactly fit into these categories and gives an explanation why they are required nevertheless.

In ontologies, properties express relations between ontology classes. They ensure that one class (the *domain*) is linked to another class (the so-called *range*) through a certain property (see Section 2.3.2 for a definition). Constructions in the EC toloG are modeled as subclasses of the class `edns:information-objects`. This means that all properties of this class are inherited by constructions, so all constructions share certain properties. The most important ones will be detailed further below.
To create constructions in the ontology, we've first created a subclass of the class `edns:information-objects` that will include all EC toloG constructions called `EC toloG:construction`.[6] According to the specification of the DnS ontology each instance or subclass of `edns:information-objects` inherits – amongst others – the following properties:[7]

- Any `edns:information-object` can be `edns:realized-by` a physical-realization. This physical realization can be an instance of the classes `inf:voice`, `inf:writing`, and `inf:gesture`.

  In other words, the class `edns:information-object` is the *domain* and the class `inf:physical-realization` the *range* of the property `edns:realized-by`.

[6] `EC toloG` denotes the namespace of the class `EC toloG:construction` It uniquely identifies the class' affiliation to the EC toloG ontology. What follows the colon is the name of the class.

[7] For more detailed information on `edns:information-objects` and the class' properties see Gangemi et al., 2004 and Section 3.4.

- Information objects can *express* a description (being linked with the `edns:expresses` property) and a description can in turn be *expressed-by* an information object (as the link via the property is a bi-directional link).

- Information objects are `ordered-by` an information-encoding-system as e.g. a classification system with the `edns:ordered-by` property.

Figure 4.3 graphically depicts what has been described previously and shows the classes and their relations to each other.

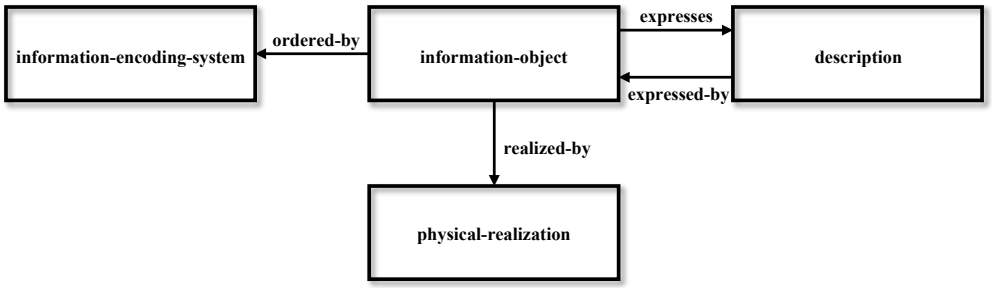


Figure 4.3: Information-objects in the ontological modeling framework and their direct relations to other classes. Arrows are relations and boxes are ontological classes. The arrows show in which direction the relation between the classes holds. You can see that, for instance, the relation between an `edns:information-object` and a `edns:description` is bi-directional.

As a construction constitutes a pairing of form and meaning in tune with the original theory of construction grammar, both its meaning pole and its form pole need to be modeled in the ECtoloG. To be more specific, each constructional instance needs to be linked with its constructional form and its constructional meaning. To accomplish this, both the `edns:realized-by` property and the `edns:expresses` property can be of use. According to the specification of the DnS ontology, `edns:expresses` can be defined as a relation between information objects that are used as representations (signs) and their content, i.e. their meaning or conceptualization. This property is, therefore, used to link the construction to its meaning. In other words, the domain of the `edns:expresses` property

will be an information object and its range will be a description. This will be elaborated in more detail in the next section.
The form pole of the construction, on the other hand, can be modeled with the help of the `edns:realized-by` property. This property designates that a non-physical object is realized by a (physical) representation. For us, this means that a construction – being a non-physical object – can be realized by a physical representation which might be, for instance, its orthographic form. Again, this can be put in other words: The domain of the `edns:realized-by` property is an information object and its range a physical representation. This will again be elaborated more in the next section. Figure 4.4 graphically depicts what has just been explained.

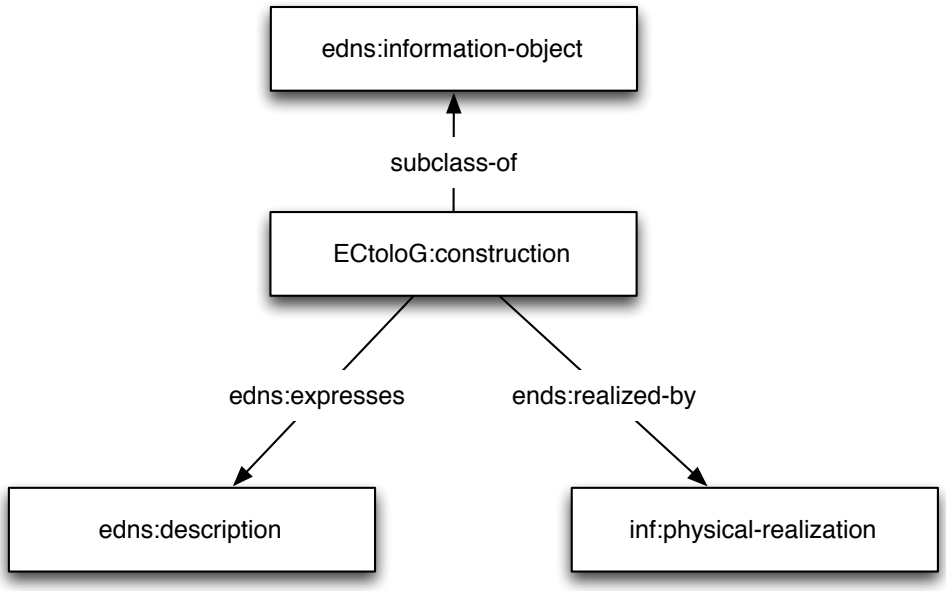


Figure 4.4: A construction in the ECtoloG embedded into in the DOLCE framework. Boxes are ontological classes and arrows relations between those classes.

The ontology class `edns:description` fits perfectly to represent the ontological equivalent of a meaning or a conceptualization representation of the ECtoloG' constructions. This means that the schemas which are supposed to constitute the meaning pole of constructions are modeled as

being a subclass of the class `edns:descriptions`. Thereby, the new class `ECtoloG:schema` inherits the super-classes's properties and among them the `edns:expresses` property which in turn can now be directly used to engineer the meaning pole of the construction.
How exactly the two poles of lexical or compositional constructions are modeled will be described in more detail in the following sections with the help of concrete example constructions.

4.4.1 Modeling of Lexical Constructions

We define the class of lexical constructions that it contains all linguistic units that have a separate entry in a dictionary, i.e. so-called *headwords*. This class can also include idioms like *to kick the bucket* or compound words as, for instance, *soccer player*, since the meaning of none of these linguistic units can completely be determined by the meanings of the distinct words composing it.[8] As previously mentioned, traditional constructions – and more specifically here – lexical constructions constitute form and meaning pairings. The modeling of both their meaning and of their form pole in the ECtoloG will be the topic of this section.

First of all, two new classes are defined:

- `ECtoloG:schema` and
- `ECtoloG:lexicalConstruction`

`ECtoloG:schema` is modeled as a subclass of the class `edns:description` while `ECtoloG:lexicalConstruction` is modeled as a subclass of the class `ECtoloG:construction`. Since that class in turn is a subclass of the class `edns:information-object` the class `ECtoloG:lexicalConstruction` inherits, among other properties, the `edns:expresses` property. Figure 4.5 displays this structure graphically.

[8] See the definition of a construction in Section 4.4.

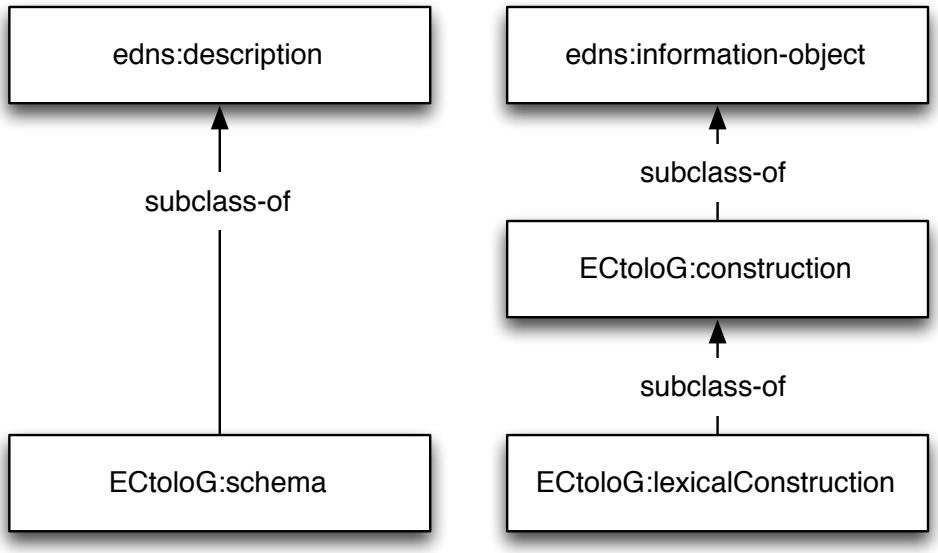


Figure 4.5: Highest level of constructional engineering in the ECtoloG.

Meaning Pole As already mentioned in the previous sections, the meaning poles of constructions can be filled with conceptual schemas. Those schemas can represent the (dynamic) semantics of constructions. The EC-toloG follows ECG in this area and includes *frame-based knowledge* and – more precisely – *executing schemas*, and *image schemas*.[9]
To assign a meaning to a lexical construction it has to be made explicit where exactly in the ontology that schematic meaning is encoded, i.e. the link between the construction itself and the construction's meaning has to be defined. To fill the meaning pole of a lexical construction with a schematic meaning, a restriction on the `edns:expresses` property is defined:

- ∃ `edns:expresses` some `ECtoloG:schema`

This restriction determines that at least one of the `edns:expresses` property's values is of type `ECtoloG:schema`. Modeling this restriction is done by means of the built-in `owl:someValuesFrom` constraint. The restriction

[9] See 2.4.1 for a definition of those terms and the Sections 4.6 and 4.7 for more information on the way of precisely modeling them in the ontology.

holds for all constructions that express a schema, that is for the complete class `ECtoloG:lexicalConstruction`. It has obviously no effect on the whole class of constructions, i.e. it is possible that there are constructions that do not express a single schema. Compositional constructions, for instance, whose meaning might be a composite of all constructions and schemas that constitute that compositional construction (and even more) do not express an instance of a single schema (see Section 4.4.2).

As mentioned in the previous section, `edns:expresses` is defined as a relation between information objects that are used as representations (signs) and their content, i.e. their meaning or conceptualization. Content, then, is reified as a description, which constitutes the reason why the class of `ECtoloG:schema` has been modeled as such. What exactly can be subsumed under the class `ECtoloG:schema` and a detailed description of the modeling of those schemas will be described in more detail in Section 4.7.

Form Pole The form pole of each construction is modeled with the help of the `edns:realized-by` property, defined in the previous section. It is inherited from the class `edns:information-object`, the superclass of `ECtoloG:construction`.
The class `edns:physical-realization` fills the range of the property `edns:realized-by`. This class is the superclass of the classes `inf:voice`, `inf:writing` and `inf:gesture`. The form pole of lexical constructions in the ECtoloG is supposed to be 'filled' with the orthographic form of the word the construction describes. Therefore, an instance of the class `inf:writing` is defined, which then fills the form pole of the respective construction. This instance has inherited once again a relation which connects it to instances of the class `inf:word`. It is predefined that instances of `inf:word` can be realized by instances of the `inf:writing` class. As determined in the ontology, words can be realized in various ways, i.e. they can, for instance be uttered or written. With linking the class `inf:word` to `inf:writing` we define that the forms of lexical constructions have to be written realizations of words. This is realized in an analogous way

as the meaning pole has been implemented: A restriction on the class `ECtoloG:lexicalConstruction` is defined. This restriction is listed in the following:

- ∃ `edns:realized-by` `inf:writing`

This restriction states that at least one value of the `edns:realized-by` property has to be of type `inf:writing`.
Figure 4.6 graphically depicts how lexical constructions are modeled in the ECtoloG.

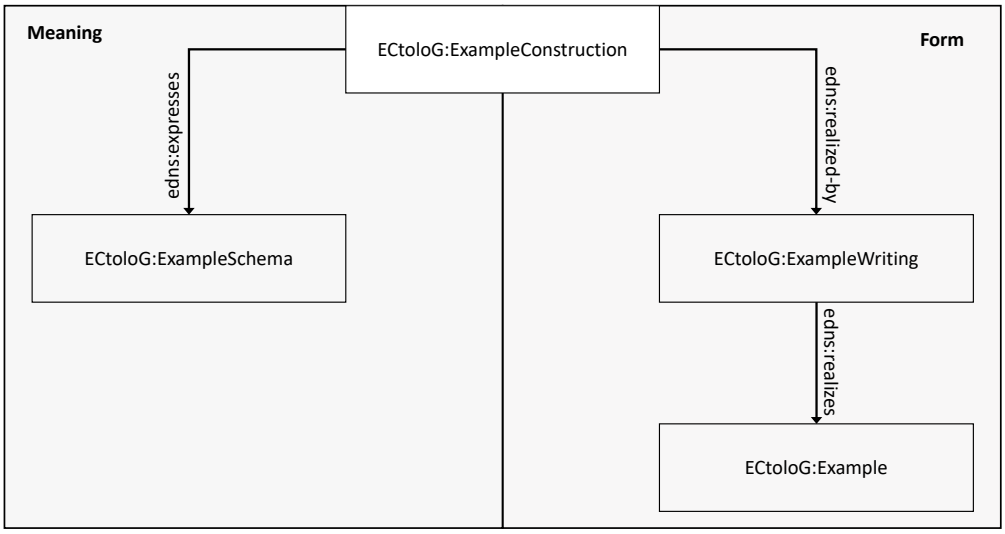


Figure 4.6: Lexical Constructions in the ECtoloG: The boxes are constructions, the arrows are the relations that hold between them. On their meaning side, constructions express an `ECtoloG:schema`. On their form side, they are realized by an `inf:writing` which in turn realizes an `inf:word`.

The class `ECtoloG:lexicalConstruction` captures classes for all common parts-of-speech as listed in standard dictionaries, as for instance the classes `ECtoloG:determinerConstruction`, or `ECtoloG:verbConstruction`, as graphically displayed in Figure 4.7.
Subclasses of the class `ECtoloG:lexicalConstruction` which additionally make reference to some object in the world, like for instance classes that capture common nouns, pronouns, or proper names are in addition

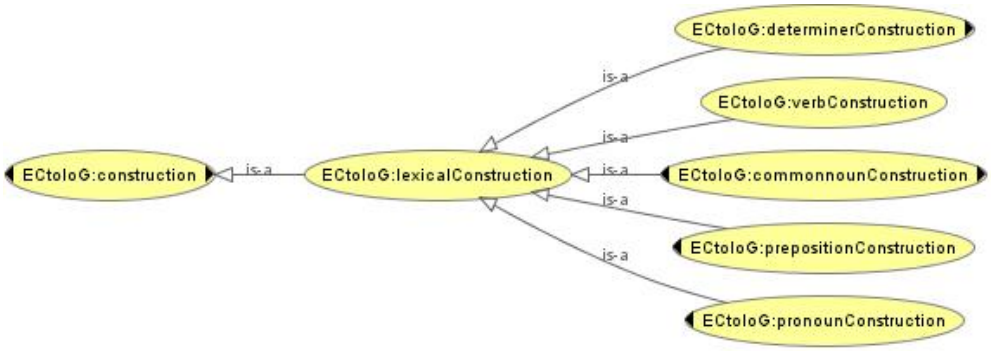


Figure 4.7: Example lexical constructions in the ECtoloG.

subclasses of the class `ECtoloG:referringConstruction`. This class is a subclass of the class `ECtoloG:construction` and inherits again – among others – the `edns:expresses` property from that class. A restriction has been imposed on this class stating that the values of the `edns:expresses` property have to be of type `ims:entity_schema`, the most upper class of the image schema ontology, described in more detail in Section 4.6.[10]

The following paragraph presents a concrete example of what has previously been explained.

A Concrete Example

The concrete example sentence – repeated here for the reader's convenience – is the following:

(2) Perrotta liegt am Boden.
Perrotta is lying on the ground.

The four lexical constructions that are needed to parse or produce that sentence are constructions for each single word, i.e. for *Perrotta*, *liegt*, *am*, and for *Boden*. Each lexical construction will be described in detail in the following. You will find more detailed information on the schemas

[10] `ims` denotes the namespace of the class `ims:schema`. It uniquely identifies the classÕ affiliation to the image schema ontology.

that will be mentioned and how they are embedded in the image schema hierarchy in the respective Section 4.6

ECtoloG:PerrottaConstruction: As a first step, a new subclass of `ECtoloG:LexicalConstruction` is created which is called `ECtoloG:SecondnameConstruction`. Additionally, this class is a subclass of `ECtoloG:referringExpressionConstruction` since constructions for last names make a reference to an entity. Then, a subclass of this class has to be defined called `ECtoloG:PerrottaConstruction`.
It is necessary, to be able to treat this class as an instance in order to assign additional linguistic information to it with the help of the later described LingInfo model. Therefore, it is additionally typed as a subclass of the class `LingInfo:ClassWithLingInfo`. Which linguistic information is assigned to this construction and how exactly this is done in general and more specifically with this construction will be described in Section 4.5.1.

Meaning Pole: With the help of the inherited `edns:expresses` property, the construction `ECtoloG:PerrottaConstruction` is linked to an instance of the `ims:FootballplayerSchema` called `ECtoloG:PerrottaSchema`.
Through this relation the meaning pole of the construction in question is defined.
The meaning of the `ECtoloG:PerrottaConstruction` might be presented by the following Schema displayed in Figure 4.8.
Form Pole: To define the `ECtoloG:PerrottaConstruction`'s form pole, the inherited property `edns:realized-by` is used. It links the class to an instance of `inf:writing`. This instance has to be created in the ontology and is called `ECtoloG:PerrottaWriting`. To link this instance in turn to its orthographic representation, an instance of the class `inf:word` has to be created. It is called `ECtoloG:Perrotta` and is linked through the `edns:realizes` property to the instance `ECtoloG:PerrottaWriting`. Figure 4.9 displays the classes, instances and their relations towards each other.

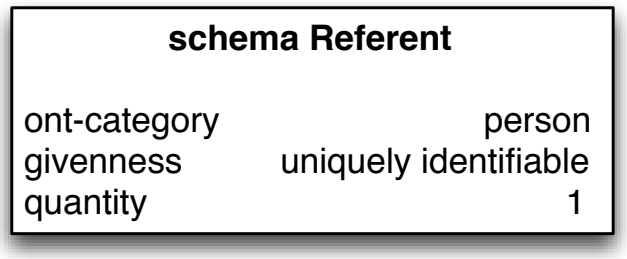


Figure 4.8: A possible presentation of the `Referent` schema in the ECtoloG.

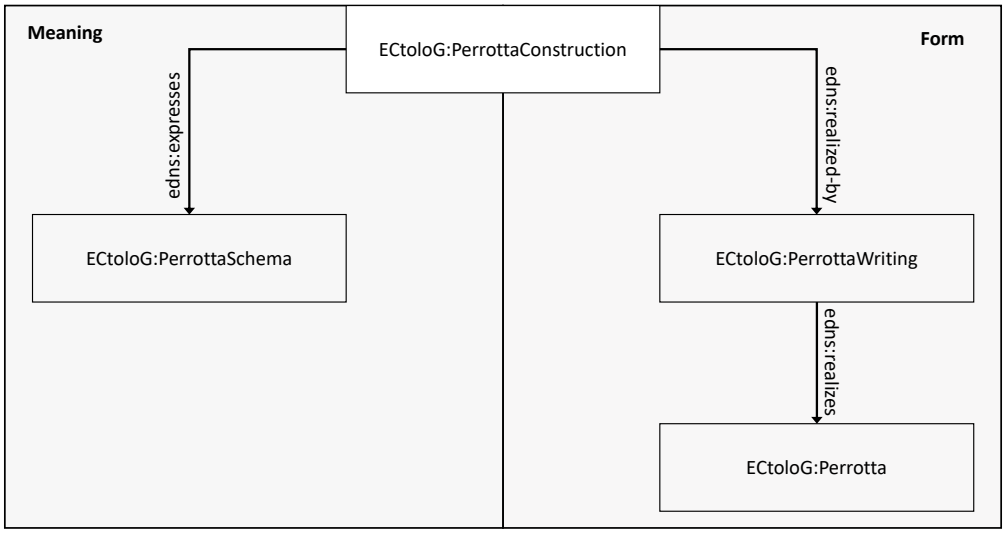


Figure 4.9: The `ECtoloG:PerrottaConstruction` in the ECtoloG. The boxes are constructions, the arrows are the relations that hold between them. On their meaning side, constructions express an `ECtoloG:schema`. On their form side, they are realized by a subclass of the class `inf:writing` which in turn realizes a subclass of the class `inf:word`.

