Thesis for The Degree of Licentiate of Philosophy

Computer-Assisted Language Learning with Grammars

A Case Study on Latin Learning

Herbert Lange

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Division of Functional Programming
Department of Computer Science & Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden, 2018

Computer-Assisted Language Learning with Grammars
A Case Study on Latin Learning

Herbert Lange

Technical Report No 181L
ISSN 1652-876X
Department of Computer Science & Engineering
Division of Functional Programming
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Cover image: Mullus surmuletus, in D. Marcus Elieser Bloch's, ausübenden Arztes zu Berlin Ökonomische Naturgeschichte der Fische Deutschlands - mit sieben und dreyssig Kupfertafeln nach Originalen (1783) (Source: Wikimedia)

This thesis has been prepared using LuaLaTeX.
Printed by Lorensberg Reproservice,
Gothenburg, Sweden 2018.

"Det är én måde at forstå
en anden kultur på. At *leve* den. Att flytte ind i den, at bede om
at blive tålt som gæst, att lære sig sproget. På et eller andet
tidspunkt kommer så måske forståelsen."
- *Frøken Smillas fornemmelse for sne*[1]

"Det finns ett enda sätt för att förstå en annan kultur. Att *leva*
den. Att flytta in i den, att be att få bli tåld som gäst, att lära sig
språket. Förr eller senare kommer kanske förståelse."
- *Fröken Smillas känsla för snö*[2]

"There is *one* way to understand another culture. *Living* it. Move
into it, ask to be tolerated as a guest, learn the language. At some
point understanding may come."
- *Miss Smilla's feeling for snow*[3]

"Tantum unus modus est intelligere aliam culturam. In ea *vivere*.
In eam collocare, qua hospes tolerari precari, linguam discere.
Serius ocius esset intellectum."
- *Virginis Smillae intuitio pro nive*[4]

[1] Høeg (2006)
[2] Høeg (1993b)
[3] Høeg (1993a)
[4] Own translation

Abstract

Learning new languages has a high relevance in today's society with a globalized economy and the freedom to move abroad for work, study or other reasons. In this context new methods to teach and learn languages with the help of modern technology are becoming more relevant besides traditional language classes.

This work presents a new approach to combine a traditional language class with a modern computer-based approach for teaching. As a concrete example a web application to help teach and learn Latin was developed.

Keywords

Compuer-Assisted Language Learning, Controlled Natural Language, Latin

Acknowledgment

I would like to start by thanking Hans Leiss, my former teacher and master's supervisor in Munich, who taught me the formal approaches to languages (including Montague Grammar, Categorial Grammar and Grammatical Framework, of course) and gave me the motivation to remain in academia to pursue my PhD.

Next I want to thank my supervisor Peter Ljunglöf who did not only support me in my work whenever I needed it but also gave great recommendations for music and concerts.

Thanks go also to my co-supervisor Koen Claessen and my Examiner Aarne Ranta who were always open to give feedback whenever asked for.

Furthermore I want to thank Torsten Zesch for his willingness to become my discussion leader and coming to Gothenburg for my Licentiate seminar.

I also appreciate the inspiring discussions with my office mates and my colleagues in general, but especially over lunch and beers on many evenings. My work here would not be the same without you.

A special thanks has to go to my family who had to accept that I move far away to follow my dreams but still support me in every way possible.

Finally I want to thank my friends here in Gothenburg who make my life enjoyable, including my choir, fencing club, photo club, hackerspace and many more.

A few people have to be named specifically: Aljoscha (for heading the PhD council, nice rounds of Skat and more), Daniel (for many nice beers, Skat, Rumble in the Pub and more), David (more distant as a colleague but more close as a neighbor, with whom I discussed many topics over plenty of folk öl), Eike (for Skat as well as karaoke and free beer at his company), Inari (the best office and boat mate I could imagine, fermentation parties and so much more), Jesper (for great Belgian food and beer), Linnea (who first thought I was Swedish because I didn't talk in hours, for helping me with learning Swedish, lending me cute children's books and being a friend in general), Pavul (for Humanist afterworks and lunches), Simon H. (fermentation parties and more), Simon R. (for being a great flat and boat mate), Thomas (for Skat, beers and as a ham radio operator the support to get a ham radio license myself, 73 de SA6HRB), Víctor (more fermentation, lunch discussions and more), and many more which I cannot all name here.

The MUSTE project is funded by the Swedish Research Council (Vetenskapsrådet) under grant number 621-2014-4788.

List of Publications

This thesis is based on the following publications:

[A] Herbert Lange "Implementation of a Latin Grammar in Grammatical Framework" *DATeCH2017, Göttingen, Germany, 2017*

[B] Herbert Lange and Peter Ljunglöf "Mulle: A grammar-based latin language learning tool to supplement the classroom setting", In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA '18), Melbourn. Australia, 2018*

[C] Herbert Lange and Peter Ljunglöf "Putting control into language learning" *SIGCNL 2018, Maynooth, Ireland, 2018*

Contents

Chapter 1

Introduction

Learning languages has a growing relevance in our globalized community. With an increased mobility it is easier and more likely to move to a foreign country: for work, studies or other reasons. With the ubiquity of computers the use of computers in language learning has increased and many new language learning platforms are available. So you can easily practice your Japanese at the breakfast table or learn some Spanish phrases in the bus on the way home. This quite different approach to language learning poses new challenges, both from a pedagogical and a technological point of view.