ECtoloG:LiegtConstruction: First, `ECtoloG:VerbConstruction` is defined being a subclass of `ECtoloG:LexicalConstruction`. This class subsumes all constructions defining verbs. A restriction is imposed on that construction that states that all values of the `edns:expresses` property have to be of type `ims:x_schema`. This ensures that any verb meaning

contains an x-schema.[11] The further modeling of the construction is done in an analogous way as it has been specified in the previous paragraph for the lexical construction `ECtoloG:PerrottaConstruction`: The construction is also an instance of the class `LingInfo:ClassWithLingInfo`. Additionally, it is linked via the `edns:expresses` property to an instance of the class `ims:x_schema`: the `ECtoloG:LiegenSchema`. This schema is in turn an instance of both the class `ims:static_schema` and the class `ims:self-Motion-XSchema`. You will find more information on those schemas in Section 4.6 and its following sections.

ECtoloG:AmConstruction: A subclass of the class containing lexical constructions is defined that is called `ECtoloG:PrepositionConstruction`. Again, a class is defined being the subclass of that construction called `ECtoloG:CasePrepConstruction`. The `ECtoloG:AmConstruction` is defined as an instance of that class. Similar to the other lexical constructions, this class is of type `LingInfo:ClassWithLingInfo`.
A condition which constrains that all values of the `edns:expresses` property have to be of type `ims:trajector-landmark_schema` is imposed on the class `ECtoloG:CasePrepConstruction`.

ECtoloG:BodenConstruction: This construction is an instance of the class `ECtoloG:noun-maleSgNomConstruction` which constitutes a subclass of the class `ECtoloG:CommonnounConstruction` again a subclass of the class `ECtoloG:LexicalConstruction`. As with all lexical constructions, the instance is also of type `LingInfo:ClassWithLingInfo` to enable adding linguistic information to it.
A constraint is imposed on the class `ECtoloG:CommonnounConstruction` saying that the meaning pole of this construction's instances is restricted to be of type `ims:entity_schema`. In this specific case, the construction is linked to the `ECtoloG:BodenSchema`, an instance of the `ims:Ground_Schema`

[11] This modeling step should be improved in future since not all verbs are action verbs. For the time being we model this that way, since for the test domain (which is going to be soccer) this seems to be true for most of the verbs.

via the `edns:expresses` property. The form pole is modeled according to the standard way of modeling it: linking the construction to an instance of `inf:writing` – called `ECtoloG:BodenWriting` – and that instance via the `edns:realizes` property to an instance of `inf:word` – called `ECtoloG:Boden`, respectively.
The following section describes how more complex constructions are modeled in the ECtoloG.

4.4.2 Modeling of Compositional Constructions

Compositional constructions are complex constructions that are on a higher level of abstraction than lexical ones. They are constructions that combine more than one construction into one unit, i.e. into a compositional construction. A simple example is the `DeterminerNounConstruction` that combines a determiner and a noun into a determined noun phrase. A more complex construction is, for instance, a clause construction like the `ClausePlusPathConstruction`, combining a clause with a path specifier. A concrete example of that compositional construction is given below.
Let's have a look at how to exactly represent compositional constructions in the ECtoloG.

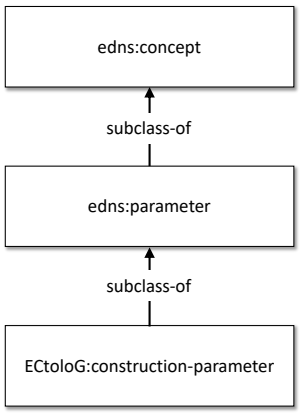


Figure 4.10: Step 1 to model complex constructions in the ECtoloG. A class `ECtoloG:construction-parameter` is defined as a subclass of the class `edns:parameter`, which in turn is a subclass of `edns:concept`.

Initially, a class `ECtoloG:construction-parameter` is defined that denotes a subclass of the class `edns:parameter`, which in turn is a subclass of `edns:concept` (see Figure 4.10).

According to the DnS ontology's specification, the class `edns:parameter` represents the qualities of *perdurants* or of *endurants* (see 2.3.4 for a definition and a basic concept hierarchy giving examples). While *perdurants* describe processes or events, *endurants* define objects or substances. They can also be *requisites* for some role or course. Requisites are defined being constraints over the attributes of entities. In that line, a `edns:requisite-for` relation holds between `edns:parameter` and the concept for roles, figures or courses (see Figure 4.11).

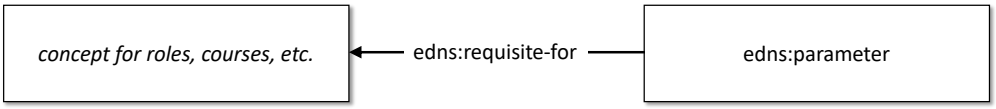


Figure 4.11: The `edns:requisite-for` relation linking `edns:parameter` (its domain) with the concept for roles, figures, or courses (its range).

Now, we find this relation to be exactly the right one to combine more than one construction into one unit, i.e. to be able to engineer complex constructions. It will be used to eventually link the following two classes to each other: `edns:construction-parameter` and `ECtoloG:construction`. To implement this linkage with the `edns:requisite-for` property, first, the class of `edns:information-object` being added to the range of the `edns:requisite-for` property.
Eventually, this results in the class `edns:parameter` being the domain of the `edns:requisite-for` property and `edns:information-object` its range (see Figure 4.12).
`ECtoloG:construction` being a subclass of `edns:information-object` and the class `ECtoloG:construction-parameter` in turn a subclass of the class `edns:parameter`, results in the fact that those classes inherit their parent classes' properties. Therefore, `ECtoloG:construction-parameter` can now fill the domain of the `edns:requisite-for` property and the class

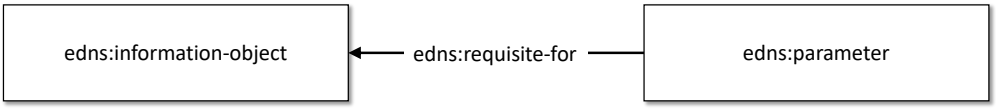


Figure 4.12: The `edns:requisite-for` relation linking `edns:parameter` (its domain) with the class `edns:information-object` (its range).

`ECtoloG:construction` its range (see Figure 4.13).

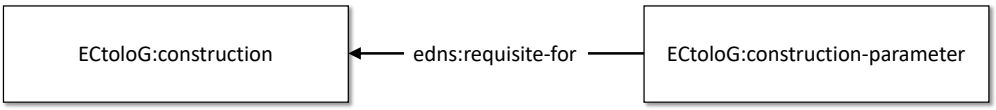


Figure 4.13: The `edns:requisite-for` relation linking the class `ECtoloG:construction-parameter` (its domain) with the class `edns:construction` (its range).

As a next step, we use the `edns:requisite-for` property to constrain the attributes of one of their entities, concretely, we define a property restriction that is mainly necessary for `ECtoloG:construction-parameter` stating that all values of the `edns:requisite-for` property have to be of a certain type, i.e. `ECtoloG:construction`. The use of the term *only* determines exactly that.

- ∀ `edns:requisite-for` only `ECtoloG:construction`

This restriction determines that from now on all instances of the class `ECtoloG:construction-parameter` can be included into a compositional construction, and, therefore, be used to form compositional constructions. One last step is, however, still missing for lexical constructions to be united in complex ones: To enable the usage of lexical constructions as components of more abstract constructions, all lexical constructions are defined to be instances not only of the class `ECtoloG:lexicalConstructions` – a subclass of the class `ECtoloG:construction` – but also of the class `ECtoloG:construction-parameter`. Figure 4.14 presents a final summary of what has been described in the previous paragraphs.

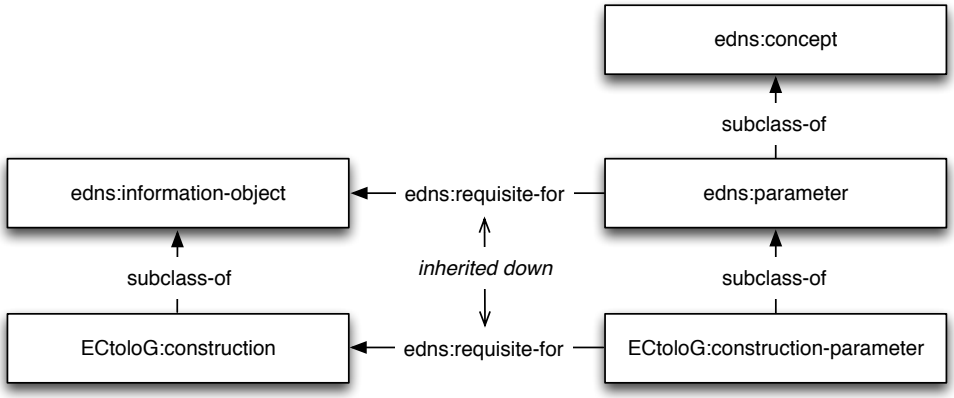


Figure 4.14: Prerequisites to model complex constructions in the ECtoloG.

Now we have defined how complex constructions can theoretically be implemented. What is still needed to concretely model specific ones, i.e. to determine exactly which constructions are supposed to be used in exactly which more abstract construction, new properties have to be defined. These properties are all sub-properties of the `edns:requisite-for` property. The number of subproperties of the `edns:requisite-for` property that are being implemented, depends on the number of components a complex construction can possibly have.

To account for the form constraints which can hold in compositional constructions, the properties have been indexed with increasing numbers: 1 is followed by 2 etc.

The following paragraphs describe those complex constructions in detail that are needed to for the analysis of the example sentence.[12]

As having been elaborated in Section 4.2, the following compositional constructions are included in our analysis of the example sentence, putting all of the sentence's components into their appropriate order:

- ECtoloG:NoArgClauseConstruction

[12]Please keep in mind that it is a subjective undertaking which constructions are considered needed for a satisfying analysis.

- ECtoloG:NomArgClauseConstruction
- ECtoloG:ClausePlusPathConstruction
- ECtoloG:CasePrepPathPhrase

First of all, the class `ECtoloG:ClauseConstruction` is created which constitutes a subclass of the class `ECtoloG:CompositionalConstruction`. At the same time, the class `ECtoloG:ClauseConstruction` is made a subclass of the class `ECtoloG:construction-parameter`, as well, since clauses often are a part of more complex constructions. This way of modeling enables the use of all clause constructions in more complex constructions that are on a higher level of abstraction than simple clauses.

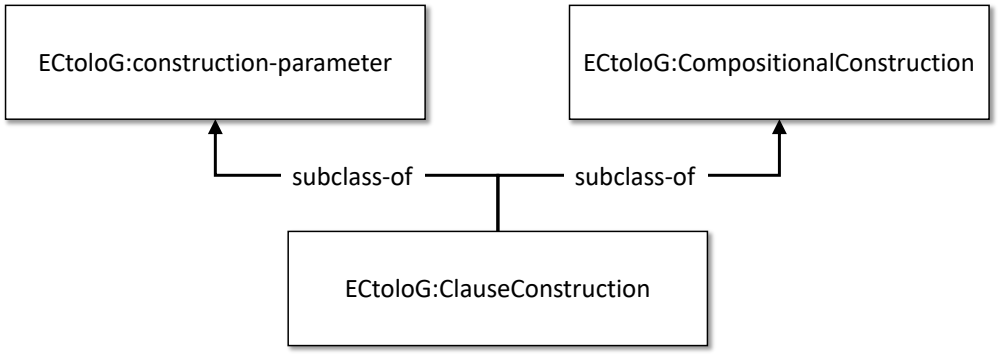


Figure 4.15: The class `ECtoloG:ClauseConstruction` as subclass of `ECtoloG:CompositionalConstruction` and `ECtoloG:construction-parameter`.

We will define that the meaning pole of a complex construction is determined to be a simple predication which is modeled in the ontology as a `ims:predication_schema`. To ensure this, we need to impose a restriction on the `edns:expresses` property of `ECtoloG:ClauseConstruction` that states that at least one value of the `edns:expresses` property has to be of type `ims:predication_schema`.

- ∃ `edns:expresses` some `ims:predication_schema`

Figure 4.16 displays a summary of what has been described.

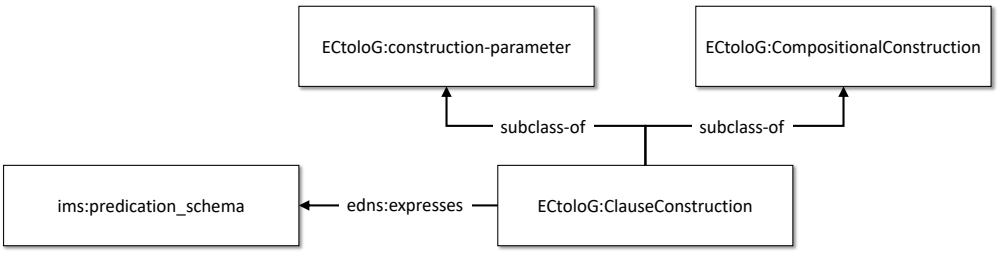


Figure 4.16: Clause constructions in the ECtoloG: The class `ECtoloG:ClauseConstruction` as a subclass of both classes `ECtoloG:CompositionalConstruction` and `ECtoloG:construction-parameter` and linked as domain to the range `ims:predication_schema` via the `edns:expresses` property.

For the analysis of the example sentence, first, the three subclasses of the class `ECtoloG:ClauseConstruction` are defined:

- `ECtoloG:NoArgClauseConstruction`
- `ECtoloG:NomArgClauseConstruction`
- `ECtoloG:ClausePlusPathConstruction`

Furthermore, the class `ECtoloG:PathSpecifier` is created. It is a subclass of the class `ECtoloG:Construction` and `ECtoloG:CasePrepPathPhrase` is its direct subclass. Figure 4.17 summarizes the network and inheritance relations of those complex constructions in the ECtoloG. All mentioned constructions will be described in more detail in the following paragraphs.

ECtoloG:NoArgClauseConstruction: As previously mentioned, this clause construction has as its only constituent an inflected verb. Just like all clause constructions, the construction inherits the `edns:requisite-for` property from its parent construction. As the construction itself has only one constituent, only one sub-property of `edns:requisite-for` is created. This property is called `ECtoloG:requisite_NoArgClauseCxn`. A restriction is imposed on the construction stating that all values of the `ECtoloG:requisite_NoArgClauseCxn` property have to be of type

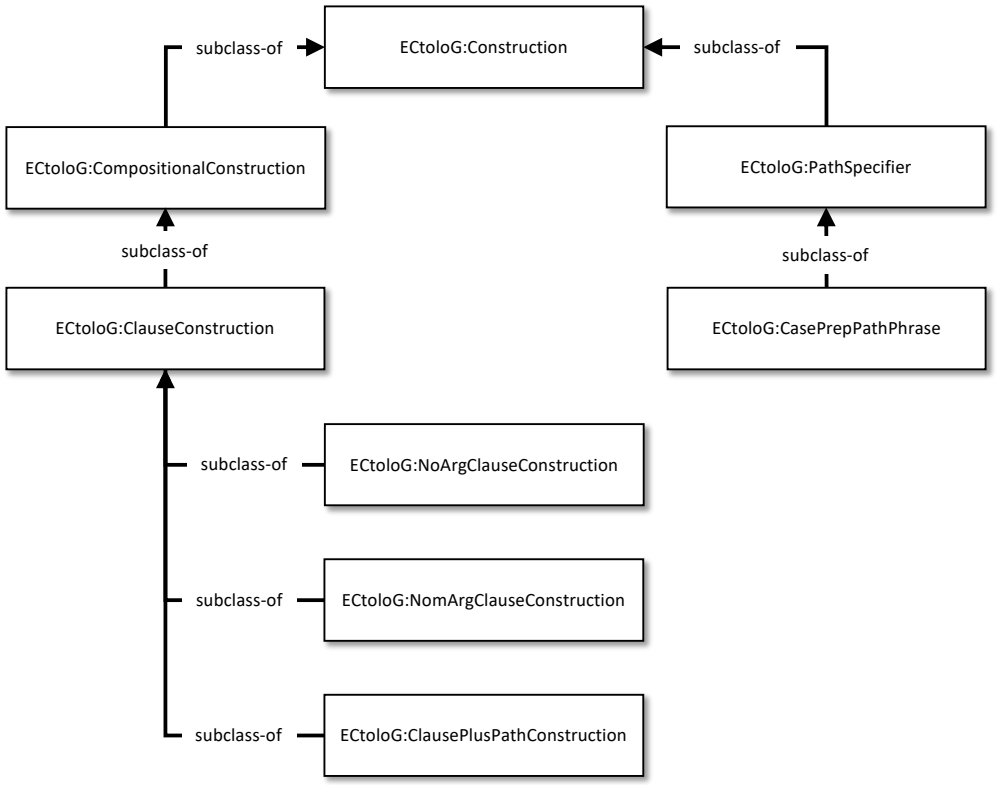


Figure 4.17: Compositional constructions and path specifier in the ECtoloG.

`ECtoloG:verbConstruction`. This way of modeling ensures that the single component of that construction is of type `ECtoloG:verbConstruction` (see Figure 4.18).

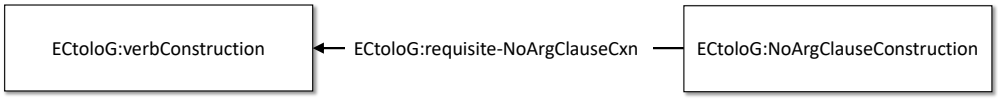


Figure 4.18: The `ECtoloG:NoArgClauseConstruction:` linked via the `ECtoloG:requisite_NoArgClauseCxn` property to `ECtoloG:verbConstruction`.

ECtoloG:NomArgClauseConstruction: This clause construction can include several constituents: referring expressions in the nominative and a clause construction of any kind. Two sub-properties of `edns:requisite-for`

are defined, at first, to ensure that this construction can actually combine two other constructions into one unit and, furthermore, to ensure that the first constituent precedes the second constituent.
The sub-properties are called `ECtoloG:requisite_NomArgClauseCxn1` and `ECtoloG:requisite_NomArgClauseCxn2`.
A restriction is imposed on the construction stating that all values of the property `ECtoloG:requisite_NomArgClauseCxn2` have to be of the specific type `ECtoloG:ClauseConstruction`. In this way the type of the second component of that construction is determined. The just mentioned way of constructional engineering also ensures that simple sentences like *Perrotta sits* consisting of a referring expression and an inflected verb (being a NoArgClauseConstruction) would be captured by this construction as well as more complex ones like our example sentence.
The second restriction states that the construction's second component has to be of type `ECtoloG:ReferringExprCxn`, i.e. the range of the `ECtoloG:requisite_NomArgClauseCxn1` property is filled by the class `ECtoloG:ReferringExprConstruction`. Figure 4.19 shows the modeling of the form pole of this construction in the ECtoloG.

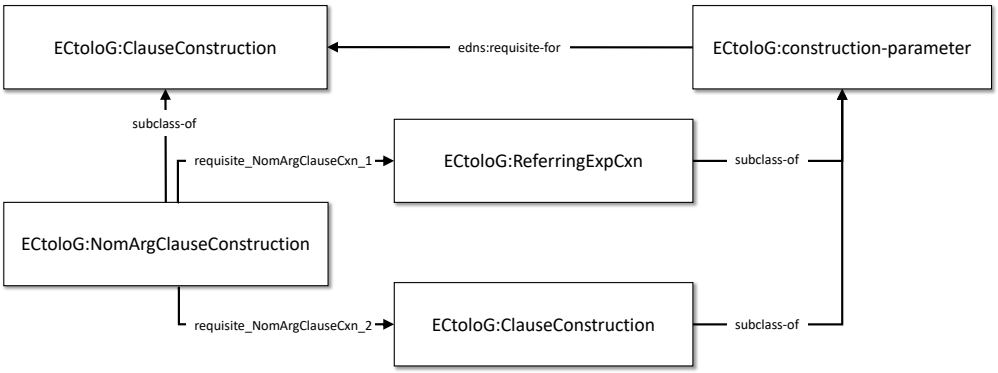


Figure 4.19: The form pole of the `ECtoloG:NomArgClauseConstruction` in the ECtoloG.

The restriction on the inherited `edns:expresses` property that links a clausal construction to its meaning states that at least one value of that property has to be of type `ims:predication_schema`.[13] This way, this

[13] See Section 2.4.1 on more information on the predication schema.

clausal construction adds the meaning encoded in the predication schema to the whole sentence.

ECtoloG:ClausePlusPathConstruction: This construction combines any kind of clause with any kind of path specifier. Therefore, it needs two subproperties of the `edns:requisite` property, one for each of its constituents. The first subproperty states that its range has to be of type `ECtoloG:ClauseConstruction` and the second one that its range has to be of type `ECtoloG:PathSpecifierConstruction`. As depicted in Figure 4.17, the class `ECtoloG:PathSpecifierConstruction` is directly a subclass of `ECtoloG:Construction` (see Figure 4.20). Its meaning pole is filled with a `ims:trajector-landmark_schema`, adding this meaning to the clause `ECtoloG:ClausePlusPathConstruction` that is being constructed.

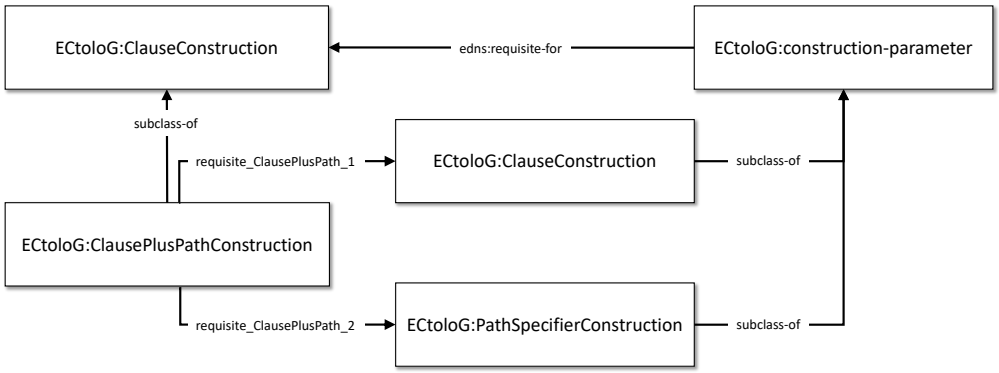


Figure 4.20: `ECtoloG:ClausePlusPathConstruction`: Its form pole modelled in the ECtoloG.

ECtoloG:CasePrepPathPhraseConstruction: This clausal construction combines a preposition and a referent into one phrasal unit. It is a subclass of the class `ECtoloG:PathSpecifierConstruction`. As the class has two constituents, two subproperties of the `edns:requisite` property are defined. For each of those properties constrains are defined: While values of the `ECtoloG:requisite_CasePrepPathCxn1` have to be of type `ECtoloG:CasePrepConstruction`, values of the property

`ECtoloG:requisite_CasePrepPathCxn2`, however, have to be of type `ECtoloG:referringExpressionCxn` (see Figure 4.21).

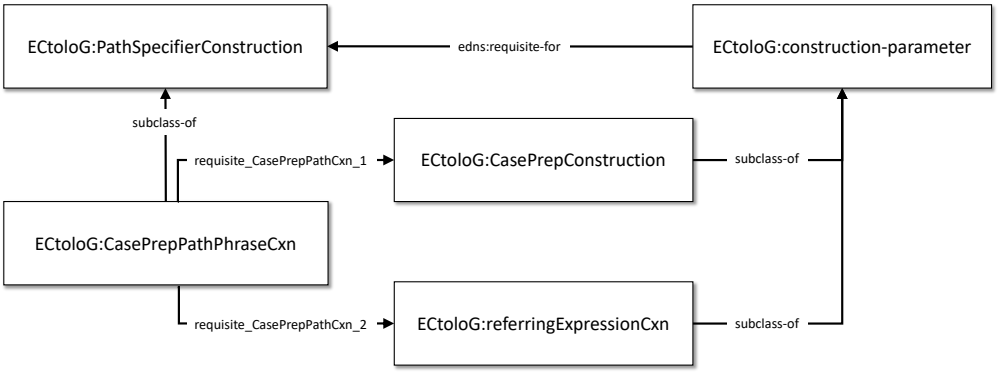


Figure 4.21: `ECtoloG:CasePrepPathPhraseConstruction`: Its form pole modelled in the ECtoloG.

Additionally, a constraint is imposed on the `edns:expresses` property stating that at least one value of the `edns:expresses` property has to be of type `ims:trajector-landmark_schema` which results in the fact that the meaning pole of this construction is denoted by that specific schema.

4.4.3 Modeling of Other Constructions

During the course of the last section, several constructions have been mentioned, that have not directly been addressed in the previous informal constructional analysis in Section 4.2. They are, however, necessary to successfully analyze the example sentence, including to completely compose its meaning.[14] The following briefly describes those constructions.

ECtoloG:ReferringExpressionConstruction: This class is a subclass of the class `ECtoloG:construction` and includes referring expressions, i.e. constructions that make reference to any kind of entity. Concrete examples from our example sentence are, for instance, lexical constructions like the `ECtoloG:BodenConstruction` for the German word *Bo-*

[14]Please note that with 'complete meaning' we mean our subjective understanding of completeness.

den (engl. *ground*) or even more complex constructions like, for instance, the `ECtoloG:CasePrepConstruction`. The meaning pole of the ECtoloG:ReferringExpressionConstruction is filled with a subclass of the class `ECtoloG:schema`, i.e. with an `ims:Entity-Schema` that denotes the meaning of entities of any kind (see Figure 4.22).

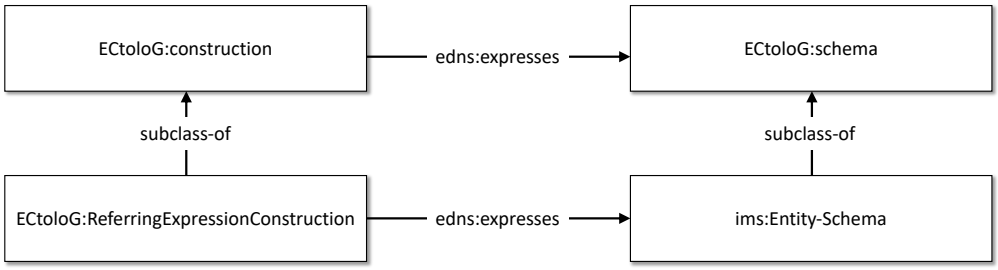


Figure 4.22: How the `ECtoloG:ReferringExpressionConstruction`'s meaning pole is filled with `ims:Entity-Schema`.

ECtoloG:PathSpecifierConstruction: This construction is a direct subclass of the `ECtoloG:construction`. Its meaning pole is filled with a `ims:trajector-landmark-schema`. This happens through making it the range of the `edns:expresses` property.
By linking the construction to the `ims:trajector-landmark-schema`, its meaning is added to the clause that is being built by the construction (see Figure 4.23).

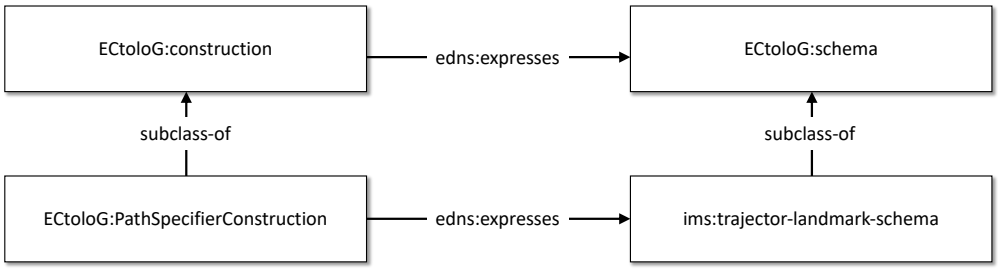


Figure 4.23: How the `ECtoloG:PathSpecifierConstruction`'s meaning pole is filled with `ims:trajector-landmark-schema`.

ECtoloG:CasePrepConstruction: This construction is a direct subclass of `ECtoloG:construction`. The meaning pole of the construction is filled with a `ims:trajector-landmark-schema` whose meaning is added to the clause that is being built by the construction (see Figure 4.24).

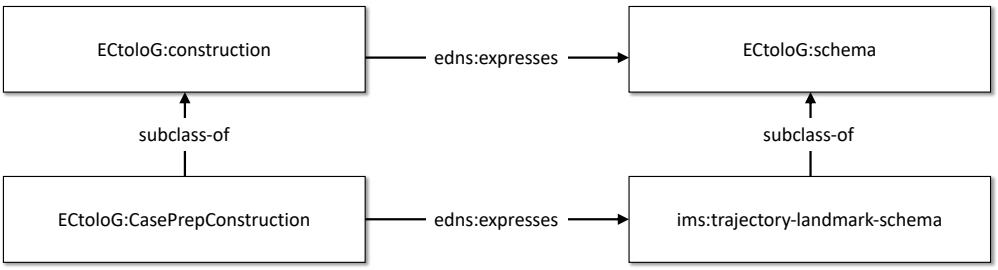


Figure 4.24: How the `ECtoloG:CasePrepConstruction`'s meaning pole is filled with `ims:trajector-landmark-schema`.

More information on the sentence's semantics and the mentioned image schemas will be given below in Section 4.6.

4.5 Linguistic Information

Since linguistic information as grammatical gender, case, number, person or part-of-speech of a word is needed for automatically analyzing natural language texts, this information needs to be modeled, as well, in the ECtoloG. The following sections firstly describe which linguistic information we want to have encoded in the various constructions. Then our approach of representing the mentioned information is described in detail.

4.5.1 Linguistic Information in Constructions

Constructions – mostly lexical ones but also more complex constructions describing referring expressions – need linguistic information to enable their successful application in higher level, i.e. compositional constructions. With *successfully* we mean that, for example, the possibility to check the agreement between a determiner construction and a noun construction regarding case, number and gender, needs to be given.

The information that is needed in lexical constructions is common linguistic information as listed in standard dictionaries.

The German language has four different cases allowing flexibility in word order since the actors of a sentence are marked by case and not by their position in a sentence, as opposed to the English language where the role of a referent can be determined by its position in a sentence, as English has a strict SVO word order. The cases that need to be encoded in common noun constructions are *nominative*, *genitive*, *dative*, and *accusative*. Additionally, information on number (*singular* or *plural*) and grammatical gender of that noun in question has to be marked.
Constructions describing verbs have to include information on its number, person, the verbform (*inflected*, *infinitive*, *past participle* and *imperative*) and its tense (*past* or *present*).
In constructions describing determiners, case information, number and grammatical gender needs to be encoded, as well. As can be seen, the same information as for nouns is needed, since in the analysis process agreement between determiner and noun constructions needs to be checked.

There is, of course, a vast amount of design choices that can be taken now how to successfully encode grammatical and/or linguistic information in a construction grammar. Within the same grammar formalisms, different grammar engineers might take different decisions in how to encode this kind of information. In ECG, for instance, we have decided to encode case information in separate constructions which serve as parent constructions for the constructions that hold a specific kind of case, whereas another ECG grammar engineer might encode case information directly in the lexical construction itself.
In any case, the design choice we took for this work is to put linguistic information in an ontological module that will be adapted to the ontological modeling framework (see the next section for more details). This way, even complete ontology classes or properties of the ECtoloG can be

assigned to specific linguistic features as listed above, which presents the main advantage of this approach besides the high reusability factor. Information like this is mostly missing in today's ontologies. Sometimes it is represented in little elaborate fashion, leaving the semantic information that is encoded in the ontology without grounding to the human linguistic and cognitive domain. There are various approaches to model linguistic information in ontologies as, for example, SKOS[15], or alternative lexicon models (see Section 2.3.5 for an overview). One of the main problems is that most of these approaches put their focus on e.g. the definition of a top ontology for lexicons and not on the assignment of linguistic features for domain ontology classes and properties as desired in our case.[16]

There exists one model that partly fulfills the mentioned demands and which we designed together with different partners within the SmartWeb project [Wahlster, 2007] to provide the SmartWeb ontology with linguistic information, i.e. the LingInfo model (see [Buitelaar et al., 2006] and Section 3.5). We decided to model linguistic information in the ECtoloG partly inspired by the structure of that model but integrating it more tightly into the DOLCE/DnS/OIO framework. Therefore, it can be of use for other ontologies in need of linguistic information, as well, provided that those ontologies are based on the same foundational ontology. Additionally, we avoid modeling redundant classes and properties and reuse what is possible of the beforehand set-up framework.

The following section presents a detailed description of the integration of linguistic information into the ECtoloG.

[15] http://www.w3.org/TR/swbp-skos-core-guide/
[16] See also Section 3.5 on more details on those approaches.

4.5.2 Modeling Linguistic Information in the ECtoloG

A first consideration has been to integrate the LingInfo model as is into the ontological DOLCE framework where the ECtoloG is embedded. However, we quickly encountered that the integration of the original LingInfo model into the SmartWeb ontologies brought about several disadvantages:

1. The LingInfo model is quite separate from the ontology it is integrated into. All LingInfo classes can be found at the same level as the most upper class `owl:class`.

2. There are various classes and properties in the *Ontology of Information Objects* that are equivalent in their meaning to the properties and classes of the LingInfo ontology which results in redundantly encoded information in the ontology.

A further point is that, generally speaking, linguistic information is implemented in a less elaborate way in the OIO as it is defined in the LingInfo model. Due to the mentioned shortcomings and the fact that linguistic information could be modeled more elaborate in the ontological framework we use in this work, we believe it makes sense to adapt the LingInfo ontology exactly to our needs and integrate it more tightly into the ontological modeling framework that was described in the Section 2.3, to be more specific, into the *Ontology of Information Objects* to receive a tightly integrated linguistic component in our ECtoloG and to enable the employment of this version of the LingInfo ontology within the foundational framework of DOLCE/DnS/OIO within other domain ontologies. Classes and properties of the OIO and DnS which are of value for our model are adopted, classes and properties that are missing but necessary in order to be able to represent the linguistic information listed above are added to the model and are given the namespace `LingInfo` to uniquely identify their affiliation to the LingInfo model.

To create the LingInfo model within the ontological framework a so-called

meta-class needs to be introduced. A meta-class is a class whose instances are themselves classes. This way you can ensure that a class can have properties which are otherwise only applicable to instances but at the same time retain the attributes of ontology classes.[17] Meta-classes are instances of `owl:class`. The class `LingInfo:ClassWithLingInfo` is created as an instance of `owl:class` and is therewith a meta-class. The class `LingInfo:ClassWithLingInfo` is linked via the `LingInfo:linginfo` property to the class `LingInfo:Linginfo`. Instances of the so-called *meta-class* `LingInfo:ClassWithLingInfo` are, therefore, again linked through the `LingInfo:linginfo` property to instances of the `LingInfo:LingInfo` class.

To ensure that constructions can be linked with their respective linguistic information that is modelled in this ontology, every construction in need of linguistic information is an instance of `LingInfo:ClassWithLingInfo`. Every linguistic information is modelled as instance of `LingInfo:LingInfo`. Figure 4.25 displays the inheritances of those classes and their relations towards each other. A concrete example below will add further understanding to this theoretical modelling.

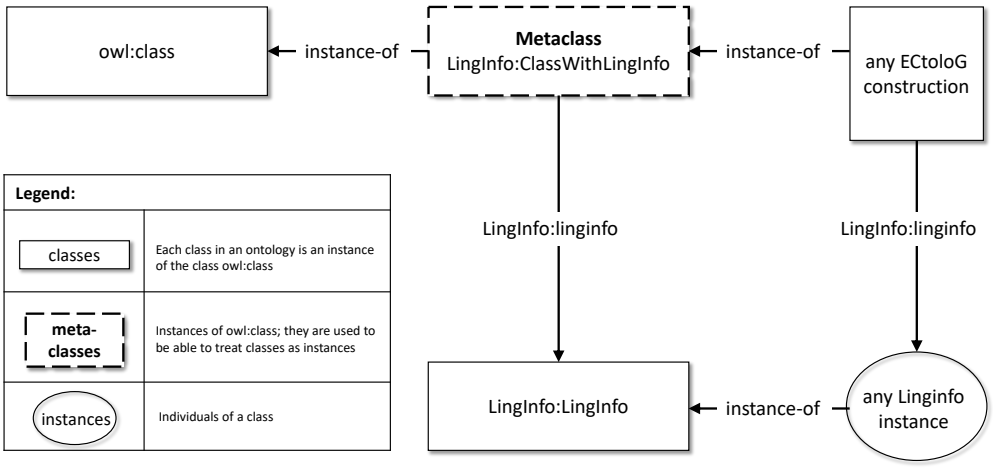


Figure 4.25: The LingInfo structure in the ECtoloG.

To embed the Linginfo model into the ontological modeling framework

[17] See further https://www.w3.org/TR/owl-ref/; last checked May 22, 2022

described in Section 4.1, `LingInfo:LingInfo` is made a subclass of the class `inf:linguistic-object`. Now, each class that should have linguistic information assigned to it, has to be an instance of the meta-class `LingInfo:ClassWithLingInfo`.
In a next step, an instance of `LingInfo:LingInfo` has to be defined containing the item that is linked to an instance of the class `inf:word` through the property `LingInfo:MorphosyntacticDecomposition`.
All children of the class `LingInfo:LingInfo` also inherit the property `LingInfo:lang`. This allows that the language ID can be assigned to each instance of the class `LingInfo:LingInfo`.

As a next step, two subclasses of the class `inf:word` have to be created: `LingInfo:Stem` and `LingInfo:InflectedWordForm`.
The class `inf:word` offers the following relevant properties: `LingInfo:case`, `LingInfo:gender`, `LingInfo:partofspeech`, and `LingInfo:number`. Additionally to the complete structure of the LingInfo model as it is embedded into the ontological modeling framework, the symbols that can fill the range of those mentioned properties are listed in Figure 4.26.[18]
The class `LingInfo:InflectedWordForm` is the domain of the property `LingInfo:inflection` whose range can in turn be filled with instances of the class `LingInfo:Affix`. That class again has two subclasses, one containing derivational affixes and another one subsuming inflectional affixes. `LingInfo:Affix` is a subclass of `inf:morpheme` which again is a subclass of `inf:linguistic-object`.

The class `inf:morpheme` is also connected to the class `inf:word` via the property `LingInfo:root`. `LingInfo:root` connects instances of `inf:word` to instances of the class `LingInfo:Root`, a subclass of `inf:morpheme`. Additionally, the class `inf:word` has a subclass `LingInfo:Stem` which is connected to it via the property `LingInfo:isComposedOf`. This property is useful and necessary in cases where the stem is a complex one being

[18] The used namespaces make explicit when a class of the ontological modeling framework is used.

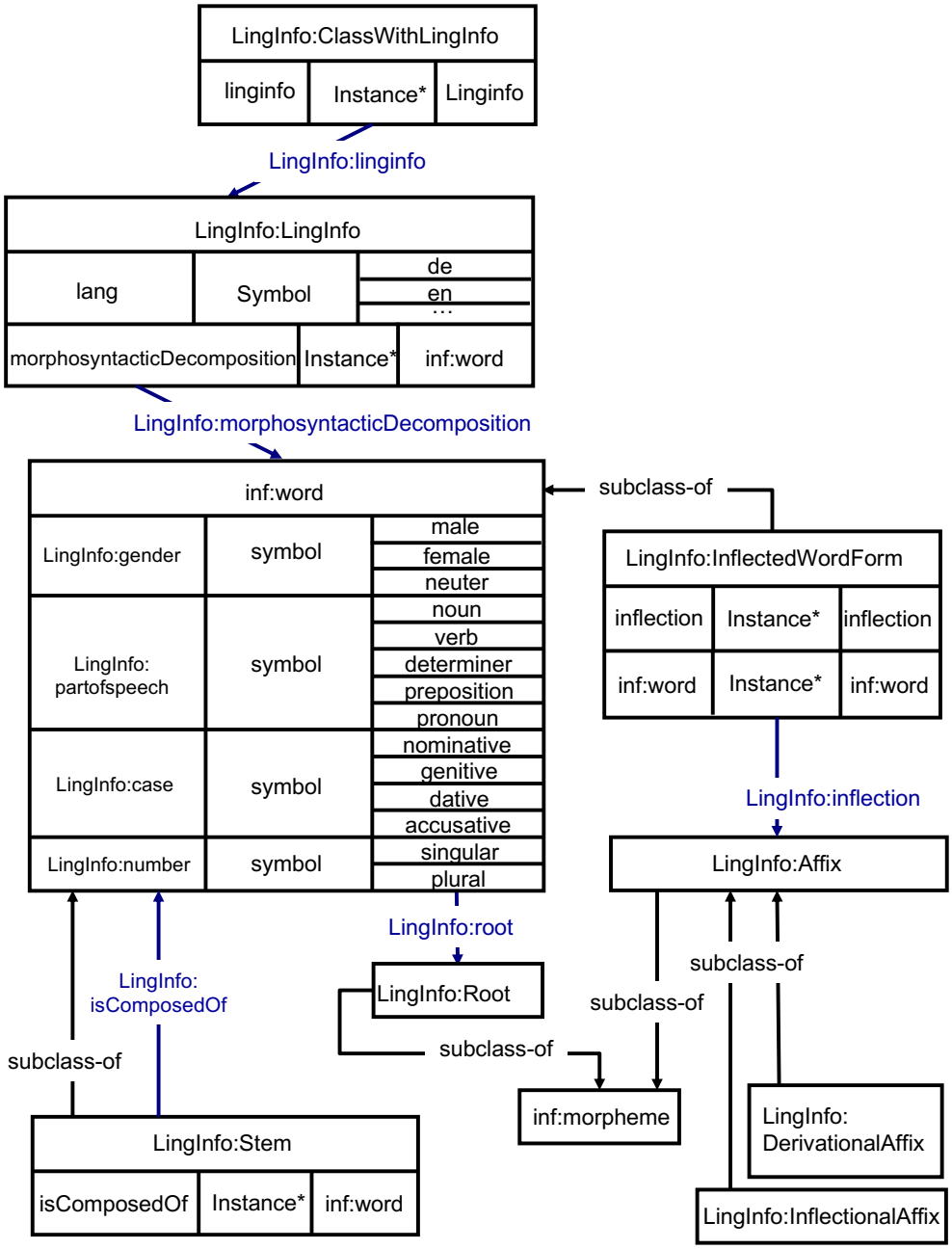


Figure 4.26: Classes and their properties to model linguistic information.

composed of various lexemes as e.g. in the term *Fußballspieler*, where the stem of the word is composed of the terms *Fußball* and *Spieler*.

The design of linguistic information as described in the previous para-

graphs entails that for each lexical construction a corresponding instance of the class `LingInfo:LingInfo` needs to be modeled. Therefore, one `LingInfo:LingInfo` instance for each lexical item in the example sentence is defined.
Each of the lexical constructions for the lexical items representing the example sentence is connected via the `LingInfo:linginfo` property to their respective instance of `LingInfo:LingInfo`. This feature is given since all classes of constructions are additionally to being subclasses of the class `ECtoloG:lexicalConstruction` also instances of the meta-class `LingInfo:ClassWithLingInfo`.