But not all languages people are interested in are as common as Japanese or Spanish, for which large amount of language resources are available. So dealing with a lack of language resources when creating a language learning application is a challenge which has to be addressed.

The main aspect of this thesis is the conceptualization of a language learning framework suitable to build full-fledged and flexible language learning systems that are suitable for less-resourced languages. The whole field of Computer-Assisted Language Learning (CALL) itself is very broad so we focus on the specific setting close to traditional classroom-style language learning. That means we have a look at how languages have been taught in the traditional way and extend this approach with a computer-based learning application that provides popular features of state-of-the-art language learning apps like gamification and an easy-to-use web interface. That way the traditional language classes, even for languages like Latin, which have not necessarily been too appealing to students for generations, can be almost as enjoyable as learning the language of a favorite holiday destination. By relying on grammars to encode the learning objectives we can guarantee for a high level of reliability.

The MUSTE Language Learning Environment (MULLE), the framework we present here, is not limited to one very specific use-case and language. Instead it provides a flexible way to control the content of the language learning application just in the way the grammars used are created. This places it between a pure language learning system and an authoring system for various language learning exercises.

To show the practicability of our approach we combine MULLE with a generic Latin grammar to build a workable language learning application for Latin. The choice of Latin can be seen as almost arbitrary, but was of course influenced by my previous work. Also, even though other approaches to language learning could be applied to less-resourced languages as well,[1] the competition for Latin seems very limited.

The MULLE system is developed as a Free and Open Source Software and we will release all resources created in the process where possible.

1.1 This Thesis

In this thesis we will have a look on aspects of CALL. The main focus is on a flexible and generic approach to teaching and learning second and foreign languages based on grammars. This system has several interesting properties, especially reliability, flexibility and being less resource hungry.

[1] Duolingo supports Klingon and High Valyrian among others

Based on the general concept we present a language learning application to teach and learn Latin in a closed and traditional classroom setting. We both developed the foundations and implemented a language learning application that has been tested with students in a language class.

1.1.1 Personal Motivation

The work of the last three years which resulted in this thesis, was inspired by my own situation:

- I moved to Sweden and struggled with learning Swedish. One of the available mobile language learning applications helped me to get a first basic intuition about the language
- My PhD project provided me with a intuitive way to modify text on touchscreen-based devices, that was looking for a real-world application and reminded me of a certain language learning application
- I am interested in historic languages and already worked on Latin for my master's thesis (Lange, 2013)

1.1.2 Research Objectives

The personal motivation from the previous section strongly influenced the main question we want to answer in this thesis: *How can we build a modern language learning application that is especially suitable for less-resourced and historic languages but offers the flexibility to extend its use far beyond that?*

This question led to a list of research objectives we follow up on:

- Design a general framework for language learning based on grammars that can be used with word-level text editing
- Show suitability for less-resourced languages
- Identify important characteristics of grammars for different kinds of language learning exercises
- Provide and test the concrete implementation of MULLE for Latin

These objects were the guidelines for our work and focused our efforts and we can claim that we mostly accomplished in completing these goals, but we also managed to extend the scope from an approach that just focuses on the learning side to also include features that support the teacher in creating teaching and learning resources.

1.2 Structure and Contributions

The work in this thesis mostly contributes to the field of Computer-Assisted Language Learning for second and foreign languages with minor contribution to the Digital Humanities in the form of the Latin grammar. The content is based on three publications, "Implementation of a Latin Grammar in Grammatical

Framework" (Lange, 2017), "MULLE: A Grammar-based Latin Language Learning Tool to Supplement the Classroom Setting" (Lange and Ljunglöf, 2018a) and "Putting Control into Language Learning" (Lange and Ljunglöf, 2018b) as well as extends them. The papers were all for the most part written by the author of this thesis. For (Lange, 2017) the author of this thesis was the sole author, while (Lange and Ljunglöf, 2018a) and (Lange and Ljunglöf, 2018b) were co-authered by Peter Ljunglöf. The individual contributions are explained in the corresponding sections of the theis.

1.2.1 Background

The first chapter (Chapter 2) provides the background to our work and helps contextualize it. It contains no new contribution and is not required for understanding the remaining content of the thesis.

The background provided covers gesture-based text input methods, Controlled Natural Languages, Latin resources and an overview of Computer-Assisted Language Learning.

1.2.2 Grammar-based Lessons

The two main parts of the thesis (Chapter 3 and 5) are both based on the two papers (Lange and Ljunglöf, 2018a,b). The papers overlap to a certain degree and together cover both the general foundations and the concrete implementation.