As the structure of modeling the linguistic information of lexical items is pretty straightforward, the most complex one – which is the inflected verb `liegt` – has been picked from the example sentence and will serve as an example.

The class for the construction `ECtoloG:liegtConstruction` is linked to the `LingInfo:Linginfo` instance `LingInfo:liegt` with the help of the property `LingInfo:linginfo`. Figure 4.27 displays that structure.

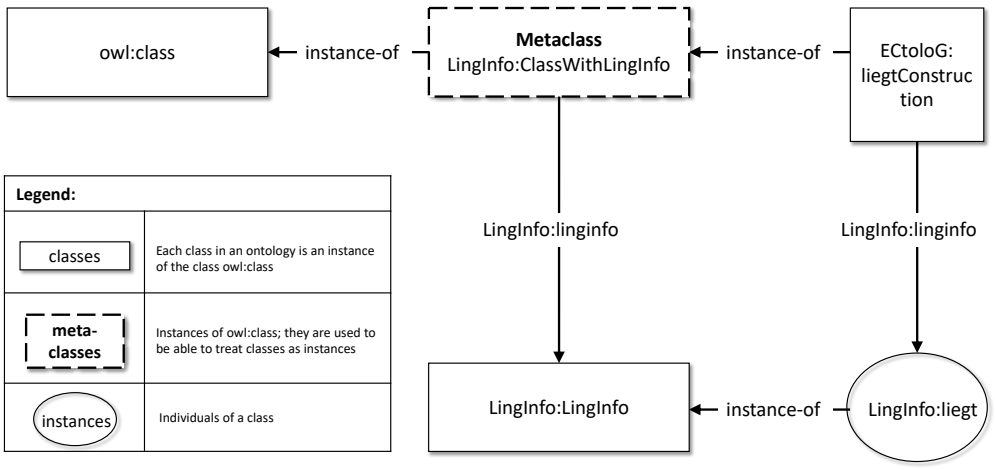


Figure 4.27: Modeling linguistic information for the instance `ECtoloG:liegt`.

Figure 4.28 presents the complete linguistic information that is modeled

for the `LingInfo:liegt` instance in the ECtoloG besides giving some brief explanation, similar to the model presented in Figure 4.26.

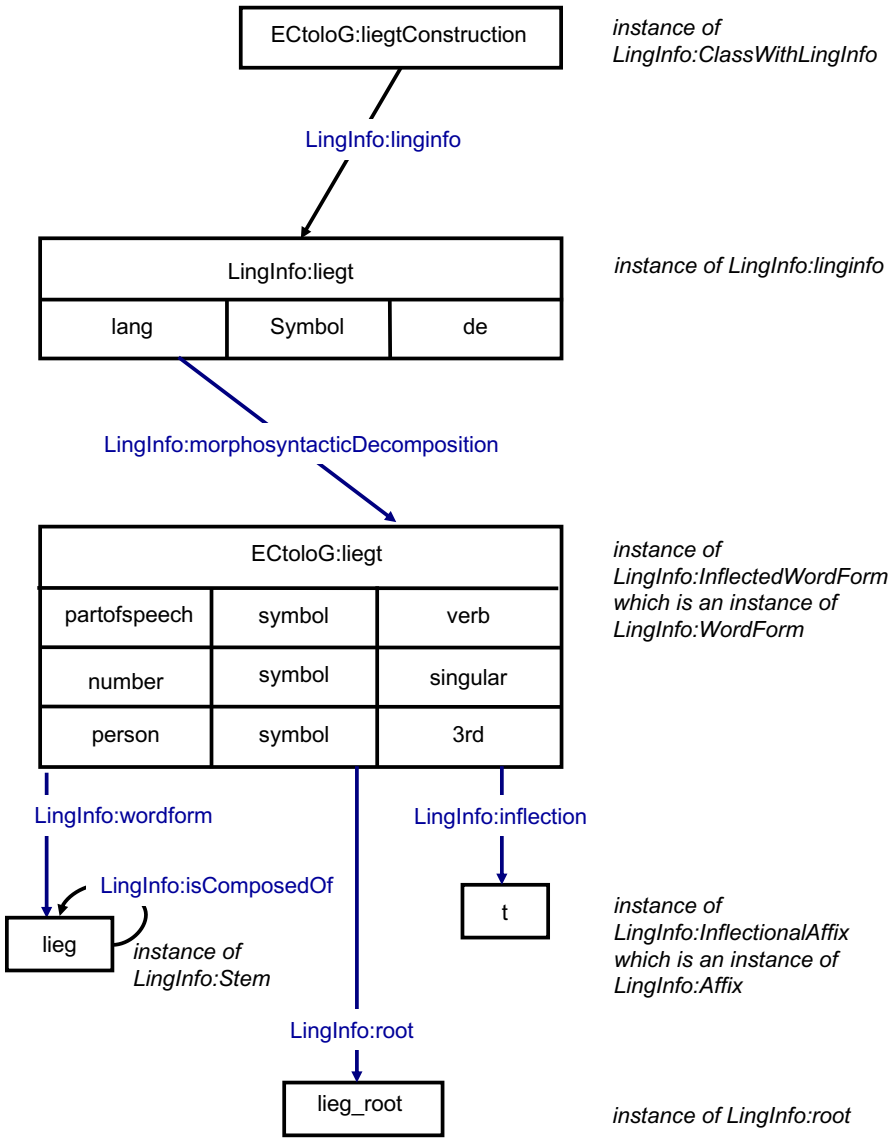


Figure 4.28: Example LingInfo structure for the instance `ECtoloG:liegt`.

The described way of representing linguistic information of lexical construction does not only present a model that can be reused in other ontologies where linguistic information is to be assigned to ontology classes but also enhances the grammar formalism that is proposed in this work as

the LingInfo model offers a way of accounting for morphological features of constructions. It can, for instance, provide a system with information on the composition of complex stems, inflectional or derivational affixation of a term or of its root.

The following section describes how the meaning pole of constructions is engineered and presents an ontological module containing basic image schemas.

4.6 Schematic Meaning in Constructions

This section describes how constructional meaning is represented in the ECtoloG. It first presents a basic hierarchy of image schemas. However, the ontological framework does not only include image schemas as known from cognitive science (see also Section 2.4) but also other frame-based knowledge, resulting in a dense network of connected schematic and role-based structures.

As already mentioned in previous chapters of this work, the meaning of constructions can be contributed by schemas which constitute parameter-based conceptual semantic feature structures. Section 3.6.1 briefly presented how other computational construction grammars deal with semantic knowledge in their formalisms. The way how semantics is represented in the ECtoloG relates strongly to how it is dealt with traditionally in ECG. FCG being mainly agnostic in which semantic representation the engineer chooses in the implementation, however, also has a way of presenting frame-based knowledge, but this is mainly secondary choice by FCG grammar engineers and the grammars that include frame-based semantics are essentially proofs of concepts. [Micelli et al., 2009] describes such a proof of concept and discusses its major issues. We refer the reader go Section 2.4 for a brief definition of frames and different kinds of schemas, which are integrated into an ontology, as described in detail in the following paragraphs.

Where to integrate constructional meaning in the ECtoloG First of all, it has to be determined, where exactly the image schema hierarchy is going to be located within the ontological framework. To enable being employed in the meaning pole of constructions, the class of `ims:image_schema` is modeled as a subclass of the class `ECtoloG:schema` in turn a subclass of `edns:description` (see Figure 4.29).

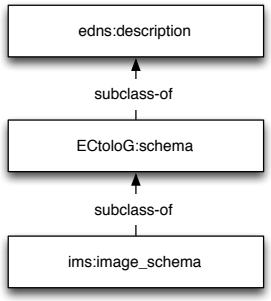


Figure 4.29: Location of the image schema hierarchy in the ontological framework.

Remember that a constraint has been imposed on what actually can fill the meaning pole of an `ECtoloG:construction` – repeated below for the readers convenience – stating that at least one of the values of the `edns:expresses` property is of type `ECtoloG:schema`:

- ∃ `edns:expresses` some `ECtoloG:schema`

As a result, anything that is located below the class `ims:image_schema` can provide a meaning for an `ECtoloG:construction`.

The Image Schema Ontology First of all, those schemas that are consistently repeated and intensely discussed in the traditional cognitive linguistics literature are collected, to define an ontological model containing the most important image schemas needed for language understanding (see Figure 4.30).[19] The is-a relation holds between the classes and denotes that a certain schema is a subclass of another one.

[19]Please note, that we do not claim that this collection of image schemas is in any way complete or carved in stone. It rather presents a suggestion to be elaborated further depending on specific needs of an application scenario.

Figure 4.30: A suggested hierarchy of the most important image schemas.

Primarily, the findings are based on schemas mainly described by Lakoff, Johnson or both [Lakoff and Johnson, 1980, Johnson, 1987]. The motion schemas that have been added to this basic hierarchy have been suggested by Mandler [Mandler, 1992]. Some of the natural language examples are taken from [Loos et al., 2004].

Five categories of image schemas have been defined, each category containing in turn further schemas:

Category 1: Spatial image schemas `ims:space_schemas`:

- `ims:verticality_schema` [Johnson, 1987]: A verticality schema is an image schema that involves up and down relations as, for instance, standing upright, climbing stairs [Loos et al., 2004] or watching water rise in a pool.

- `ims:center-periphery_schema` [Lakoff, 1987, Johnson, 1987]: This is an image schema that involves a physical or metaphorical core and edge, and degrees of distance from the core. Examples are an individual's social sphere, with family and friends at the core and others having degrees of peripherality ([Loos et al., 2004]).

- `ims:up-down_schema` [Lakoff, 1987]: An up-down schema is a spatial image schema that is used with a respect to a vertical axis. Natural language examples are - mostly metaphorical - as in "sad is down" or "happy is up" ([Lakoff and Turner, 1989, p.275]).

- `ims:front-back_schema` [Lakoff, 1987]: A spatial image schema denoting the location or an object with respect to another one.

- `ims:near-far_schema` [Johnson, 1987]: Given a center and a periphery, the near far schema is experienced as stretching along our perceptual or conceptual perspective [Lakoff and Turner, 1989, p.125]

- `ims:contact_schema` [Johnson, 1987]

- `ims:straight_schema` [Cienki, 1997]

- `ims:left-right_schema` [Clausner and Croft, 1999]

Category 2: Locomotion image schemas `ims:locomotion_schemas`:

- `ims:locomotion_schema` [Dodge and Lakoff, 2005, p.72-84]: The locomotion schema includes various roles as gait, speed, effort and body parts. Therefore, different kinds of locomotion can be distinguished from each other by for instance their speed. For instance, crawling or run are two motion types that involve a locomotion schema and distinguish each other by speed and by the body parts that are involved in the movement.

Category 3: Motion image schemas `ims:motion_schemas`: Motion schemas as defined by [Mandler, 1992] involve a kind of animacy whereby animate objects are able to fulfill a movement on their own while inanimate objects need an external force to move.

- `ims:inanimate-motion_schema` [Mandler, 1992]: An inanimate motion schema involves a kind of force applied to the object that fulfills the motion. A natural example is the verb *push*.

- `ims:animate-motion_schema` [Mandler, 1992]: An animate motion schema is an image schema that is active when animate objects fulfill motions on their own, like *walking*.

- `ims:self-motion_schema` [Mandler, 1992]: Mandler defines the image schema of self motion as a kind of animate motion schema where an object fulfills a movement without an external force.

- `ims:caused-motion_schema` [Mandler, 1992]: A caused motion image schema is a kind of inanimate motion schema that involves the external force to be set in motion.

Category 4: Containment image schemas `ims:containment_schemas`: Containment schemas are image schemas that are composed of the following physical or metaphorical entities:

- a boundary
- an enclosed area or a volume
- an excluded area or an excluded volume [Lakoff, 1987, Johnson, 1987].

The following image schemas belong to the category of containment image schemas:

- `ims:container_schema` [Lakoff, 1987, Johnson, 1987]
- `ims:surface_schema` [Johnson, 1987]
- `ims:full-empty_schema` [Johnson, 1987]

Category 5: Force image schemas `ims:force_schemas`: [Johnson, 1987, 42-44] Force schemas are image schemas that involve causal interaction involving the following entities:

- source and target
- direction and intensity
- path of motion of both the source and the target
- a sequence of causation [Loos et al., 2004].
- `ims:force_schema` [Lakoff, 1987]: A natural language example for the force schema is *wind* or *gravity*. A metaphorically example is *love as a physical force*.
- `ims:attraction_schema` [Lakoff, 1987, Johnson, 1987]: An attraction schema is active when an object pulls another object towards itself by asserting either a physical or a metaphorical force on another object. Examples are magnetism, gravity, or romance [Loos et al., 2004].

- `ims:balance_schema` [Lakoff, 1987, Johnson, 1987]: A balance schema involves two counteracting forces (both physical or metaphorical). An example is a temperature.
- `ims:blockage_schema` [Lakoff, 1987, Johnson, 1987]: A blockage schema is a force image schema that involves a force that is blocked by an obstacle.
- `ims:compulsion_schema` [Lakoff, 1987, Johnson, 1987]: A compulsion schema is a force image schema that included an external force pushing an object (metaphorically or physically). An example is the experience of a person being pushed by strong wind.
- `ims:counterforce_schema` [Lakoff, 1987, Johnson, 1987]: A counterforce schema involves at least two opposing forces that meet either physically or metaphorically. An example is a soccer game where two opposing teams fight against each other.
- `ims:diversion_schema` [Lakoff, 1987, Johnson, 1987]: A diversion schema is a force schema that involves a force that effects a redirection, e.g. being directed off course by a strong current while paddling.
- `ims:enablement_schema` [Lakoff, 1987, Johnson, 1987]: An enablement image schema is a force schema that involves having the power to perform an action.
- `ims:restraint-removal_schema` [Lakoff, 1987, Johnson, 1987]: A restraint-removal schema is a schema that involves the removal of a barrier to an action of a force.

Interschematic Relations Having determined which schemas build the foundation of the image schema ontology, the relations that hold in-between the schemas are to be defined. Due to the fact that the image schemas are embedded in an ontological framework, those relations can go beyond simple inheritance and can result in a semantically dense network.

Literature is rather scarce on that topic, which means that the relations are defined according to our understanding on the schemas' usage and function.
Following the assumption made in Embodied Construction Grammar and being grounded in Cognitive Grammar [Langacker, 1987] and Frame Semantics [Fillmore, 1982], schemas can be evoked by or can evoke other schemas, i.e. particular schematic roles of another schema can be imported into and thereby made accessible to another schema. A schema can, therefore, be defined against the background of another schema (see Section 3.2.4 on the `evokes` operator). This evokes relation can find its place within the ECtoloG: The property `ECtoloG:evokes` and its inverse counterpart property `ECtoloG:evoked-by` are defined as subproperties of the `edns:generically-dependent-on` property and its inverse property `edns:generic-dependent`, respectively (see Figure 4.31).

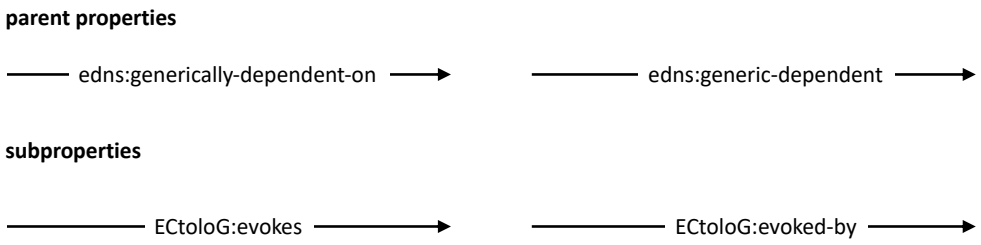


Figure 4.31: Definition of the `ECtoloG:evokes` and its inverse counterpart property `ECtoloG:evoked-by` in the ontological framework.

Generic dependence is a relation that has already been modeled in the foundational ontology and has been defined as the dependence on an individual of a given type at some time. In our understanding, this relation models the possibility of creating copies of information objects that take place or exist some place or some time. We suggest that there is a generic dependence between schemas that evoke other schemas or between schemas that are evoked by other schemas, respectively. Therefore, the previously mentioned properties `edns:generically-dependent-on` and `edns:generic-dependent` are determined as parent properties of the properties `ECtoloG:evokes` and its corresponding reverse property

`ECtoloG:evoked-by`.

Schematic Roles As a first approach, schema roles were modeled as instances or classes, depending on the schemas' nature. It seemed obvious to us that roles should be treated as ontology classes or instances, respectively. However, a vast amount of problems arose thereby which lead to a different approach solving those issues: Schematic roles are modeled as properties in the ECtoloG, instead. The first approach is shortly sketched out in the subsequent paragraph, pointing out the main problems arising by having taken the firstly mentioned modeling approach. Afterwards, the second approach is described in the subsequent paragraphs.

Modeling Schematic Roles as Instances or Classes In our first approach of modeling schematic roles an ontology class `ims:schematic-role` is defined as a subclass of the `edns:concept` class, as depicted in Figure 4.32.

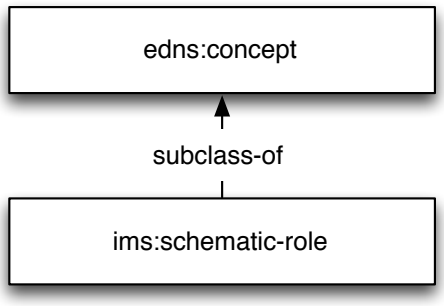


Figure 4.32: Definition of the class `ims:schematic-role` in the ontological framework.

According to the specification of the DnS ontology, a concept is classified as a non-physical object which again is defined by a description. Its function is classifying entities from a ground ontology in order to build situations that can satisfy the description. Schematic roles are parameters that allow other schemas or constructions to refer to the schema's key variable features. For instance, the role of a trajector in a trajector-

landmark schema can be played by the same entity that denotes the mover in a caused motion schema.
In the image schema ontology, schematic roles are modeled with the help of the `edns:defines` property, at first. This means that a schema defines its schematic roles by this property, as graphically displayed in Figure 4.33.

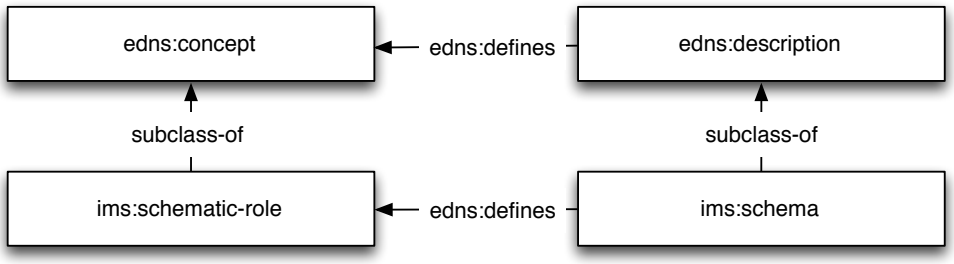


Figure 4.33: Being a subclass of `edns:description`, the class of `ims:schema` inherits the `edns:defines` property. The class `edns:schematic-role` can fill the property's range.

The following restriction on the class `ims:schematic-roles` is defined:

- ∃ `edns:defined-by` some `ims:schema`

It determines that at least one of the values of the `edns:defined-by` property is of type `ims:schema`. The domain of the `edns:defines` property is `edns:description`. As the class `ims:schema` is a subclass of `edns:description`, it can fill the property's domain. Its range, however, is set to either `edns:concepts` or `edns:figures`. As previously mentioned, the parent class for schematic roles is `edns:concepts`, which automatically makes it a possible candidate for the property's range. The cardinal problem occurring hereby is that it is not possible to fill the roles by entire ontology classes. This, however, is necessary in a lot of cases, since the parameters of a schema do often not refer to atomic values but possibly to whole classes of entities. An example is the `Source-Path-Goal` schema as depicted in Section 2.4.1, where the source or the goal can be a landmark that in turn can be any kind of referent. Here we would need to

use a class denoting spatial referents as the range for the `edns:defines` property – not a single instance of that class. This problem is not to be overcome in another way than to model schematic roles as properties, instead.

Modeling Schematic Roles as Properties Each role of each single image schema is represented by a certain property. The domain of the respective property is set to the corresponding schema class, while its range is set to the corresponding class whose subclasses and instances can possibly fill its range (see Figure 4.34).

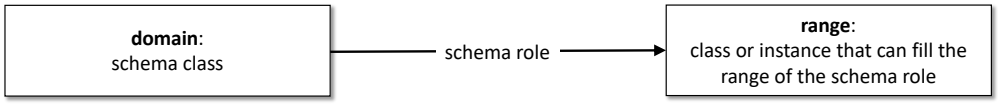


Figure 4.34: Schematic roles as properties with a schema class as domain and a class whose subclasses and instances can fill its range.

This way, the problem mentioned in the previous paragraph that roles cannot be filled by ontology classes is overcome: It is now possible to fill roles of schemas either with complete classes or with single instances of classes.
The following section goes into detail of which FrameNet frames have been integrated into the ECtoloG.

4.7 Schemas: Frames

A detailed definition of Frames has already been given in Section 2.4.2. They are the building blocks of FrameNet, an online database containing more than ten thousands of English lexical units, annotated with their corresponding semantic frames. The following describes how the Kicktionary has been integrated into the ECtoloG (see Section 2.4.2 for a brief description of the Kicktionary and a brief motivation for its usage in this work).

Integration of the Kicktionary As a first step, it has to be investigated if frames, scenarios or even both comply with schemas as they are defined in this work. Scenarios, as defined in the Kicktionary, correspond to events taking place during a soccer game. A scenario can be described as a schema, listing all the roles that are later inherited by the frames which are subordinates of the respective scenario.
A concrete example is the Scenario `Chance` listing the following roles:

- GOAL, TEAM, OPPORTUNITY, PLAYER, SOURCE, SHOT

The Frame `Create_Chance` needs the following roles:

- GOAL, TEAM, PLAYER

Unfortunately, there is no is-a relation between Scenarios and Frames. This means that the Kicktionary's hierarchy of Scenarios and Frames cannot automatically be integrated into the ECtoloG. Instead, the integration has to be conducted manually, thereby carefully deciding on the relations that hold between them.
For now, it is assumed, that either a is-a relation or a evokes relation holds between Scenarios and Frames. However, future work might include a more thorough investigation if this issue is true for all Scenarios and Frames, also for all of those encoded in the FrameNet database, and to find out a way how to automatically map the roles between constructions and frames, scenarios and schemas. This is, however, not in the scope of this work.

4.8 Sum Up: What Have We Gained So Far?

In the previous chapter, we described the concrete engineering of the ECtoloG. We showed exactly how simple and complex constructions and schemas are modeled and which external knowledge bases can be included and how. At this point, we would like to pause for a moment and sum

up which benefits and which advantages might so far be won with the modeling of a constructional grammar within an ontological framework.

At this point, the benefits of the resulting ontological framework can be summed up as follows:

- **Formal Construction Grammar:** The ontological framework includes construction grammar's major components, which are constructions on different levels of abstraction, i.e. complex and simple ones. It offers those constructions a solid, well-grounded foundation into which they can be integrated and benefit from its format.

- **Semantically Well-Defined Relations:** Constructions in the ECtoloG can now be connected with semantically rich relations going beyond inheritance relations as the ontological framework offers a well-defined, dense semantic network.

- **Image Schema Ontology:** The ontological framework includes a basic image schema ontology, where most important image schemas are collected and semantically related to each other. Those semantically well-defined relations come with the ontology they are integrated into. Additionally, the image schema ontology offers an extendable base that could serve other scientists as a base.

- **Focus on Semantics** To our knowledge, the ECtoloG is far more fine-grained, well-defined and semantically dense than the ontologies used currently in other Construction Grammar formalisms.

- **Applicability:** There is no need for a dedicated editor for either modeling the grammar or visualizing it as state of the art ontology editors as, for instance, protégé can be used.

- **Reusability:** Its standardized format guarantees reusability of the model. In addition, it allows the reuse of already existing modules as we previously saw, like the Kicktionary or the LingInfo model.

- **Extensibility:** The ECtoloG offers an interface where further Frames, Scenarios, and of course constructions can be integrated. Domain ontologies or other modules to cover certain phenomena that are needed can be integrated in a straightforward way given they are based on the same foundational ontology. In addition, it allows to reuse existing ontology learning models to automatically extend and populate the ontology.

- **Relevance:** The undertaking is relevant in theoretical and computational linguistic. We aimed at describing everything in a comprehensive way so it can be used, reconstructed or extended by the interested research community.

- **Coverage:** Currently, the grammar contains approximately 150 classes and 500 instances. To extend the lexicon, semi-automatic methods might be of most value as proposed in the next chapter.[20]

- **Representation:** Its standardized, state-of-the-art ontological format format makes the ECtoloG comparable to other ontologies based on the same foundational ontological framework.

- **Editability:** The amount of expertise to edit the grammar is considered reasonable. The user group might be extended by using a known and standard ontological format from standards gramamr engineers to ontology experts and engineers. Existing editors offer a vast amount of accessible learning materials that allow quick onboarding.

- **New Format that Opens New Possibilities:** The ECtoloG offers a new framework of formalized construction grammar based on ontologies to be experimented with.

Of course, the model does not come without flaws. As every grammar and ontology engineering effort, it does include a huge amount of manual

[20]You can find the ECtoloG and a readme file including exact numbers on the accompanying USB stick.

work. However, future work might include the incorporation of various machine learning mechanisms. This issue will further be discussed in the final Chapter 6.

The following chapter focuses on an example application of the engineered ontological model and describes the different steps taken in that process.

Chapter 5

Application and Population

This chapter describes an example how the ECtoloG could be utilized in an NLP system that uses ontology-based knowledge sources. The main idea pursued in this example is the following:

Given a domain-specific ontology – incorporated into the ECtoloG – which models processes and objects that are particular for that specific domain, natural language processing and partial natural language understanding is performed with the help of a constructional analyzer, using the information from the ECtoloG as its knowledge source.
Additional tools are described that provide a representation of the processed sentences in RDF triples at the end of the complete application flow. The last section of this chapter presents a way how to automatically populate the ECtoloG with instances to increase its coverage.

5.1 Ontological Levels

Briefly recapitulated, the ECtoloG is assumed to present an ontological framework – together with the foundational ontology it is based on and with all additional modules that have been integrated, as for instance the image schema hierarchy or the LingInfo model, as presented in detail in the Sections 4.6ff. of the previous chapter. The main benefits have just

been summarized, one of them being its extensibility: A domain-specific ontology can be plugged into this framework and represent the knowledge source in form of constructions and schemas that might be needed for processing example sentences from that specific domain. As important as the constructional and schematic format are the well-defined relations between those constructs. The domain ontology has to additionally meet the requirement that its format is compatible with the ECtoloG's format. As the ECtoloG is, however, built in OWL, representing the state of the art in ontological modeling, this requirement is not expected to represent a problem.

To ensure that the ECtoloG is as simply extensible as possible, it has been divided into different levels, depending on the nature of the ontology classes. Each of those levels is supposed to be extensible in one way or another, especially the ECtoloG's coverage, however, has to be extensible so that it can potentially be used not only for domain-specific but also for domain-independent natural language processing.

The lowest level of the ECtoloG contains domain-specific constructions and schemas, representing objects and processes typical for that domain on a separate level, called domain layer. On top of this level, you find the domain-independent layer, containing domain-independent constructions and corresponding schemas. Followed by the foundational level. Figure 5.1 shows the layered design of the ECtoloG.

The **foundational layer** consists of the DOLCE ontology, and its extensions DnS and OIO (see Section 3.4. It constitutes the ontological foundation the ECtoloG has been built on. The **domain-independent layer** then contains both abstract constructions as, for instance, lexical ones and abstract schemas like the `ECtoloG:schema` definition. Those schemas and constructions are assumed to be that general that they occur in several different domains. They are still on a higher level of abstraction than concrete constructions and schemas contained in the domain layer, and include main parts of the image schema hierarchy as described in

Section 4.6. Then, there is the additional **domain layer** that includes domain-specific constructions and corresponding schemas. Examples for those instances are the `ECtoloG:FußballspielerConstruction` and the corresponding `ECtoloG:FußballspielerSchema`. Detailed examples for both domain-specific constructions and schemas have already been given in the Sections 4.2ff., where an example sentence from the soccer domain has been constructionally analyzed, first informally and then using constructions and schemas as formally represented in the ECtoloG.

The resulting ontological model presents a rich knowledge base with densely intertwined semantic relations holding between its classes and instances, which can possibly be used as a knowledge source for a parser, analyzing natural language. Given that we assume a construction grammatical approach to language processing, both form and meaning, we need a parser that takes into account both of these phenomena. As argued in the beginning of this work, the complete meaning of a sentence is not covered by the mere sum of its parts but complex constructions might add significantly to the meaning. Therefore, we have been searching for a constructional analyzer that fulfills standard parsing techniques, working out the grammatical structure of natural language sentences based on both their constructional and their schematic composition which yielded exactly two results: One that is included in the Fluid Construction Grammar framework and uses FCG grammars as input and one that uses ECG grammars as its input. There is no constructional analyzer yet that takes a grammar formalism in ontological format as its input.

As the implementation of such a constructional analyzer is not in the scope of this work and the ECtoloG is heavily based on Embodied Construction Grammar (and also because the output of the analyzer that is used in ECG is currently preferred by the author as it appears to be easier to be be grasped), it has been decided to use the secondly mentioned constructional analyzer. The following sections describe the various steps of an example analysis in detail.

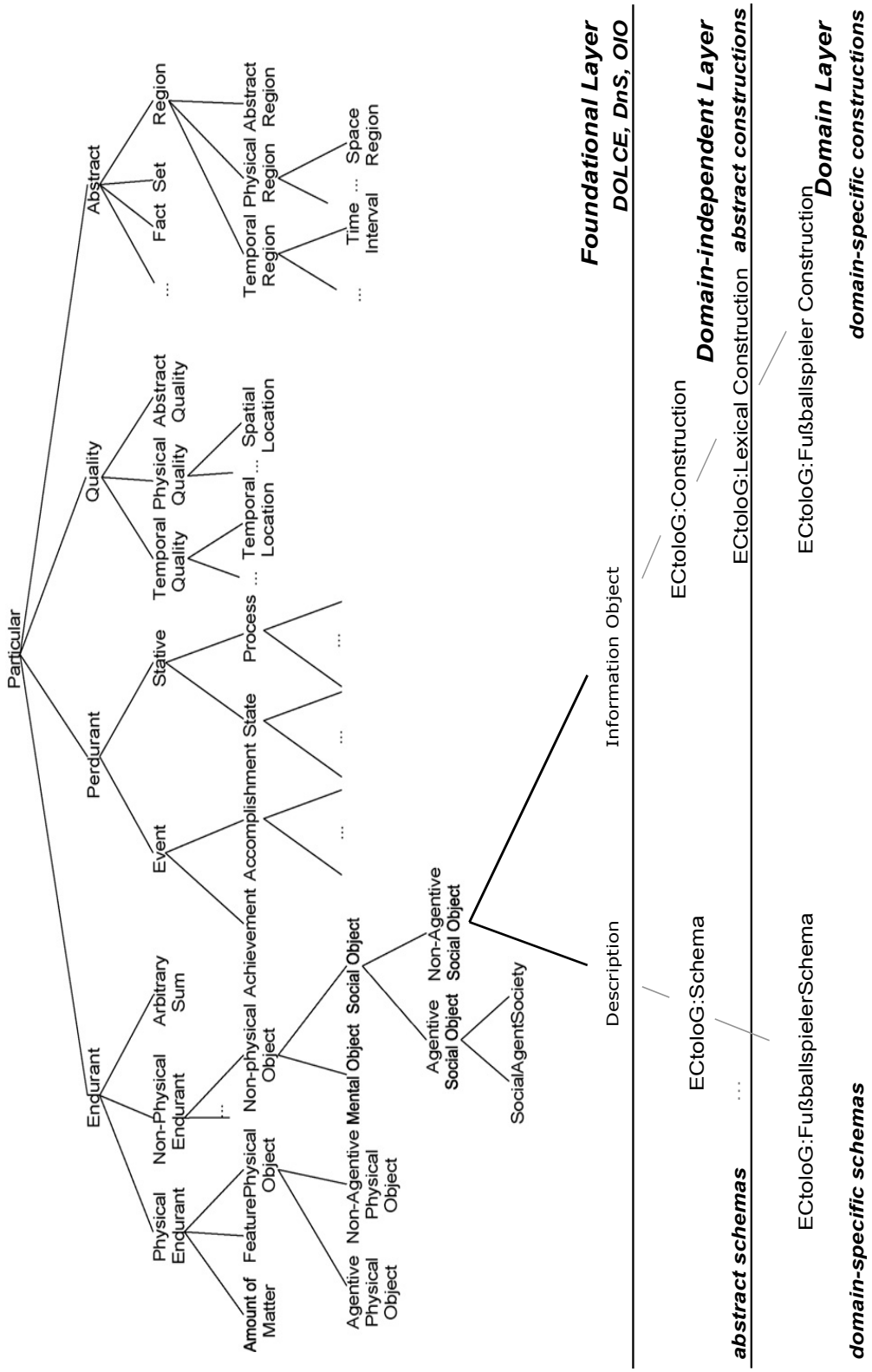


Figure 5.1: The different ontological levels in the ECtoloG.

5.2 Application Flow

This section describes the application flow, detailing each step of the complete processing cycle. It stays on a more abstract level in describing them, while later sections in this chapter detail a concrete application example (see Sections 5.3ff.).

Figure 5.2 shows the complete processing cycle, which is described in detail the following sections below.

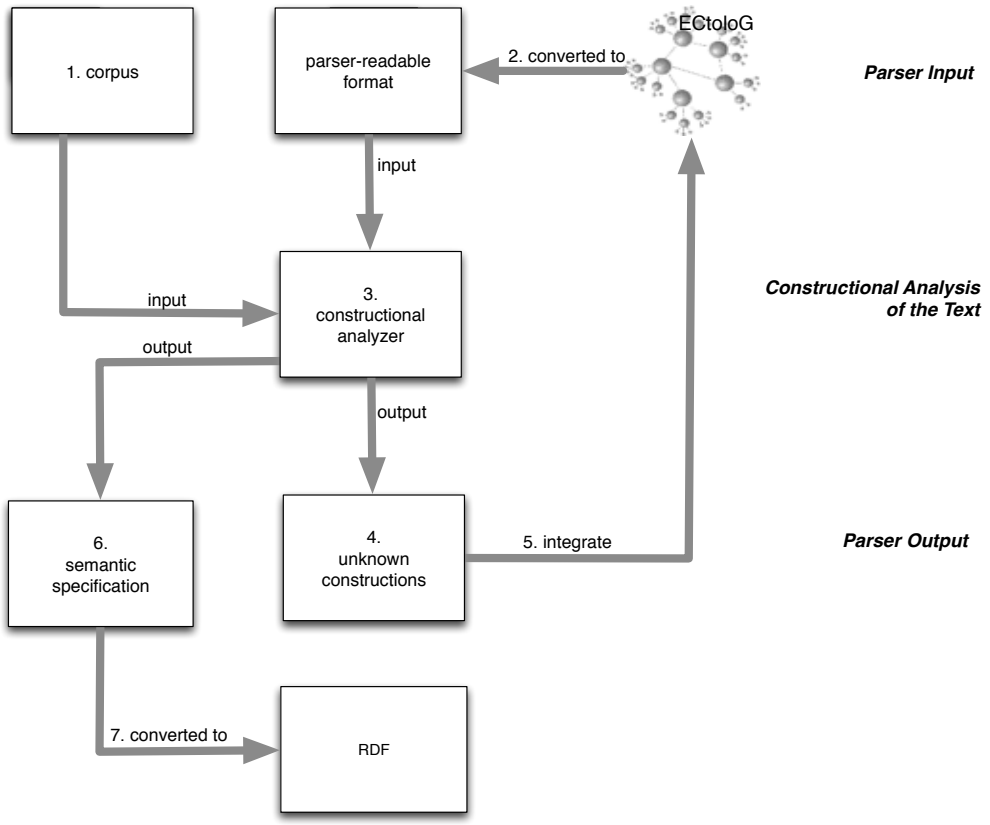


Figure 5.2: Example processing cycle on how to use the ECtoloG with the constructional analyzer.

Parser Input

1. **Corpus**: The corpus used in this work includes domain-specific nat-

ural language sentences. The chosen domain is the soccer domain and the corpus contains sentences from a news ticker from different soccer matches. Section 5.3.1 gives more details on the corpus, the language used therein, as well as on its creation.

2. **Format Adaptation**: **Conversion from ECtoloG format into parser-readable format:** The parser that is being used can constructionally analyze sentences. To perform the analysis, it needs a construction grammar as its input. The parser has been designed to utilize grammars in ECG format.[1] Now, there are two ways to overcome this incompatibility and to be able to use both the ECtoloG and the constructional analyzer: First, the analyzer has to be adapted to take grammars in ontological format as input, or, second, the ECtoloG needs to be converted into parser-processable format. In this work, the decision has been made to choose the second approach. The proposal of changing the format of the ECtoloG to be parser-readable is of course against many of the advantages and values the ECtoloG brings, especially when it comes to values like the state of the art in knowledge representation that increases efficiency when it comes to reusability or extensibility. However, it would go heavily beyond the scope of this work to now implement a constructional analyzer that can use the ECtoloG as its input grammar. Other tools like editors or reasoners can be used with the ECtoloG out of the box and these advantages are not to be neglected. Plus, an example analysis might motivate someone else to implement a constructional analyzer that takes ontologies as input in another research project.

Constructional Analysis of the Text

3. The constructional analyzer that is used for parsing in this work is described in [Bryant, 2004]. The parser is robust enough, to deal

[1] For more information, see the introduction to ECG and the syntax of some example constructions in Section 2.2.3.

with non-grammatical or incomplete sentences. The parsing process can be divided into two sub-processes: semantic chunking and semantic integration.

Similarly to the Abney parser [Abney, 1996], the constructional analyzer works on different levels. Constructions can belong to a specific level, depending on their level of abstraction. Lexical constructions, for instance, are level 0-constructions, while a relative clause-construction might be on level 4. The parser works bottom-up, and starts with level 0 constructions, i.e. with the lexical constructions. Then bigger chunks are analyzed, like noun phrases or prepositional phrases. The analysis is complete when there are no more units to be analyzed and no more constructions to be applied.

So-called construction recognizers do not only search for syntactic units in a sentence, but also check the semantics of those units once, a construction can be applied. The parser also performs backtracking and goes back to a point in analysis where it had several choices when coming to the end of an analysis without having parsed the complete sentence, yet. Then it tries another track until it finds a satisfying analysis. The final step in the analysis process is to check if the constraints in the feature structures are met. Parts of the analysis are stored in a chart.

In the final step – called *semantic integration* – those chunks that include the complete sentence are put together. Those analyses that include more semantic features are ranked higher then those where not all features and constraints are satisfied.

Parser Output

4. **Dealing with Missing Constructions:** In case the constructional analyzer discovers a construction it does not yet recognize during an analysis process, it returns a message, that the construction is not yet present in its grammar. This construction has to be added to its input grammar. This can happen automatically, by triggering an external ontology learning mechanism or manually (see next step).

5. **Ontology Extension**: In case the ontology has to be extended, the manual process, as described in detail in Chapter 4.4, is performed. Automatic or semi-automatic ontology learning mechanisms could be applied, as introduced, for example, in [Cimiano, 2006] or in [Litz, 2010].

6. **Semantic Specification**: In case the constructional analyzer does not find any unknown constructions in the analysis process, it outputs a so-called semantic specification (SemSpec) of the processed sentence. The semantic specification lists those schemas that are needed for understanding that sentence. Co-indices unambiguously mark identical entities in the sentence. The following section will present a semantic specification resulting from parsing an example sentence.

7. **Format Adaptation**: **Automatic Conversion into RDF:** To be more compliant with ontological formats and with the way how information is represented in the web, the SemSpec should be converted into an ontological format. It has been tested how to automatically convert it into RDF. How exactly that conversion process works will be explained in more detail in Section 5.3.5.

5.3 Concrete Application of the Processing Cycle

The focus of this section lies in describing a concrete example application of the workflow that has been briefly illustrated in the previous section.

5.3.1 Corpus Creation and Description

As previously mentioned, construction grammar's strength lies in how it represents both form and meaning of a linguistic unit. Also, in how it deals with real, occurring language, that uses metaphors, partial language, or new terms, invented on the spot. The corpus that is used in this work

contains data like that. That data has automatically been collected from the ARD live news ticker, containing comments on soccer games of the FIFA soccer world cup in 2006.[2]

The match report describing the match Italy against France and more concretely the example sentence in (3) will serve as the showcase.

(3) Perrotta liegt am Boden.
Perrotta is lying on the floor.

As mentioned, the news tickers contain comments on soccer matches. Each minute of a match there are one or two new messages which are posted on the web site. The sentences are full of information and relatively short.

Structured and semi-structured content that is embedded inside HTML documents can be extracted reasonably well, using increasingly automatic wrapper systems, as for instance presented in [Kushmerick et al., 1997] or [Simon, 2005]. Text wrappers, also called *agents*, can extract natural language data from web pages. Those texts can then be processed further, as described in the previous section. The text wrapper agent extracts text from a website when given a specific URL and stores that text into a text file. The data collected in the text file still needs to be cleaned. The cleaning process includes the removal of HTML leftovers, does minor substitutions of signs or new lines, and splits the text into one single sentence per line. The time tags of the events are kept, as those might become useful at some later point in time.

[2]Due to a change in law, the mentioned sites do not exist anymore and the news tickers are no longer available online. The law states that data like that has to be deleted after a year.

5.3.2 Conversion from ECtoloG into Parser-Readable Format

Table 5.1 shows both the number of constructions and schemas that are needed for the analysis of example sentence (3).[3] The respective ECtoloG containing all those constructions and schemas, as well as a list containing the converted versions of those constructions and schemas, is attached to this work and can be inspected on the accompanying USB stick.[4]

	Constructions	**Schemas**
total	23	23
domain independent	18	19
domain specific	5	4

Table 5.1: Amount of constructions and schemas needed to constructionally analyze the example sentence *Perrotta liegt am Boden*.

Constructions or schemas are counted as being domain independent when it is likely that they occur in other domains as often as they occur in this one. Else, they are domain-specific, as is, for instance the name of a soccer player, like *Perrotta*.

5.3.3 Constructional Analyzer: Parsing of the Sentence

This subsection provides a short description of the parsing process of the example sentence. We refer to Bryant, 2004 for further details on the analyzer's architecture and parsing processes.