The first part (Chapter 3) focuses on the general ideas, starting with concepts not directly connected to language learning and bridging over to the central part of the thesis. It starts with a necessary introduction to the grammar formalism used and presents the general method of grammar-based text editing from Ljunglöf (2011). After this introduction the focus is shifted to language learning related topics: creating language learning lessons based on information found in textbooks and generate exercises within a lesson. The chapter is concluded with considerations about properties of suitable grammars and the language teaching approach we aim for.

The main contribution is the creation of a general language learning framework that works out of the box given a grammar and can provide different kinds of language learning exercises. Another important finding is the closeness of the suitable grammars to Controlled Natural Languages in general.

The original articles are co-authored by Peter Ljunglöf who made a list of valuable contributions:

- invented the method for word-based text editing in the first place
- provided knowledge about the background of gesture-based input
- contributed ideas about relaxing grammars for morphology exercises
- gave valuable feedback to both papers

About 80 percent of the content of (Lange and Ljunglöf, 2018a) and the whole content of (Lange and Ljunglöf, 2018b) was written by the author of this thesis. Peter Ljunglöf contributed one section about user input and one

section about relaxing grammars to the first paper as well as general feedback to both papers.

1.2.3 The Latin Resource Grammar as a Language Resource

A minor interlude is the description of the Latin Resource Grammar described in (Lange, 2017). It is part of the Grammatical Framework Resource Grammar Library which will be in larger extent described in Section 3.1.2.

As a step towards a concrete application for teaching less-resourced languages, with Latin as an example language, we sketch the construction of a computational grammar, a basic and general computational resource that can be created with limited human effort. It only requires some experience in grammar development and a generally available resource like a school grammar book.

The process of creating a Latin grammar including the challenges in both creating such a grammar in general and problems in handling Latin specifically are described in Chapter 4 as an example for how to develop a computational grammar. The content of the paper is extended by further development of the grammar as well as some evaluation of the coverage. These extensions include increased flexibility in word order and adding rules necessary to cover text fragments required for the language learning task.

The author of this thesis was the only author of both the grammar and the paper describing it.

1.2.4 The Latin Language Learning Application

The next to last part of this work is the design and implementation of a ready-to-use application to teach Latin based on the ideas of previous chapters.

Here we describe the step from the general ideas presented in Chapter 3 to a usable system in the specific application area of Latin learning to augment the traditional classroom setting. We give a more detailed description of the user interaction with the system, describe measures to improve the user experience and sketch an experiment for a full evaluation of the system including a description of a pilot for this experiment.

The main contribution is the presentation of a workable system based of the ideas of the previous chapters, using modern features for gamification. We also suggest a design for an experiment to fully evaluate the system.

1.2.5 Conclusion and Future work

We conclude the thesis by looking back at the research questions and show in what way they have been answered in this thesis appropriately.

Ultimately, not every aspect of the work described in this thesis is finished yet. This gives opportunity for a list of potential future topics which can be addressed at a later point. A discussion of these topics is given in the end.

Chapter 2

Background

Before presenting our own work it is important to put it into the scientific context which provided the foundation and background to it. Without this contextualization a claim of proper scientific work is hard to justify.

The thesis touches on several different topics and even different fields ranging from Linguistics and Computational Linguistics to Computer Science and Pedagogics. These include aspects of Latin linguistics, Computer-Assisted Language Learning and Human-Computer-Interaction.

This chapter provides background information to alternative methods of text input, introduces the field of Controlled Natural Languages, gives an overview of other formal or computational Latin resources and concludes with an overview over the development of Computer-Assisted Language Learning. To each of these areas we will relate concepts of our own work.

2.1 Text Input and Modification

Every time a user wants to input and modify text on a modern device they are confronted with methods that haven't changed much since the dominance of typewriters. One still uses the same kind of keyboard and the only major improvement is that it is possible to more seamlessly remove and change text. However, the cursor has to be moved to the correct position, then the parts to be changed have to be deleted character by character and finally the new content has to be typed.

Some drawbacks of the traditional keyboard are well known, mostly ergonomic factors (Ciobanu, 2014), which lead to problems reaching from minor discomfort to complete disability to use a device depending on the nature of the device or other impairments.

People with various health-related limitations can either use this method of text input only in a very limited way or not at all. This can be caused both by physical or mental impairments. Over the time different technologies have been developed to support these people, e.g. Braille keyboards and displays for blind people (Becker et al., 2004) or speech or gesture based methods (Felzer et al., 2014; Kumar et al., 2012; Ward et al., 2002).

Independently, the kind of devices we use to create and modify text changed significantly. Mobile phones became ubiquitous while at the same time they became smaller and more powerful. These facts already required radical changes in the way text can be created, starting with the T9 predictive text creation system used in earlier feature phones (King et al., 2000) to touch-based methods on current smart phones. These methods like the academic Dasher (Ward et al., 2002) or the commercial systems Swype (Kushler and Marsden, 2004) and Swiftkey improve the input speed by adding suggestion and post-correction features, but still suffer from some of the same drawbacks by using the typewriter-style keyboard on small touch-based devices.