The constructional analyzer performs semantic analysis by means of a

[3] Note, that as in every work dealing with linguistic analyses, this analysis contains many subjective decisions. There are numerous ways to constructionally analyze that sentence, depending not only on how fine-grained the analysis might be, i.e on how small the smallest linguistic unit is, but also on how the semantics of the constructions are represented.

[4] Note that the ECtoloG contains many more constructions and schemas, however, the ones listed in the table are the ones that are needed for the constructional analysis of the example sentence and where we put the current attention on.

semantic chunker and a chart. It analyzes semantics and syntax in parallel. During the analysis constructions and schemas that represent the semantics of the sentence are retrieved and finally arranged in a so-called *semantic specification* (SemSpec) of the sentence. This semantic specification is constituted by a coindexed lattice of schema instances. In the next section the resulting SemSpec of the example sentence can be inspected.

Since no unknown constructions were discovered in the analysis of the example sentence, the next two steps of the application flow dealing with how to deal with unknown constructions can be skipped, and the reader is referred to Section 4.4 and the respective literature on automatic ontology extension.

5.3.4 Semantic Specification

The parser outputs a semantic specification for each processed sentence. This SemSpec is a graph and an example is displayed in Figure 5.3. It represents the resulting SemSpec for the example sentence in 3.

```
Ground1 of type Ground

SelfMotion1 of type SelfMotion
      protagonist: {Entity5}
      means: {Lie1}
      path: {TrajectorLandmark3}

TrajectorLandmark3 of type
TrajectorLandmark
      trajector: {Entity5}
      landmark: {Ground1}
      relation: {"am"}

Lie1 of type Lie
   path: {TrajectorLandmark3}
   executor: {Entity5}
   scene: {SelfMotion1}
   smscene: {SelfMotion1}

Entity5 of type Perrotta
   distribution: single
   givenness: namedEntity
   sex: male
   status: player
    name: SimonePerrotta
```

Figure 5.3: SemSpec for *Perrotta liegt am Boden.*

The semantic specification lists the schemas that are necessary for understanding that sentence. First comes the name of the instance then its type followed by the roles that are part of the schema. When a role is linked to another schema, its value is listed in brackets. When it is a symbol, it

follows the colon after the role name. Roles that denote the same entities are referring to each other by the co-indices following their schema name. Figure 5.3 displays five schemas that jointly determine the meaning of the sentence: a ground schema, a self-motion schema, a trajector-landmark schema, a lie-schema and an entity schema..

- `Ground1`: According to the specification, `Ground1` is an instance of the `ground schema`. This is indicated by saying that it is of type `Ground`.
- `SelfMotion1`: This schema is of type `SelfMotion`. This schema has three roles: `protagonist`, `means`, and `path`. The values of these roles are embraced in brackets.
- `TrajectorLandmark3`: The `TrajectorLandmark3` is an instance of the `TrajectorLandmark schema`. IT has three roles: `trajector`, `landmark`, and `relation`. The values of these roles is embraced in brackets.
- `Lie1`: This schema is of type `Lie`. It has four roles: `path`, `executor`, `scene`, `smscene`. The values of these roles is embraced in brackets.
- `Entity5`: This schema is of type `Perrotta`. The schema has five roles: `distribution`, `givenness`, `sex`, `status` and `name`. The values of these roles is embraced in brackets.

In a next step, the graph can be converted into ontological format. This way, it is made interpretable by tools taking ontologies as input. We propose RDF triples as output so that RDF validators can be used for verification. The following section describes the conversion in more detail.

5.3.5 Automatic Conversion into RDF

This section describes the automatic conversion of the semantic specification into RDF with the help of a rule-based conversion tool, that we have implemented in the context of a previous research project. The tool has

been named *2RDF tool and its source can be found on the accompanying USB stick.[5]

Using the rule-based *2RDF tool, RDF instances can automatically be generated based on a semantic specification in a process consisting of the following steps:

1. **Mapping:** The first step includes the mapping from the source structure elements to the target structure elements, i.e. from schema instances, or roles to corresponding ontology concepts:
   The transformation process starts by defining simple mappings to create RDF instances for each node. If the source tag set is identical to the ontology's naming of concepts and properties no mapping rules are required. The mapping-engine, then, derives the appropriate concepts and properties from the ontology.

   Otherwise, a tag can be mapped manually to an ontology concept or a property. Lists of instances can be converted if declared as a list. Lists may, however, violate the hierarchical concept/property order and have to be declared using a list rule *list* $<tag>$, where $< tag >$ indicates that all nodes beneath this list node will be interpreted as concept - fillers for the property defined by the parent node of the list tag. Additionally, there is a `maplist` rule, which has a concept tag as parent node. Here, the property has to be provided with the rule, the list node will be mapped to this property.

2. **Expansion:** Most ontologies model the world in a very fine-grained way. Oftentimes, more fine-grained than it is described on web pages. Ontology axioms, for example, may require to distinguish between roles and types, endurants or perdurants (see Section 2.3.4). Therefore, the expansion rules are defined in a way to cover such cases. Expansion rules are production rules that anchor on ontological concepts or properties. When these anchors occur during

[5]For a brief description of RDF, the *Resource Description Framework*, see Section 2.3.2.

the transformation, additional instances are potentially inserted at that place generating additional instances to conform to the target ontology axioms.

3. **Merging:** Redundant instances are merged.

The SmartWeb ontology[6] is used as a reference ontology for the *2RDF tool for the conversion of SemSpecs into RDF triples. Using a different ontology than the ECtoloG enables us to test the conversion of SemSpecs from different domains than just the soccer domain as the SmartWeb ontology combines various distinct domain ontologies, besides a domain ontology containing concepts from the language used in the soccer domain. Therefore, mapping rules for the SmartWeb's ontology concepts are required.

For converting the respective SemSpec in Figure 5.3 into RDF triples, initially, a list of mapping rules has to be created manually. The created list of rules for the example sentence in (3) looks as follows:

```
list SEMSPEC
insert sportevent:FootballPlayer[smartdolce:HAS-DEMOMINATION]
smartdolce:denomination[smartdolce:NAME]
insert sportevent:Field[smartdolce:HAS-DEMOMINATION]
smartdolce:denomination[smartdolce:NAME]
concept SelfMotion navigation:SelfTransportation
property protagonist smartsumo:hasAgent
property trajector smartsumo:hasTrajector
concept TrajectorLandmark navigation:SelfTransportation
concept Lie smartsumo:BodyPosition
concept executor smartsumo:hasTrajector
concept Ground sportevent:Field
```

[6]The SmartWeb ontology can be found at *http://www.smartweb-project.de/ontology.html*

```
concept Perrotta sportevent:FootballPlayer
property name smartdolce:HAS-DENOMINATION
```

These mapping rules map both the concept and property names that are used in the semantic specification to the names of the ontology concepts. Additionally, the property `smartdolce:HAS-DENOMINATION` is added as a football player has a name and that information had been modeled differently in the SmartWeb ontology.

To summarize, the processing cycle starts with the parser reading in the example sentence and the parser-readable grammar. The sentence is constructionally analyzed and outputs a SemSpec, containing the sentence's schematic meaning. The *2RDF tool reads in both the SemSpec and the list of rules and initiates the conversion process, from the semantic specification into RDF triples. Finally, the tool outputs structured, machine-interpretable RDF triples for the example sentence:

```
<rdf:RDF
<sportevent:FootballPlayer
  rdf:about="http://www.eml-d.de/ecg#Entity1">
<smartdolce:HAS-DENOMINATION>Perrotta</smartdolce:
  HAS-DENOMINATION>
</sportevent:FootballPlayer>

<smartsumo:BodyPosition rdf:about=
  "http://www.eml-d.de/ecg#Lie1"/>

<navigation:SelfTransportationrdf:about=
  "http://www.eml-d.de/ecg#TrajectorLandmark1">

<smartsumo:hasTrajector rdf:resource=
  "http://www.eml-d.de/ecg#Entity1"/>
</navigation:SelfTransportation>
</rdf:RDF>
```

If needed, the output RDF structure can be automatically validated with an RDF validator. Among other advantages, validated triples save data consumers from wrong or malformed input. In addition, the output shows what data form is considered valid. Thereby, the RDF triples are displayed as a directed graph. Figure 5.4 displays the directed graph, generated by the RDF validator found at *http://www.w3.org/RDF/Validator/*.

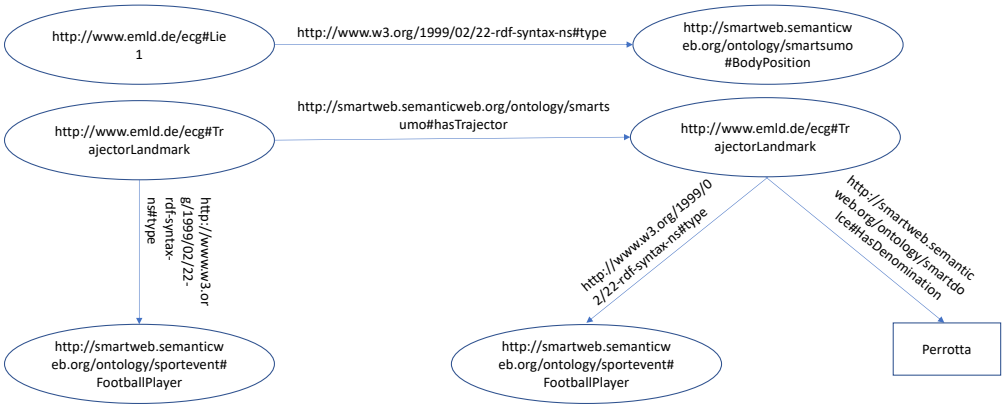


Figure 5.4: The RDF statements as a directed graph.

5.4 Semi-Automatic Population of the ECtoloG and the Integrated LingInfo Model

Populating an ontology is a tedious manual task. You've by now carefully followed the description of how to create constructions in the ECtoloG in Sections 4.4ff. There are open source tools that can help in automating certain steps to save time so we can work towards a semi-automatic way of ontology population. We'll be using the open source software package *Morphy* to morphologically analyze, synthesize, and determine the part-of-speech of terms for the German language automatically.[7]
The following sections describe the (semi-)automatic population both of the ECtoloG and of the integrated LingInfo model.

[7] See *https://www.wolfganglezius.de/morphy/, last checked May 27, 2022 and [Lezius, 2000]*.

5.4.1 Automatic Population of the ECtoloG

To increase the ECtoloG's coverage, it has to be populated with classes and instances, i.e. in our case with constructions. Parts of this process can be automated by using tools performing state-of-the-art linguistic analysis as mentioned before. The analysis process of the Morphy tool is described below. We have improved Morphy's usability a bit by building a simple user interface that helps generating sorted word lists from Morphy's output files more intuitively (see Figure 5.5). This UI allows to easily create lists of words sorted alphabetically and by their grammatical attributes like part of speech, grammatical gender, number, etc. We've selected attributes according to classes of constructions we would like to create in the ontology. Figure 5.5 shows all part of speeches including the respective attributes that can be used for sorting in the Morphy UI. You can find a detailed description on how to use it and the source code for the UI on the accompanying USB stick.

General Workflow To use the Morphy tool for the analysis of files, you can either use it online or it can be installed on a local machine. Morphy creates two files in its analysis process: one storing analyzed words (saved as *.lem), and one storing unknown ones (saved as *.unb). Morphy offers a user interface to enter the unknown words and store them for later analyses in its database. This way, you can analyze a complete file even if certain terms are missing in the data base. The analysis of each term of each sentence yields information about its stem, its part-of-speech, its case, its number, and its grammatical gender. The result of the analysis is the foundation that is now used to create sorted word lists. Now the UI Morphy Output Filter (MOF) comes into play. It reads in Morphy's output file (the one called *.lem) and generates exactly those word list that have been entered into the user interface.

The user is offered various options: Lists can be generated from a selected Morphy-Output-File (selectable in the upper left corner of the application's interface) for the following parts of speech: nouns, verbs, adjec-

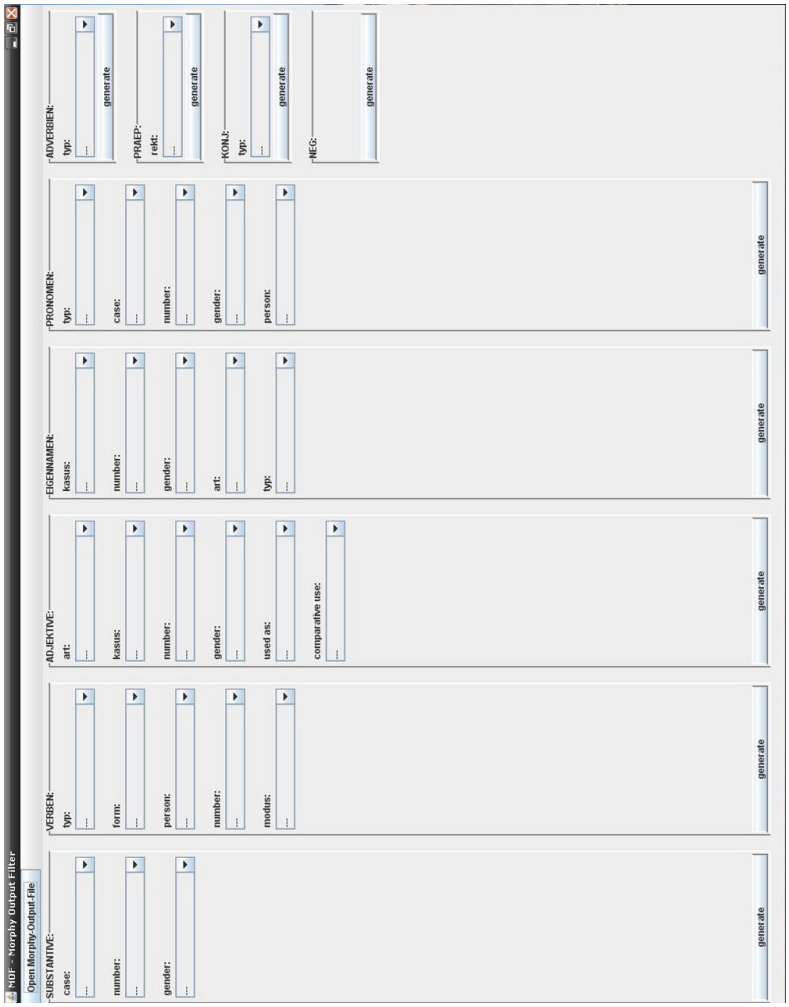


Figure 5.5: Screenshot displaying the user interface of the Morphy Output Filter.

tives, proper nouns, pronouns, adverbs, prepositions, conjunctions and negations. Then, various additional criteria can be selected for each part of speech: for instance, in the case of nouns the user can decide which case, number or gender the noun list should contain. The different selection criteria are all displayed in the screenshot.

Concrete Example As previously described in Section 5.3.1, natural language texts are extracted from specific websites and further processed automatically with the help of simple perl scripts, in a way that in the

end the plain texts are stored – one sentence per line – in a text file. Meta-information or empty rows are automatically removed by using these scripts that have been created for this purpose. The cleaned-up text files are afterwards analyzed by Morphy, yielding world lists sorted as described above that can be used as input for creating new, lexical constructions in the ECtoloG. Each term of a list is used as new instance of a lexical construction including the linguistic information that has been acquired in the analysis process.
To finish the creation of the form pole of each lexical construction in the ECtoloG, the LinInfo model needs to be populated with respective instances. The following sub-section describes exactly that.

5.4.2 Automatic Population of the LingInfo model

Similarly to what has been described above, the LingInfo model is populated with respective instances. All information that is required for a lexical item in the LingInfo, i.e. word class, gender, or language is retrieved from the output of the Morphy analysis. This means that all terms that have been retrieved in the analysis and are stored in a sorted word list get their LingInfo instances in the ontology.
A simple script allows for reading in the word list and creating one instance per word list entry for a certain parent class in the ontology that has to be entered manually. This way, the ontology engineer can decide how much coverage the lexicon in the ontology should have.

The last two sections described how lexical constructions can be automatically created with a focus on the form side of the construction only. This means that the lexical constructions will be created as instances of constructions embedded in the ECtoloG as described in the previous chapters. In addition, their LingInfo equivalents can also be automatically created so that all linguistic information that is needed for a lexical construction as described is represented in the ontology. What is missing is the link to its meaning pole, i.e. the respective schema of the construction. Currently,

this step has to be done completely manually. However, the foundation of image schemas that has been described in sections 4.6ff. can help embedding respective missing schemas and creating those meanings that are needed.
The following chapter summarizes the findings and gives an outlook on topics to build upon in future.

Chapter 6

Conclusions, Future Issues and Final Discussion

In this work, we have proposed a formalization of construction grammar by means of formal ontologies. The result of this undertaking is a powerful ontological model which is enriched with a cognitively motivated grammar layer that can be used in natural language applications – especially those making use of ontologies already.
We've elaborated on why the decision to create a new formalism makes sense and, in addition, why we have decided to base it on construction grammar and more specifically on Embodied Construction Grammar.
The following section summarizes the outcomes of implementing the proposed framework.

6.1 Outomes Modeling the ECtoloG

The following summarizes what has been described and achieved by now:

- Natural language processing systems like spoken dialogue systems, automatic translation systems, chatbots, or question answering systems are suddenly everywhere. Computing power and needed technology finally reached a point where the masses adopted them – be it in your homes, cars, or on your smartphones. However, natural

language processing and especially grammar engineering still constitutes a serious bottleneck in the development of applicable natural language processing systems. Speech systems, no matter how advanced, will always lag behind when it comes to understanding real spoken language. And we believe that there will always be manual, human input needed in order to cover language phenomena occurring in natural language interaction with a system. We consider proposals on how to automate certain tasks and offering robust enough frameworks including rich semantic foundations of high value.

- We have adopted a constructivist position of grammar representation in this work, meaning that a grammar should ideally include every layer of language, i.e. form and meaning or function as suggested in construction grammar theory as Construction Grammar has mainly been developed to handle phenomena that occur in natural language that other theories were not able to sufficiently handle (see Section 1.1).

- The progress towards formalizing constructions and towards a construction grammar architecture that is machine-processable raises a number of promising issues and challenging questions. And at this stage of research both in the field of theoretical and computational construction grammar, open issues leave lots of room for experimenting with or studying various ways of representing constructivists' ideas (see Section 3.2). This work attempted to fill in some of the open issues and delivered a formalization of construction grammar using the state-of-the art in knowledge representation and exemplifying it concretely with the help of one formalized example sentence.

- Chapter 3 detailed out, which existing resources absolutely make sense to be reused, potentially extended as in the case of the LingInfo model 3.5 and eventually integrated into an ontological foundation with a good reputation across academia.

- The tight integration into the foundational ontology of constructions of different complexity, be it lexical or grammatical ones, has been described in much detail in Sections 4ff.
- We've proposed example areas of application in Chapter 5 as an input source in a natural language processing system using a constructional analyzer and a potential processing cycle on how to use outcomes of the analysis further 5.3.
- Section 5.4 showed the automatic population and thereby the extension of the lexicon. This way, the ontology can contain quickly domain knowledge used to analyze texts from a certain domain.

Recapturing from Chapter 1, grammar engineering and therefore also construction grammar engineering faces a multitude of challenges. Figure 6.1 repeats the summary of major challenges.

Figure 6.1: Major challenges in grammar engineering and especially in engineering constructing grammars.

Let's have a look at how the previously described representation of an ontological construction grammar, i.e. of the ECtoloG, and the used engineering environment attempted to tackle these challenges summarized in the following:

- The proposed and described approach provides a concrete method of implementing a formalization of construction grammar based on ontologies. This method is well documented in order to make the ideas and analyses reusable, extending the framework with additional (domain) knowledge or even for writing new grammars.

- By providing a standardized grammar format which NLP tools that use ontologies can use, reusability for various NLP applications is guaranteed. Current research studies dealing with building ontologies to be used in NLP further pushed the idea of including a constructional grammar layer in such an ontology to find out if that is actually possible and eventually beneficial for NLP systems

- Concrete advantages coming with the ontological format are that they offer various new and semantically rich possibilities how constructions and schemas can be related to each other: The lattice that can be built both among constructions and among schemas within an ontology can be much more semantically fine-grained, as relations in an ontology go beyond simple inheritance relations, which basically come for free in an ontology. In addition, the format of the ECtoloG is compatible with other ontologies that are based on the same foundational model which automatically increases the compatibility and extensibility of the ECtoloG.

- Existing editors, like protégé facilitate accessibility, readability and extensions of ontologies in various ontological formats.

- The format which results from building the ontological grammar model is one of its major advantages. As previously discussed, grammar engineering is often done in a non-efficient way, resulting in models that are difficult to reuse and difficult to adapt to new application areas or even impossible to be used in different processing models. Since ontologies are used in a variety of applications and especially since natural language processing with ontologies is getting more and more popular, we believe that our model can be of benefit in various existing applications that can already handle ontological formats.

- Looking at the proposed processing cycle in Chapter 5, the semantics of analysed sentences based on the presented formalism are made machine-interpretable and usable for further use and processing. In

addition, analyzed sentences can automatically be converted into RDF triples, thereby their semantic representation is easily in compliance with common standards of knowledge representation.

- Embedding constructions and schemas into a foundational ontology provides a setting including semantics and a modelling basis that enables a comparison to other ontologies that can be used as reference points.
- The standardised format enables merging with other ontologies.
- Existing tools allow the automatic check of the ontology's consistency.
- Extending the grammar can be achieved by applying standard ontology learning mechanisms.

In addition, the ECtoloG provides added scientific value for the construction grammar community. It offers a formalization by means of formal ontologies which is:

- Extendable by external (linguistic) knowledge bases (as exemplified with the LingInfo ontology)
- Reusable, by adopting or extending the ideas and analyses for extending the presented framework with additional (domain) knowledge or even for writing further, even more elaborate grammars
- Reusable in NLP tools that require knowledge to be represented in ontological format
- Applicable to any domain that offers a domain-specific ontology and is compliant with the DOLCE ontology foundation

6.2 Outlook

Sentence formation *"...is a process of free creation;"* ([Chomsky, 2006], p.18).

As argued before, we believe that systems dealing with natural language input and especially with spoken natural language input need to be able to be robust enough to deal with the creative use of natural language. Speech systems, no matter how advanced these days, will always lag behind when it comes to understanding real spoken language. And there will always be human, manual input needed to cover real spoken languages. What has been proposed in this work is a formalism embedded in a foundational ontology allowing on the one hand to create further constructions to increase coverage in a well documented graphical user interface but also to use state-of-the-art ontology learning mechanisms to be used to do that automatically. By reusing and relying on standard ontological modelling rules, we tried to decrease complexity in creating new constructions.

We suggest the outcome of this work to be explored further in multiple ways. It is considered as a foundation that can be used to be embedded in an existing language processing system using ontologies based on the DOLCE framework. It can be explored further by the construction grammar community. We expect the dense semantic foundation to allow for novel deep semantic modelling of constructional meaning. Especially the frame-based meaning representation has in our opinion strong potential to make language understanding even better and increase adoption through an increase in user satisfaction. Because these kinds of approaches to language understanding might eventually allow to speak to the system in your language instead of adopting theirs.

Bibliography

[Abney, 1996] Abney, S. (1996). Partial parsing via finite-state cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, pages 8–15, Prague, Czech Republic.

[Alatrish et al., 2014] Alatrish, E., Tošić, D., and Milenkovic, N. (2014). Building ontologies for different natural languages. *Computer Science and Information Systems*, 11:623–644.

[Alexa et al., 2002] Alexa, M., Kreissig, B., Liepert, M., Reichenberger, K., L. Rostek, K. Rautmann, W. S.-S., and Stoye, S. (2002). The duden ontology: an integrated representation of lexical and ontological information. In *OntoLex Workshop at LREC*.

[Allen, 1983] Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11).

[Antoniou and van Harmelen, 2004] Antoniou, G. and van Harmelen, F. (2004). *A Semantic Web Primer*. The MIT Press.

[Arapura and Raja, 1990] Arapura, J. G. and Raja, K. K. (1990). Philosophical elements in vedic literature. In Coward, H. G. and Raja, K. K., editors, *The Philosophy of the Grammarians*. Princeton, NJ: Princeton University Press.

[Baclawski et al., 2002] Baclawski, K., Kokar, M. M., Waldinger, R. J., and Kogut, P. A. (2002). Consistency checking of semantic web ontologies. In *Proceedings of the First International Semantic Web Confer-*

ence on The Semantic Web, ISWC '02, pages 454–459, London, UK, UK. Springer-Verlag.

[Bahja et al., 2020] Bahja, M., Hammad, R., and Butt, G. (2020). A user-centric framework for educational chatbots design and development. In *HCI International 2020*, pages 32–43.

[Baker et al., 1998] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet P roject. In *Proceedings of COLING-ACL*, Montreal, Canada.

[Bateman, 2010] Bateman, J. A. (2010). Ontologies of language and language processing. In Poli, R., Healy, M., and Kameas, A., editors, *Theory and Applications of Ontology: Computer Applications*, pages 393–410. Springer, Dordrecht, Heidelberg, London and New York.

[Bateman et al., 1995] Bateman, J. A., Henschel, R., and Rinaldi, F. (1995). Generalized Upper Model 2.0: documentation. Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany.

[Battaglia, 2004] Battaglia, A. G. C. C. M. (2004). Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In Pisanelli, D. M., editor, *Ontologies in Medicine*. IOS Press, Amsterdam.

[Beckner et al., 2009] Beckner, C., Blythe, R., Bybee, J., Christiansen, M. H., Croft, W., Ellis, N. C., Holland, J., Ke, J., Larsen-Freeman, D., and Schoenemann, T. (2009). Language is a complex adaptive system: Position paper. *Language Learning*, 59:1–26.

[Bender et al., 2002] Bender, E. M., Flickinger, D., and Oepen, S. (2002). The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars the grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. *COLING-02: Grammar Engineering and Evaluation*.

[Bergen and Chang, 2005] Bergen, B. K. and Chang, N. C. (2005). Embodied Construction Grammar in Simulation-Based Language Understanding. In Östman, J.-O. and Fried, M., editors, *Construction Grammars: Cognitive Grounding and Theoretical Extensions*, pages 147–190. John Benjamins.

[Bergen et al., 2001] Bergen, B. K., Chang, N. C., and Paskin, M. A. (2001). Simulation-based language understanding in Embodied Construction Grammar. In Östman, J.-O., editor, *Construction Grammar(s): Cognitive and Cross-language dimensions*. John Benjamins.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, May.

[Beuls, 2011] Beuls, K. (2011). Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In Steels, L., editor, *Design Patterns in Fluid Construction Grammar*, pages 237–264. John Benjamins, Amsterdam.

[Beuls, 2012] Beuls, K. (2012). Handling scope in Fluid Construction Grammar: A case study for Spanish modals. In Steels, L., editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

[Beuls and Van Eecke, 2023] Beuls, K. and Van Eecke, P. (2023). Fluid Construction Grammar: State of the art and future outlook. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 41–50.

[Beuls and Van Eecke, 2024] Beuls, K. and Van Eecke, P. (2024). *Construction grammar and artificial intelligence*. Cambridge University Press.

[Boas, 2002] Boas, H. (2002). Bilingual framenet dictionaries for machine translation. In González Rodríguez, M. and Araujo, C. P. S., editors, *Proceedings of the Third International Conference on Language*

Resources and Evaluation, volume IV, pages 1364–1371, Las Palmas, Spain.

[Boas, 2008] Boas, H. C. (2008). Towards a frame-constructional approach to verb classification. Grammar, Constructions, and Interfaces. Special Issue of Revista Canaria de Estudios Ingleses.

[Borgo and Masolo, 2009] Borgo, S. and Masolo, C. (2009). Foundational choices in dolce. *Handbook on ontologies*, pages 361–381.

[Borjars et al., 2019] Borjars, K., Nordlinger, R., and Sadler, L. (2019). *Lexical-Functional Grammar*. Cambridge University Press.

[Bradley, 2020] Bradley, A. (2020). Brace yourself for an explosion of virtual assistants. *https://blogs.gartner.com/anthony*$_b$*radley*/2020/08/10/*brace* − *yourself* − *for* − *an* − *explosion* − *of* − *virtual* − *assistants*/, 01/04/2022.

[Bresnan, 2001] Bresnan, J. (2001). *Lexical-functional Grammar*. Blackwell, Oxford.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. In *Artificial Intelligence*, number 47, pages 139–159. MIT Press, Cambridge, MA.

[Bryant, 2004] Bryant, J. (2004). Scalable construction-based parsing and semantic analysis. In *Proceedings of the 2nd International Workshop on Scalable Natural Language Understanding (ScaNaLU)*, Boston, Mass, USA.

[Buitelaar et al., 2006] Buitelaar, P., Declerck, T., Frank, A., Racioppa, S., Kiesel, M., Sintek, M., Engel, M., Romanelli, M., Sonntag, D., Loos, B., Micelli, V., Porzel, R., and Cimiano, P. (2006). Linginfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proceedings of OntoLex 2006*, Genoa, Italy.

[Carpenter, 1992] Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, U.K.

[Chang, 2008] Chang, N. (2008). *Constructing grammar: A computational model of the emergence of early constructions*. PhD thesis, Computer Science Division, University of California at Berkeley.

[Chang et al., 2012] Chang, N., De Beule, J., and Micelli, V. (2012). Computational construction grammar: Comparing ecg and fcg. In Steels, L., editor, *Computational Issues in Fluid Construction Grammar*, volume 7249 of *Lecture Notes in Computer Science*, pages 259–288. Springer, Berlin.

[Chang et al., 2002] Chang, N., Feldman, J., Porzel, R., and Sanders, K. (2002). Scaling cognitive linguistics: Formalisms for language understanding. In *Proceedings of the 1st International Workshop on Scalable Natural Language Understanding (ScaNaLU)*, Heidelberg, Germany.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*. Mouton and Co, The Hague.

[Chomsky, 1965] Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass.

[Chomsky, 2006] Chomsky, N. (2006). *Language and Mind*. Cambridge University Press, 3rd edition.

[Cienki, 1997] Cienki, A. (1997). Some properties and groupings of image schemas. In Marjolijn Verspoor, Kee Dong Lee, E. S., editor, *Lexical and Syntactical Constructions and the Construction of Meaning*, pages 3–15. John Benjamins, Amsterdam.

[Cimiano, 2006] Cimiano, P. (2006). *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer.

[Clausner and Croft, 1999] Clausner, T. and Croft, W. (1999). Domains and image schemas. *Cognitive Linguistics*, 10:1–31.

[Copestake, 2000] Copestake, A. (2000). Appendix: Definitions of typed feature structures. *Natural Language Engineering*, 6(1):109–112.

[Croft, 2001] Croft, W. (2001). *Radical Construction Grammar*. Oxford University Press.

[Croft, 2005] Croft, W. (2005). Logical and Typological Arguments for Radical Construction Grammar. In Östman, J.-O. and Fried, M., editors, *Construction Grammars: Cognitive Grounding and Theoretical Extensions*, pages 273–314. John Benjamins.

[Davies et al., 2006] Davies, J., Studer, R., and Warren, P., editors (2006). *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons.

[De Beule and Steels, 2005] De Beule, J. and Steels, L. (2005). Hierarchy in Fluid Construction Grammar. In Furbach, U., editor, *KI 2005: Advances In Artificial Intelligence. Proceedings of the 28th German Conference on AI*, volume 3698 of *Lecture Notes in Artificial Intelligence*, Koblenz. Springer.

[de Saussure, 1985] de Saussure, F. (1916/1985). *Cours de linguistique générale*. Bibliothèque scientifique Payot, Paris, France.

[Diessel, 2004] Diessel, H. (2004). The acquisition of complex sentences. *Cambridge Studies in Linguistics*, 105.

[Dik, 1978] Dik, S. C. (1978). *Functional grammar*. North-Holland.

[Dodge and Lakoff, 2005] Dodge, E. and Lakoff, G. (2005). Image schemas: From linguistic analysis to neural grounding. In Hampe, B. and Grady, J., editors, *From Perception to Meaning: Image Schemas in Cognitive Linguistics*, pages 57–92. Mouton de Gruyter, Berlin, New York.

[Doumen et al., 2023] Doumen, J., Beuls, K., and Van Eecke, P. (2023). Modelling language acquisition through syntactico-semantic pattern finding. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1347–1357.

[Dourish, 2001] Dourish, P. (2001). *Where The Action Is: The Foundations of Embodied Interaction*. MIT Press.

[Falbo, 2014] Falbo, R. (2014). Sabio: Systematic approach for building ontologies. *Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering co-located with 8th International Conference on Formal Ontology in Information Systems*.

[Feldman et al., 1996] Feldman, J., Lakoff, G., Bailey, D., Narayanan, S., Regier, T., and Stolcke, A. (1996). l_0—the first five years of an automated language acquisition project. *AI Review*, 10:103–129.

[Feldman, 2006] Feldman, J. A. (2006). *From Molecule to Metaphor: A Neural Theory of Language*. MIT Press, Cambridge, MA.

[FIBO, 2022] FIBO (2022). Financial industry business ontology.

[Fillmore, 1982] Fillmore, C. (1982). Frame semantics. *Linguistics in the morning calm*, pages 111–137.

[Fillmore, 1985a] Fillmore, C. (1985a). Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254.

[Fillmore, 1985b] Fillmore, C. (1985b). Syntactic intrusions and the notion of grammatical construction. In et. al., M. N., editor, *Proceedings of the Eleventh Annual Meeting of the Berkeley Linguistics Society*, pages 73–86, Berkeley. Berkeley Linguistics Society.

[Fillmore, 1988] Fillmore, C. (1988). The mechanisms of construction grammar. In *Berkeley Linguistics Society*, volume 14, pages 35–55.

[Fillmore and Kay, 1987] Fillmore, C. and Kay, P. (1987). The goals of construction grammar. Technical Report 50, University of California, Berkeley.

[Fischer and Stefanowitsch, 2006] Fischer, K. and Stefanowitsch, A., editors (2006). *Konstruktionsgrammatik: Von der Anwendung zur Theorie*. Stauffenburg Linguistik, Tübingen, Germany.

[Francez and Wintner, 2011] Francez, N. and Wintner, S. (2011). *Unification Grammars*. Cambridge University Press.

[Fujii, 1993] Fujii, S. Y. (1993). *The Use and Learning of Clause-linkage: Case Studies in Japanese and English Conditionals*. PhD thesis, University of California, Berkeley.

[Furetière, 1658] Furetière, A. (1658). *Nouvelle Allegorique, Ou Histoire Des Derniers Troubles Arrivez Au Royaume D'Eloquence*.

[Gangemi et al., 2004] Gangemi, A., Borgo, S., Catenacci, C., and Lehmann, J. (2004). Task taxonomies for knowledge content. Technical report, Metokis Project.

[Gangemi et al., 2002] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening ontologies with dolce. In Gomez-Perez, A. and Benjamins, V., editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002*, pages 166–181. Springer.

[Gangemi and Mika, 2003] Gangemi, A. and Mika, P. (2003). Understanding the semantic web through descriptions and situations. In *Proceedings of the ODBASE Conference*. Springer.

[Gangemi et al., 2003a] Gangemi, A., Navigli, R., and Velardi, P. (2003a). The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. pages 270–288.

[Gangemi et al., 2003b] Gangemi, A., Sagri, M.-T., and Tiscornia, D. (2003b). A constructive framework for legal ontologies. In *Law and the Semantic Web*, pages 97–124.

[Garvin, 1954] Garvin, P. L. (1954). Prolegomena to a theory of language by louis hjelmslev; francis j. whitfield. *Language*, 30(69–96).

[Goldberg, 1995] Goldberg, A. E. (1995). *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

[Goldberg, 2006a] Goldberg, A. E. (2006a). *Constructions at Work, The Nature of Generalization in Language*. Oxford University Press, Great Clarendon Street, Oxford.

[Goldberg, 2006b] Goldberg, A. E. (2006b). *Constructions at Work: the nature of generalization in language*. Oxford University Press.

[Gromann and Macbeth, 2019] Gromann, D. and Macbeth, J. C. (2019). Crowdsourcing image schemas. *Computer Science: Faculty Publications Smith College, Northampton, MA*.

[Gross, 2002] Gross, G. (2002). Comment décrire une langue de spécialité? *Cahiers de lexicologie: revue internationale de lexicologie et lexicographie*, 80:179–200.

[Gruber, 1993] Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition (5)*.

[Grüninger and Fox, 1995] Grüninger, M. and Fox, M. (1995). Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*.

[Guarino, 1997] Guarino, N. (1997). Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *SCIE*, pages 139–170.

[Guarino, 1998] Guarino, N. (1998). Formal ontology and information systems.

[Guarino, 2006] Guarino, N. (2006). Ontology library. wonderweb deliverable d202.

[Hitzler et al., 2009] Hitzler, P., Krötzsch, M., and Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

[Hitzler and Shimizu, 2018] Hitzler, P. and Shimizu, C. (2018). Modular ontologies as a bridge between human conceptualization and data. *ICCS*.

[Hopper and Traugott, 2003] Hopper, P. and Traugott, E. (2003). *Grammaticalization*. Cambridge University Press.

[Jackendoff, 1983] Jackendoff, R. (1983). *Semantics and Cognition*. MIT Press, Cambridge, MA/London.

[Johnson, 1987] Johnson, M. (1987). *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. University of Chicago Press.

[Joshi and Schabes, 1997] Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, New York.

[Jurafsky and Martin, 2008] Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing*. Prentice Hall.

[Kartunnen, 1989] Kartunnen, L. (1989). Radical lexicalism. In Baltin, M. and Kroch, A., editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago, USA.

[Katz, 1972] Katz, J. (1972). *Semantic Theory*. Harper & Row.

[Kay, 1984] Kay, M. (1984). Functional unification grammar: A formalism for machine translation. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 75–78.

[Kay, 1997] Kay, P. (1997). Construction Grammar. In *Words and the Grammar of Context*, pages 123–131. Stanford:CSLI.

[Kay, 2002] Kay, P. (2002). An informal sketch of a formal architecture for construction grammar. *Grammars*, 5(1).

[Kay and Fillmore, 1999] Kay, P. and Fillmore, C. (1999). Grammatical constructions and linguistic generalizations: the *What's X doing Y?* construction. *Language*, 75(1):1–33.

[Kersloot et al., 2020] Kersloot, M. G., van Putten, F. J. P., Abu-Hanna, A., Cornet, R., and Arts, D. L. (2020). Natural language processing algorithms for mapping clinical text fragments onto ontology concepts: a systematic review and recommendations for future studies. *Journal of Biomedical Semantics*, 11(1):14.

[Klabunde, 1998] Klabunde, R. (1998). *Formale Grundlagen der Linguistik*. Gunter Narr Verlag, Tuebingen.

[Kolb, 2004] Kolb, P. (2004). Graphentheorie und Merkmalsstrukturen. In Carstensen, K.-U., Ebert, C., Endriss, C., Jekat, S., Klabunde, R., and Langer, H., editors, *Computerlinguistik und Sprachtechnologie*, pages 91–110. Spektrum Akademischer Verlag, Heidelberg, Germany.

[Kuhn, 2005] Kuhn, W. (2005). Geospatial semantics: Why, of what, and how? In *J. Data Semantics III*, pages 1–24.

[Kuhn, 2007] Kuhn, W. (2007). An image-schematic account of spatial categories. In *Proceedings of the 8th international conference on Spatial information theory*, COSIT'07, pages 152–168, Berlin, Heidelberg. Springer-Verlag.

[Kushmerick et al., 1997] Kushmerick, N., Weld, D., and Doorenbos, R. (1997). Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*.

[Lakoff, 1987] Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press.

[Lakoff and Johnson, 1980] Lakoff, G. and Johnson, M. (1980). *Metaphors We Live By*. University of Chicago Press.

[Lakoff and Johnson, 1999] Lakoff, G. and Johnson, M. (1999). *Philosophy in the Flesh. The Embodied Mind and its Challenge to Western Thought*. Basic Books, New York.

[Lakoff and Turner, 1989] Lakoff, G. and Turner, M. (1989). *More Than Cool Reason: A Field Guide to Poetic Metaphor*. University of Chicago Press.

[Langacker, 2003] Langacker, R. (2003). Construction grammars: Cognitive, radical, and less so. In *Proceedings of 8th International Conference on Cognitive Linguistics*.

[Langacker, 1987] Langacker, R. W. (1987). *Foundations of Cognitive Grammar, Vol. 1*. Stanford University Press.

[Langacker, 1991] Langacker, R. W. (1991). *Concept, Image, and Symbol: The Cognitive Basis of Grammar*. Cognitive Linguistics Research. Mouton de Gruyter, Berlin and New York.

[Langacker, 2000] Langacker, R. W. (2000). A dynamic usage-based model. In Kemmer and Barlow, editors, *Topics in Cognitive Linguistics*, pages 127–161. John Benjamins, Amsterdam.

[Lenat, 1995] Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

[Lezius, 2000] Lezius, W. (2000). Morphy - german morphology, part-of-speech tagging and applications. In *Proceedings of the 9th EURALEX International Congress*.

[Lieven and Tomasello, 2008] Lieven, E. and Tomasello, M. (2008). Childrenõs first language acquisition from a usage-based perspective. In Robinson, P. and Ellis, N., editors, *Handbook of Cognitive Linguistics and Second Language Acquisition*. Routledge.

[Litz, 2010] Litz, B. (2010). *Incremental ontology extension*. PhD thesis.

[Loos et al., 2004] Loos, E. E., Anderson, S., Day, Dwight H., J., Jordan, P. C., and Wingate, J. D. (2004). Glossary of linguistic terms. volume 5. SIL International.

[Mahesh and Nirenburg, 1995] Mahesh, K. and Nirenburg, S. (1995). A situated ontology for practical nlp. In *Workshop on Basic Ontological Issues in Knowledge Sharing at the International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada.

[Mandler, 1992] Mandler, J. M. (1992). How to build a baby ii: Conceptual primitives. *Psychological Review*, 99:587–604.

[Marques and Beuls, 2016] Marques, T. and Beuls, K. (2016). Evaluation strategies for computational construction grammars. In *COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1137–1146, Osaka, Japan. The COLING 2016 Organizing Committee.

[Masolo et al., 2003] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). Ontology library. wonderweb deliverable d18.

[Masolo et al., 2005] Masolo, C., Guarino, N., Oltramari, A., and Shneider, L. (2005). The wonderweb library of foundational ontologies. Wonderweb deliverable d18.

[Matsumoto, 1989] Matsumoto, Y. (1989). *Grammar and Semantics of Adnominal Clauses in Japanese*. PhD thesis, University of California, Berkeley.