Different approaches work on word level instead of character level. A simple approach is used in the MOLTO project (Ranta et al., 2012). Here the incremental Grammatical Framework parser is used to suggest only valid continuations, word by word, depending on a grammar. It provides useful support when creating a new text to guarantee that the text is within the language defined by the grammar.

The input systems on smart phones like Swype also provide word-based methods for post-correction, e.g. to fix typos.

The approach chosen by Ljunglöf (2011) is agnostic to the creation part and focuses only on the editing of already existing text. It uses grammars to provide syntax trees for the surface text and then maps editing operations on the surface to tree operations.

Text input is also a relevant aspect in language learning. Depending on the application the amount of text input can be minimized, e.g. by providing multiple choice for gap-filling. For larger scale translation exercises a more flexible method for text input and modification is necessary which has to be handled appropriately. Duolingo and others for example introduced the concept of a "word bank". A list of words is provided to the user which contains all words to form the correct sentence and additionally a set of distractors. This approach seems to have disadvantages because already the choice of vocabulary can give major hints for the translation and make it too easy. For our translation exercises we adopted Ljunglöf's method which seems particularly suitable in combination with our grammar-based approach to language learning. The implications of this choice will be discussed later in depth.

2.2 Controlled Natural Languages

The idea to work with small grammars that describe limited language fragments to increase the reliability is not a new idea. Already Richard Montague started formalizing language fragments that have well-defined semantics, even though his claims were much broader[1] (Montague, 1974). However, exactly this intentionally limited approach is the concept behind Controlled Natural Languages (CNLs), an active research area within computational linguistics.

A good definition of CNLs and overview of the field is given by Kuhn (2014). The definition is based on four criteria:

- Exactly one language
- Restrictions on lexicon, syntax and/or semantics
- Preserving most of the natural properties
- Explicitly constructed language

This definition describes a set of languages located in a spectrum ranging from purely formal languages on the one hand and full natural languages on the other.

Constructed Natural Languages have several fields of application, an overview of which Kuhn gives (ibid.). These include document authoring, automatic reasoning and translation, all of which are supported by the closeness to formal languages.

A rather underrepresented application is CNLs for language learning. There have been a few approaches like Basic English (Ogden, 1930), which predates the use of computers for language learning purposes, and provides a restricted

[1] *"I reject the contention that an important theoretical difference exists between formal and natural languages"* (Montague, 1974)

version of the English language to help people learn it as a foreign or second language. A more modern approach is ALCOGRAM (Adriaens and Schreors, 1992) which tried to combine document authoring and CALL and finally there is the master's thesis by Abolahrar (2011) that uses simple but multilingual phrase grammars for language learning in combination with the MOLTO input method.

The criteria for CNLs also mostly match the grammars we use in our work. That means we use restricted fragments of a language that seem natural. However, we add multilinguality and relax the constraint that the languages have to be explicitly constructed. Instead we use languages implicitly defined by text fragments in textbooks. This aspect will be discussed further in Section 3.5.

2.3 Latin Resources

Even though Latin has been in the interest of people including linguists for a very long time, it can still be considered an under-resourced or less-resourced language when the primary focus is on computational resources. There is lots of written material in the form of e.g. grammar books over the centuries from the classical antiquity to modern times, but there are not too many web pages in Latin which leads to the lack of a large multilingual web corpus including extensive Latin resources. However, one exception and a potentially surprising and valuable resource in this context is Wikipedia. The Latin version contains more than 100 000 pages which are linked to the corresponding articles in other languages (Vicipaedia, 2018).

Most of the older and traditional resources lack a degree of formality that is required to convert them into a representation that can be used in the context of computational linguistics. That even concerns the extensive work by Harm Pinkster (1984, 2015) on Latin syntax and semantics which, to explain the rather free Latin word order refers to pragmatic features such as topic and focus. But these elements are already for humans hard to identify.

On the other hand, some interest in Latin can be found in the area of formal syntax, already present in the work of Chomsky (1988) and in Combinatory Categorial Grammar by Steedman (2016). However, that usually never resulted in an extensive linguistic resource. Some work has also been done in adopting Chomsky's minimalist program for Latin (Sayeed and Szpakowicz, 2004) which led to a computational implementation which still only focused on plain syntax, ignoring the challenges of Latin morphology.

There are also some more specific and limited resources language resources for Latin. These include a Latin finite state morphology (Springmann et al., 2016) which is considered state of the art. Additionally effort has been put into the creation of Latin treebanks (Bamman and Crane, 2006; Gregory R. Crane, editor, 2018), some of which now are included in the Universal Dependencies project (Nivre et al., 2016). Especially the treebanks provide the material for a quantitative analysis of syntactic constructions which is in quite some detail covered by Bamman and Crane (2006).