[Micelli, 2004] Micelli, V. (2004). Von einzelnen Wö rtern zu grammatischen Konstruktionen. In *Master Thesis at the University of Heidelberg*.

[Micelli, 2009] Micelli, V. (2009). An FCG Approach to Word Order in German Declarative Sentences. In *Presentation at the Workshop on Grammar Theory and Grammar Implementation, Free University, Berlin, Germany*.

[Micelli, 2012] Micelli, V. (2012). Field topology and information structure: A case study for German constituent order. In Steels, L., editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

[Micelli et al., 2009] Micelli, V., van Trijp, R., and Beule, J. D. (2009). Framing Fluid Construction Grammar. In Taatgen, N. and van Rijn, H., editors, *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 3023–3027. Cognitive Science Society.

[Müller, 1999] Müller, S. (1999). *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Number 394 in Linguistische Arbeiten. Max Niemeyer Verlag, Tübingen.

[Müller, 2008] Müller, S. (2008). *Head-Driven Phrase Structure Grammar: Eine Einführung*. Number 17 in Stauffenburg Einführungen. Stauffenburg Verlag, Tübingen, 2 edition.

[Müller, 2021] Müller, S. (2021). Hpsg bibliography https://hpsg.hu-berlin.de/hpsg-bib/.

[Murata, 1989] Murata, T. (1989). Petri nets: Properties, analysis, and applications. In *Proc. IEEE-89*, volume 77, pages 541–576.

[Narayanan, 1997] Narayanan, S. (1997). *KARMA: Knowledge-Based Active Representations for Metaphor and Aspect*. PhD thesis, Computer Science Division, University of California, Berkeley, Cal.

[Nevens et al., 2022] Nevens, J., Doumen, J., Van Eecke, P., and Beuls, K. (2022). Language acquisition through intention reading and pattern finding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 15–25.

[Nevens et al., 2019] Nevens, J., Van Eecke, P., and Beuls, K. (2019). Computational construction grammar for visual question answering. In *Linguistics Vanguard*, volume Linguistics Vanguard.

[Niles and Pease, 2001] Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In Welty, C. and Smith, B., editors, *Workshop on Ontology Management*, Ogunquit, Maine. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001).

[Noy and McGuinness, 2001] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University School of Medicine.

[Nunez, 1999] Nunez, R. (1999). Could the future taste purple? reclaiming mind, body, and cognition. In Nunez, R. and Freeman, W. J., editors, *Reclaiming Cognition: The Primacy of Action, Intention and Emotion*, pages 41–60. Imprint Academic, Thorverton, UK.

[Oberle, 2006] Oberle, D. (2006). *The Semantic management of middleware*. Springer, New York, NY.

[Östman and Fried, 2005] Östman, J.-O. and Fried, M. (2005). The cognitive grounding of construction grammar. In Östman, J.-O. and Fried, M., editors, *Construction Grammars, Cognitive grounding and theoretical extensions*, volume 3, chapter 1, pages 1–14. John Benjamins Publishing Company, Amsterdam/Philadelphia.

[Paranjape et al., 2020] Paranjape, A., See, A., Kenealy, K., Li, H., Hardy, A., Qi, P., Sadagopan, K. R., Phu, N. M., Soylu, D., and Manning, C. D. (2020). Neural generation meets real people: Towards emotionally engaging mixedinitiative conversations. *3rd Proceedings of Alexa Prize*.

[Pellet, 2022] Pellet (2022). http://pellet.owldl.com/.

[Pereira and Shieber, 1984] Pereira, F. C. N. and Shieber, S. M. (1984). The semantics of grammar formalisms seen as computer languages. *COLING-84*, pages 123–129.

[Petruck, 1996] Petruck, M. R. (1996). Frame semantics. In Verschueren, J., Ostman, J., Blommaert, J., and Bulcaen, C., editors, *Handbook of Pragmatics*. John Benjamins, Philadelphia.

[Pilla, 1971] Pilla, K. (1971). *The Vākyapadīya: Critical Text of Cantos I and II: (with English Translation, Summary of Ideas and Notes)*. Studies in the Vākyapadīya, v. 1. Motilal Banarsidass.

[Pinker, 1989] Pinker, S. (1989). *Learnability and Cognition*. MIT Press, Cambridge, MA/London.

[Pollard and Sag, 1987] Pollard, C. and Sag, I. A. (1987). *Information-based Syntax and Semantics, Vol. 1*. Number 13 in Lecture Notes. CSLI Publications, Stanford University.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

[Raja, 1969] Raja, K. K. (1969). *Indian Theories of Meaning*. Adyar, Madras: Adyar Library and Research Centre, 2nd ed edition.

[Rizk, 2020] Rizk, Y. (2020). A conversational digital assistant for intelligent process automation. In *LNBIP*, volume 393, pages 85–100.

[Rizzolatti et al., 1996] Rizzolatti, G., Fadiga, L., Gallese, V., and Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3:131–141.

[Sag, 2012] Sag, I. A. (2012). Sign-based construction grammar: An informal synopsis. In Boas, H. C. and Sag, I. A., editors, *Sign-BasedConstruction Grammar*, pages 61–197, Stanford, CA. CSLI Publications.

[Sanfilippo and Borgo, 2015] Sanfilippo, E. M. and Borgo, S. (2015). Feature-based modelling and information systems for engineering. *Advances in Artificial Intelligence*, pages 151–163.

[Schneider, 2003] Schneider, L. (2003). Designing foundational ontologies: The object-centered high-level reference ontology ochre as a case study. volume 2813, pages 91–104.

[Seelbach, 2001] Seelbach, D. (2001). Das kleine multilinguale Fussball-Lexikon. *Philologica et Linguistica. Historia, Pluralitas, Universitas*.

[Serban et al., 2018] Serban, I. V., Lowe, R., Henderson, P., Charlin, L., and Pineau, J. (2018). A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue Discourse*, 9:1–49.

[Sharma et al., 2019] Sharma, M., Singh, G., and Singh, R. (2019). Design of ga and ontology-based nlp frameworks for online opinion mining. *Recent Patents on Engineering*, 13:159–165.

[Shieber, 1985] Shieber, S. M. (1985). Evidence against the non-context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

[Shieber, 1986] Shieber, S. M. (1986). *An Introduction to Unification-based Approaches to Grammar*. Stanford University/CSLI, Stanford, Cal.

[Simon, 2005] Simon, K.and Lausen, G. (2005). Viper: Augmenting automatic information extraction with visual perceptions. In *Proceedings of the 2005 ACM International Conference on Information and Knowledge Management (CIKM '05)*.

[Spranger and Loetzsch, 2011] Spranger, M. and Loetzsch, M. (2011). Syntactic indeterminacy and semantic ambiguity: A case study for German spatial phrases. In Steels, L., editor, *Design Patterns in Fluid Construction Grammar*, pages 265–298. John Benjamins, Amsterdam.

[Steels, 2000] Steels, L. (2000). Language as a complex adaptive system. *Lecture Notes in Computer Science*, pages 17–26.

[Steels, 2001] Steels, L. (2001). Language games for autonomous robots. *IEEE Intelligent Systems*, pages 16–22.

[Steels, 2003] Steels, L. (2003). Evolving grounded communication for robots. In *Trends in Cognitive Science Volume 7*, pages 308–312.

[Steels, 2004] Steels, L. (2004). Constructivist development of grounded construction grammars. In Daelemans, W., editor, *Proceedings Annual Meeting of Association for Computational Linguistics*, Barcelona. ACL.

[Steels, 2005] Steels, L. (2005). The Role of Construction Grammar in Fluid Language Grounding. *Submitted*.

[Steels, 2011] Steels, L., editor (2011). *Design Patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam.

[Steels, 2012] Steels, L., editor (2012). *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

[Steels, 2017] Steels, L. (2017). Basics of Fluid Construction Grammar. *Constructions and Frames*, 9:178–225.

[Steels and De Beule, 2006] Steels, L. and De Beule, J. (2006). A (very) brief introduction to fluid construction grammar. In *Proceedings of the HLT-NAACL 2006 Workshop on Scalable Natural Language Understanding (ScaNaLU 2006)*, pages 73–80, New York, USA.

[Talmy, 1988] Talmy, L. (1988). Force dynamics in language and cognition. *Cognitive Science*, 12:49–100.

[Tesnière, 1959] Tesnière, L. (1959). *Eléments de Syntaxe Structurale*. Klincksieck.

[Tomasello, 2003] Tomasello, M. (2003). *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Harvard University Press.

[Tomasello, 2014] Tomasello, M. (2014). *Introduction: A cognitive-functional perspective on language structure*, volume 1 of *Cognitive and Functional Approaches to Language Structure*. Psychology Press Ltd.

[Tomasello and Brooks, 1999] Tomasello, M. and Brooks, P. J. (1999). Early syntactic development: A construction grammar approach. In Barrett, M., editor, *The Development of Language*. Psychology Press, London.

[Upward and Jones, 2016] Upward, A. and Jones, P. H. (2016). An ontology for strongly sustainable business models. *Organization & Environment*, 29:123 – 97.

[Uszkoreit and Zaenen, 1997] Uszkoreit, H. and Zaenen, A. (1997). Grammar formalisms. In Cole, R. A., Mariani, J., Uszkoreit, H., Varile, G., Zaenen, A., Zue, V., and Zampolli, A., editors, *Survey of the State of the Art in Human Language Technology*, pages 100–102. Cambridge University Press and Giardini, Cambridge.

[Van Eecke, 2018] Van Eecke, P. (2018). *Generalisation and Specialisation Operators for Computational Construction Grammar and their Application in Evolutionary Linguistics Research*. PhD thesis, Vrije Universiteit Brussel.

[Van Eecke et al., 2022] Van Eecke, P., Nevens, J., and Beuls, K. (2022). Neural heuristics for scaling constructional language processing. *Journal of Language Modelling*, 10:287–314.

[Van Eecke et al., 2023] Van Eecke, P., Verheyen, L., Willaert, T., and Beuls, K. (2023). The candide model: How narratives emergence where observations meet beliefs. In *Proceedings of the 5th Workshop on Narrative Understanding*, pages 48–57, Toronto, Canada. Association for Computational Linguistics.

[van Trijp, 2008] van Trijp, R. (2008). The emergence of semantic roles in Fluid Construction Grammar. *The Evolution of Language. Proceedings of the 7th International Conference (EVOLANG 7)*, pages 346–353.

[van Trijp, 2009] van Trijp, R. (2009). Distinctive Feature Matrices in Fluid Construction Grammar. In *Presentation at the Workshop*

on Grammar Theory and Grammar Implementation, Free University, Berlin, Germany.

[van Trijp, 2010] van Trijp, R. (2010). Argument realization in Fluid Construction Grammr. In Boas, H. C., editor, *Computational Approaches to Construction Grammar and Frame Semantics*. John Benjamins, Amsterdam.

[van Trijp, 2011] van Trijp, R. (2011). Feature matrices and agreement: A case study for German case. In Steels, L., editor, *Design Patterns in Fluid Construction Grammar*, pages 205–235. John Benjamins, Amsterdam.

[van Trijp, 2012] van Trijp, R. (2012). Not as awful as it seems: Explaining german case through computational experiments in fluid construction grammar. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics.*, Avignon: France.

[van Trijp, 2017] van Trijp, R. (2017). How a construction grammar account solves the auxiliary controversy. *Constructions and Frames*, 9(2):251 – 277.

[van Trijp et al., 2022] van Trijp, R., Beuls, K., and Van Eecke, P. (2022). The FCG editor: An innovative environment for engineering computational construction grammars. *PLOS ONE*, 17(6): e0269708.

[Van Valin and LaPolla, 1997] Van Valin, R. and LaPolla, R. (1997). *Syntax: structure, meaning and function.* Cambridge University Press, Cambridge.

[Verhagen, 2009] Verhagen, A. (2009). The conception of constructions as complex signs: Emergence of structure and reduction to usage. *Constructions and Frames*, 1(1):119–152.

[Verheyen et al., 2023] Verheyen, L., Ekila, J. B., Nevens, J., Van Eecke, P., and Beuls, K. (2023). Neuro-symbolic procedural semantics for

reasoning-intensive visual dialogue tasks. In *ECAI 2023*, pages 2419–2426. IOS Press.

[Wahlster, 2007] Wahlster, W. (2007). Smartweb - ein multimodales Dialogsystem für das semantische Web. *40 Jahre Informatikforschung in Deutschland*.

[Weissweiler et al., 2022] Weissweiler, L., Hofmann, V., Köksal, A., and Schütze, H. (2022). The better your syntax, the better your semantics? probing pretrained language models for the english comparative correlative. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10859–10882, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza – a computer program for the study of natural language communication between man and machine. *CACM*, 9:36–45.

[Wierzbicka, 1996] Wierzbicka, A. (1996). *Semantics. Primes and Universals*. Oxford University Press, Oxford.

[Wilks, 2009a] Wilks, Y. (2009a). *Machine Translation. Its Scope and Limits*. Berlin: Springer Verlag, Berlin, Germany.

[Wilks, 2009b] Wilks, Y. (2009b). *Machine Translation. Its Scope and Limits*. Springer Verlag, Berlin.

[Willaert et al., 2022] Willaert, T., Banisch, S., Van Eecke, P., and Beuls, K. (2022). Tracking causal relations in the news: data, tools, and models for the analysis of argumentative statements in online media. In *Digital Scholarship in the Humanities*, volume 37(4), pages 1358–1375.

Appendix A

Appendix

A.1 The Complete LingInfo Model

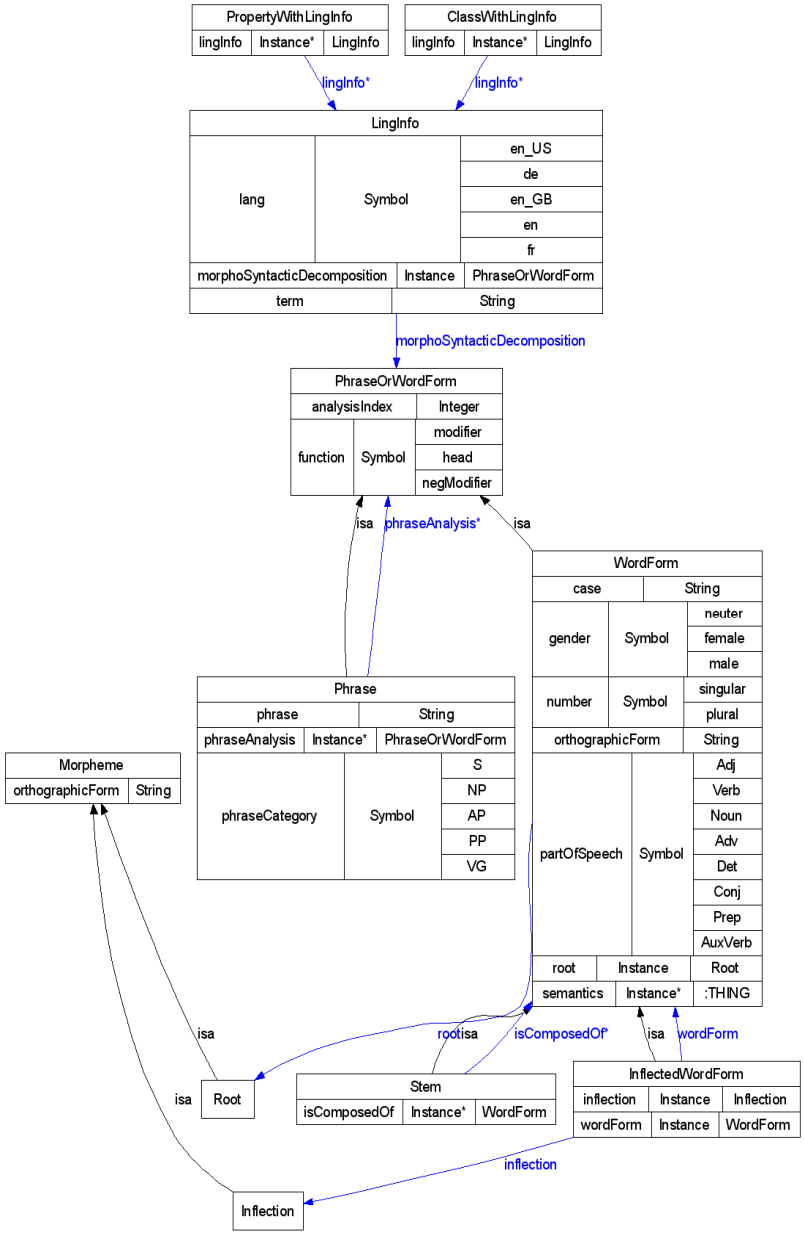


Figure A.1: The complete LingInfo model.

A.2 An Image Schema Hierarchy

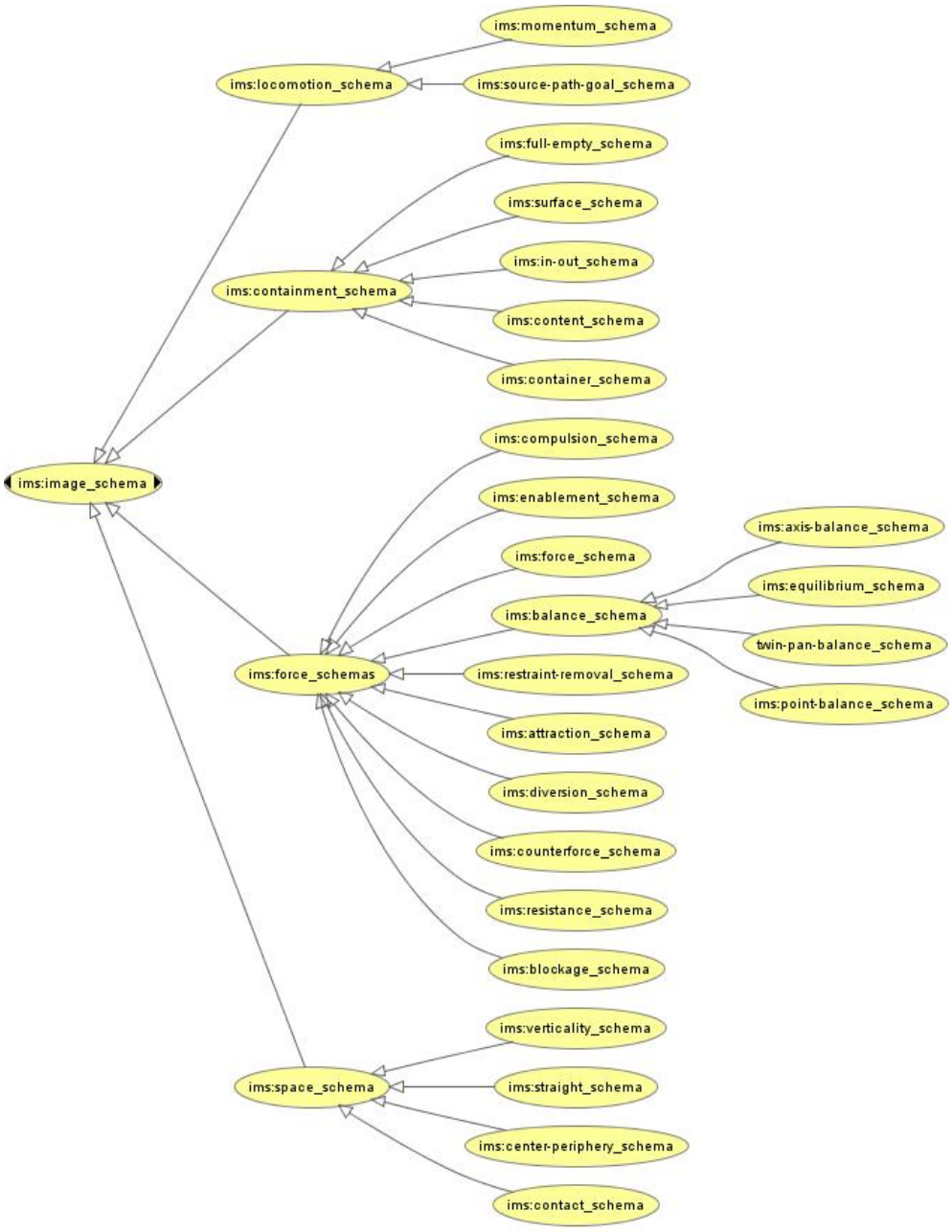


Figure A.2: A hierarchy of most important image schemas

List of Acronyms

CUG Categorial Unification Grammar

CxG Construction Grammar

DOLCE Descriptive Ontology for Linguistic and Cognitive Engineering

DnS Descriptions and Situations

EAGLES Expert Advisory Group on Language Engineering Standards

ECG Embodied Construction Grammar

FCG Fluid Construction Grammar

FUG Functional Unification Grammar

HPSG Head-Driven Phrase-Structure Grammar

ISLE International Standards for Language Engineering

LFG Lexical Functional Grammar

MOF Morphy Output Filter

NLP Natural Language Processing

NP Noun Phrase

OCHRE Object-Centered High-Level Reference Ontology

OIO Ontology of Information Objects

OWL Ontology Web Language

OWL-DL OWL-Description Logic

RDF Resource Description Framework

RDFS Resource Description Framework Schema

SVO Subject Verb Object

SKOS Simple Knowledge Organization Systems

SUMO Suggested Upper Merged Ontology

TAG Tree Adjunction Grammar

UI User Interface

List of Figures