We add to the list of available resources the Latin implementation of the Grammatical Framework Resource Grammar Library (see Subsection 3.1.2), which will be described in more detail in Chapter 4. It is part of a growing

Stage	1970s-1980s: Structural CALL	1980s-1990s: Communicative CALL	21st Century: Integrative CALL
Technology	Mainframe	PCs	Multimedia and Internet
English-Teaching Paradigm	Grammar Translation & Audio-Lingual	Communicate Language Teaching	Content-Based, ESP/EAP
View of Language	Structural (a formal structural system)	Cognitive (a mentally constructed system)	Socio-cognitive (developed in social interaction)
Principal Use of Computers	Drill and Practice	Communicative Exercises	Authentic Discourse
Principal Objective	Accuracy	Fluency	Agency

Table 2.1: Epochs of CALL from the 1960s to the 1990s according to Warschauer (2004)

collection of large natural language grammars that are freely available and provides an essential resource for the creation of application-specific grammars, precisely the type of grammars we need in our application.

2.4 Computer-Assisted Language Learning

The history of Computer-Assisted Language Learning (CALL) goes back almost as far as the history of modern computers, at least to the middle of the 20th century, some people even choose earlier dates (Computer History Museum, 2010). Its relevance increased in the same way as technology got more accessible, now with ubiquitous computing it has arrived in the mainstream.

It is almost impossible to give a full account of the history of CALL. An overview for the earlier periods of CALL ranging from the 60s to the 90s of the previous century is given by Levy (1997). For each of the three periods 60s/70s, 80s and 90s he presents major projects, their technological background and their influence on the field. Most of theses historic developments are described in a similar way by Warschauer (2004), whose classification of the CALL history can be found in Table 2.1. However, Bax (2003) opposes this classification by pointing out what he claims to be contradictions. That involves the separation into epochs even though most of the systems survived their own epochs in one way or another. Furthermore he opposes the classification of the pedagogic concepts in some epochs neglecting evidence given by Levy.

Beyond the description of the first three epochs additional projects are mentioned to sketch the more recent development in the field.

2.4.1 First Epoch: 1960-1979

The first of theses epochs coincides with the mainframe era of computer history which means that computers where not generally available but instead large mainframes could be found at special places like universities which could only be accessed remotely by dumb terminals. In this context two major projects were incepted, PLATO[2] and TICCIT[3]. However, large steps in the development

[2]Programmed Logic for Automatic Teaching Operations (a backronym)
[3]Time-Shared, Interactive, Computer-Controlled Information Television

of CALL were already taken in this earlier period together with a push in the necessary technology. This was possible due to large-scale funding and large team efforts in the development.

PLATO was initiated in 1960 to develop a computer system for educational purposes. The design goal was interactive, self-paced instructions for a large number of students. One of its first applications was to teach French, and it already provided many features of modern learning environments: high-resolution graphics, multimedia, and features for collaboration. Its use was not limited to language learning; Lessons and other applications could be developed in various ways, including using a high level authoring language called TUTOR.(Levy, 1997, pp. 15ff.)

The *TICCIT* system was started 1971 at Brigham Young University combining television technology and computers. One difference between PLATO and TICCIT is that the second one already from the start used exactly one authoring system which made use of a specific theory of teaching. This framework dictated the requirements on hardware and software. Even stronger than with PLATO, multimedia aspects were essential with TICCIT. Furthermore, the system gave more control to the learner: they could not only select the content but could also change the behavior of the system with dedicated keys for difficulty levels and other parameters. (ibid., pp. 18f.)

Both systems were evaluated by the Education Testing Service. While the PLATO system appealed to both teachers and students the system did not show any major positive or negative effect in learner outcome. TICCIT on the other hand showed large improvement in the student achievements. (ibid., pp. 20f.)

2.4.2 Second Epoch: 1980-1989

The second phase was dominated by the development of personal computing in contrast to mainframes. Home computers became available and were accessible to everyone. A special initiative in that area was the BBC Computer Literacy Project in the UK that led to the development of the BBC Micro, a home computer specifically designed and built for educational purposes. On the other hand powerful workstations with multimedia capabilities found their way into universities and were connected in large networks. These developments influenced CALL. Most of the home computers provided an implementation of BASIC in which simple CALL-programs could be developed and the interconnectedness allowed for increased relevance of Communicative Language Teaching (Savignon, 1987). Later more specific authoring systems like HyperCard for Apple Macintosh provided important tools for developing CALL solutions. Around this time both large-scale projects and projects by individual language teachers could be found. (ibid., pp. 22ff.)

The first of the two relevant projects of this decade was the *Storyboard* program, a typical example of the authoring systems of the 80s. The task associated with it was to reconstruct a text from hints like the title and context. Other than the previous projects it was more of a personal project of individual researchers and language teachers. It spread all over the microcomputer ecosystem and was ported to many different computer systems and programming languages. In this process it got extended and modified to become more user-

friendly and feature-rich. The text reconstruction task can be seen as one of the first generation CALL tasks together with gap-filling and text modification. It also provides context for communicative learning given that the texts are authentic. (ibid., pp. 24f.)

The second project again falls more in the category of large-scale projects. The *Athena Language Learning System* (ALLP) was part of Project Athena that started in 1983 at MIT to explore the use of computers in education. The ALLP itself was focusing on communicative approaches for language learning. Project Athena provided a large network of 450 workstation including modern multimedia workstations. Two major initiatives were included in the project, the MUSE authoring framework provided means to connect many different kinds of media with a hypertext-like system. More importantly, as part of ALLP the use of AI and NLP techniques for language learning was explored, which led to a game based on interactive video narratives. The goal was to teach vocabulary in context, exercise reading and listening comprehension, raise cultural awareness as well as practice communication strategies. (ibid., pp. 26ff.)

2.4.3 Third Epoch: 1990-1996

The major influence on the development of third generation CALL projects was the easier access to global networks, especially the internet with its use of hypertext to present information and fast ways of communication by email. (ibid., pp. 31ff.)

The *International Email Tandem Network* is a direct consequence of this development. It began in 1993 and promoted language learning by communication over the internet. The system was divided into subnets organized as mailing lists, which provide bilingual discussion forums where participants can ask for advice in either language. The system provided a high level of flexibility and independence in time management and teaching approaches. Instead of at a fixed time in class, the students practiced the language distributed all over the week. Also, the teachers and instructors would not just point out mistakes but instead could influence the learner's language use by positive example. This was supported by a more informal teacher-learner-relationship. (ibid., pp. 32f.)

The next to last project in the historical excursion by Levy is the *CAMILLE*[4] project that grew out of France InterActive. It combines a communicative competence approach with an interactive multimedia environment: it puts the concept of a learning environment in the center. Instead of focusing on methods of teaching it focuses on the learning process and gives more freedom to the learner. The system itself uses metaphors like the desktop to organize the learning. On a virtual desk the learner can organize various learning resources starting with an interactive textbook. Students are directed by lessons divided into units and modules. Exercises include classic drills but also quizes and role-playing tasks. In the context of France InterActive, a thorough model for the multimedia development process was presented. (ibid., pp. 34ff.)

The Oral Language Archive is the last project to be mentioned. It was initiated 1994 at Carnegie Mellon University. Its aim is to create a large collections of native speaker's sound recordings together with a learning infrastructure

[4]Computer-Aided Multimedia Interactive Language Learning

based on this resource. Furthermore, it provides integration into a broad range of authoring systems including Hypercard. (ibid., pp. 37ff.)

2.4.4 Fourth Epoch: after 1996

Levy's book was published in 1997 which makes its description of the historical context of CALL stop around that time. But that of course does not mean that the development of CALL stopped, more the opposite. Some important factors in the most recent period of CALL include crowd-sourcing and a growing interest in commercial applications that target a general audience as well as specialized systems for usage in a very specific course.

A look both at the recent publications in CALL-related conferences and workshops like NLP4CALL,[5] BEA,[6] NLPTEA[7] and more, as well as at the popular apps in the app stores, can give a picture of the current CALL landscape. Besides full-fledged language learning systems like ours, a lot of current research is in specific sub-problems like learner modeling, automatic essay grading, and so on.

One of the most popular apps, which now is a commercial system that started with a clear crowd-sourcing approach, is *Duolingo* (Garcia, 2013). Furthermore, a variety of specific systems for different languages and language concepts using different technologies developed in recent years:

- Morphology training with finite state methods (Kaya and Eryiğit, 2015)
- Linguistic resources like annotated data or semantic resources with rule-based algorithms (Michaud, 2008; Moritz et al., 2016; Redkar et al., 2017)
- Linguistic resources combined with both machine learning and rule-base approaches (Volodina et al., 2014)
- Crowd-sourcing in combination with machine learning (Horie, 2017)

In this context we need to place our work. It makes no sense to deny the influence of Duolingo, still we aim in a different direction. Also it should be obvious that this work is not done in a large team effort or a big commercial setup. Superficially, our system seems like other systems that are built for this very specific use case of teaching historic languages, for example "King Alfred" (Michaud, 2008), but for teaching Latin.

However, our general approach is very flexible for reasons that will be explained in the next Chapter, which puts it close to general authoring systems. Even though some properties of MULLE are fixed, a lot of flexibility in the creation of lessons and exercises is possible in the way the grammars are designed.

[5] Natural Language Processing for Computer-Assisted Language Learning
`https://spraakbanken.gu.se/eng/research/icall/nlp4call`
[6] Workshop on Innovative Use of NLP for Building Educational Applications
[7] Workshop on Natural Language Processing Techniques for Educational Applications

Chapter 3

Grammar-based Language learning

The central part of this thesis is the development of a general and flexible framework for language learning by providing access to different types of exercises. For that we adopt the general method for grammar-based text modification on the word level to a general framework for language learning suitable for different settings including the traditional classroom setting.

This chapter gives an insight into the general mechanics behind the MULLE system. We first present the necessary background by introducing the grammar formalism which we use in this approach. Building on this formalism we introduce the method for text modification on word level supported by grammars as it was suggested by Ljunglöf (2011). These two sections are mostly independent from the topic of language learning. This topic becomes relevant in the following two sections: first we describe how to create grammar files suitable for our ideas for CALL, and building on that we present methods to create exercises from grammars. The chapter is concluded with a reflection of the properties of both the grammars and the pedagogical approach to language learning inherent in our framework.

3.1 Grammatical Framework

Before we can more precisely describe our grammar-based approach to language learning, we need to introduce the grammar formalism we use to formalize the grammars which both encode the learning objectives and control the user input.

This grammar formalism, called Grammatical Framework (or GF in short), is a special-purpose programming language with a syntax similar to the functional programming language Haskell. It provides several advanced features which either help the development of grammars or are of theoretical interest. It has also been shown that this grammar formalism's expressivity is equivalent to parallel multiple context-free grammars (PMCFG), a mildly context-sensitive[1] grammar formalism (Ljunglöf, 2004).

3.1.1 Grammar Formalism

A grammar in GF is separated into one abstract syntax and one or more concrete syntaxes. The abstract syntax defines a general framework of syntax rules that defines what atomic components exist and how more complex components can be assembled from smaller parts, without going into the detail about the shape of these components. In that way it defines a grammar similar to traditional context-free grammars without terminals. The concrete syntax in turn defines how a specific language can be modeled concretely. By sharing the same abstract syntax across several concrete syntaxes it is easy to provide multilinguality to applications by using transfer-based translation via the abstract syntax trees. For that the concrete strings are parsed to abstract syntax trees and then linearized, i.e. converted back from an abstract syntax tree into a concrete string, in another concrete language.

[1] Ljunglöf shows that it is even more expressive than mildly context-sensitive but still parsable in polynomial time

3.1.2 The Resource Grammar Library

The separation into abstract and concrete syntax provides the foundation for the Resource Grammar Library (RGL) (Ranta, 2009), a large abstract syntax and a set of concrete syntaxes of various languages implementing this common interface. The RGL is both a collection of more than 30 language grammars and a common interface to use these languages in multilingual applications. These grammars can be used as support for building specific applications, like the one we want to describe here, by providing a higher-level abstraction to the underlying languages.

3.1.3 Simple Example

```
1  abstract Chomsky = {
2    cat
3      S ; NP ; VP ; N ; V ; A ; Adv ;
4    fun
5      mkS : NP -> VP -> S ;
6      mkNP : N -> NP ;
7      adjNP : A -> NP -> NP ;
8      mkVP : V -> VP ;
9      advVP : VP -> Adv -> VP ;
10     colorless_A : A ;
11     green_A : A ;
12     ideas_N : N ;
13     sleep_V : V ;
14     furiously_Adv : Adv ;
15 }
```

Listing 3.1: An example for an abstract syntax in GF to cover the sentence *Colorless green ideas sleep furiously*

A first example of a GF grammar can be seen in Listing 3.1 and Listing 3.2. The Listing 3.1 shows a simple abstract syntax with seven terminal and non-terminal categories, five lexical rules and five phrasal rules. The first three categories `S`, `NP` and `VP` stand for the usual syntactic categories for sentence, noun phrase and verb phrase while the other four categories `N`, `V` and `A` and `Adv` stand for the lexical categories of nouns, verbs, adjectives and adverbs.

The rules in lines 10 to 14 introduce new lexical items with the names `colorless_A`, `green_A`, `ideas_N`, `sleep_N` and `furiously_V` with the corresponding categories for adjectives, nouns, verbs and adverbs. Note that GF itself does not make a formal distinction between lexical and phrasal rules. The only difference is the number of categories on the right-hand side. If it is just one category, or in the concrete syntax a function without a parameter, it is called a lexical rule and otherwise a phrasal rule (Ranta, 2011, pp. 98).

The remaining rules in lines 5 to 9 define how noun phrases can be created from nouns (`mkNP`), that verbs can be used as verb phrases (`mkVP`) and that a noun phrase and a verb phrase together can form a sentence (`mkS`). Additionally, adjectives can modify noun phrases (`adjNP`) and adverbs can modify verb phrases (`advVP`). As the reader might already have expected, the grammar can recognize the Chomskyan sentence *colorless green ideas sleep furiously* (Chomsky, 1957, p. 15) and variations if it.

```
1  concrete ChomskyEng of Chomsky = {
2    lincat S,NP,VP,N,V,A,Adv = Str;
3    lin
4      mkS np vp = np ++ vp ;
5      mkNP n = n ;
6      adjNP adj np = adj ++ np ;
7      mkVP v = v ;
8      advVP vp adv = vp ++ adv ;
9      colorless_A = "colorless" ;
10     green_A = "green" ;
11     ideas_N = "ideas" ;
12     sleep_V = "sleep" ;
13     furiously_Adv = "furiously" ;
14 }
```

Listing 3.2: The English concrete syntax for the abstract syntax given in Listing 3.1

In the concrete version for English in Listing 3.2 we assign concrete values to the definitions from the abstract syntax in Listing 3.1 (line 1). In line 2 we define that all categories are represented by strings. In the lines 9 to 13 we just give the string literals for the lexical items.

Slightly different are the other three phrasal rules. In the context of abstract syntax we talked about rules, in the concrete syntax the corresponding concrete versions are usually called linearization functions, in a similar sense as functions in functional programming. They can be seen as a kind of mathematical functions that take the components as parameters and based on these compute new value. The simplest cases are the two functions `mkNP` and `mkVP` in line 5 and line 7 respectively. They seem similar to the identity function and to do nothing specific at all except for returning the parameter unchanged. But internally the category is changed from `N` to `NP` and from `V` to `VP` to make them usable in the respective context[2]. Not a `V`, only a `VP` can be modified by an adverb with the rule `advVP`. Slightly more complex are the functions like `adjNP`, `advVP` and `mkS` which combine several parts to a new one. For example `adjNP` modifies a noun phrase with an adjective, that means it takes an adjective and a noun phrase as parameters and produces a new noun phrase. The order of the parameters and their types are defined in the abstract syntax. In the concrete syntax we can assign arbitrary names to them, but it is good practice to use reasonable names like `adj` for the adjective and `np` for the noun phrase. Since in this simple example all values are just strings, we can concatenate them with the `++` operator to form a new string. The same happens for `advVP` and `mkS`.

3.1.4 Extended Example

To show more of the power of GF we extend the abstract syntax from Listing 3.1 to the one in Listing 3.3. Previously we had just one form for each of the lexical entries. Now we add inflected word forms. This also requires that we enforce agreement between different parts of the sentence. Furthermore, we add a

[2]Usually called a type coercion

```
1  abstract ChomskyExt = {
2    cat
3      S ; NP ; VP ; PN ; Det ; N ; V ; V2 ; A ; Adv ; Pron ;
4    fun
5      mkS : NP -> VP -> S ;
6      mkNP : Det -> N -> NP ;
7      mkNP2 : PN -> NP ;
8      mkNP3 : Pron -> NP ;
9      adjN : A -> N -> N ;
10     mkVP : V -> VP ;
11     mkVP2 : V2 -> NP -> VP ;
12     advVP : VP -> Adv -> VP ;
13     aSg_Det : Det ;
14     aPl_Det : Det ;
15     theSg_Det : Det ;
16     thePl_Det : Det ;
17     Chomsky_PN : PN ;
18     colorless_A : A ;
19     green_A : A ;
20     idea_N : N ;
21     sleep_V : V ;
22     furiously_Adv : Adv ;
23     love_V2 : V2 ;
24     I_Pron : Pron ;
25 }
```

Listing 3.3: Extended abstract syntax based on Listing 3.1 to demonstrate word inflection and agreement

proper name, definite and indefinite articles, a transitive verb and a pronoun.

GF differs from other grammar formalisms insofar that it has a strong distinction between types and values of these types. That means so far that we have the string type **`Str`** and string values as literals surrounded by quotation marks. But we can extend this notion by defining new complex types like tables and records which then can be used to specify values of these types. Tables usually give rise to inflection tables while records enable us to store several values separately, which can be used to store inherent grammatical features alongside the plain string values. For the grammatical features we can define finite parameter types, i.e. types that are defined by listing all their values.

We start by defining the types again. First we need to define the parameters in line 2–3 along which we want to inflect word forms by giving all possible values. We need a **`Number`** feature for noun inflection and **`Number`** together with **`Person`** for verb forms. This is also encoded in the types we define in lines 4 to 10. Even for still simple cases like **`S`**, **`A`** and **`Adv`** we now have **`{s : Str}`**[3] instead of just `Str`. These types are now so-called records in GF. A record is a collection of values, called record fields, of potentially various types, each of which is assigned a label to access this value. So in this case we only have one record field with the label `s` of type `Str`. It is common practice in GF to use record types and values for all categories, each containing a `s` field for the main string representation. We can also use records to store additional features like inherent properties. In our example we use it to store an inherent **number** feature in determiners, proper names and noun phrases (lines 5 and

[3]Record types and values in GF are surrounded by squiggly brackets

```
concrete ChomskyExtEng of ChomskyExt = {
  param Number = Sg | Pl ;
        Person = P1 | P3 ;
  lincat S = { s : Str } ;
         Det = { s : Str ; n : Number };
         N = { s : Number => Str } ;
         PN = { s : Str ; n : Number } ;
         Pron, NP = { s : Str ; n : Number ; p : Person } ;
         V, V2 = { s : Person => Number => Str } ;
         VP = { s : Person => Number => Str };
         A, Adv = { s : Str } ;
  lin
    mkS np vp = { s = np.s ++ vp.s ! np.p ! np.n } ;
    mkNP det n = { s = det.s ++ n.s ! det.n ; n = det.n ; p = P3 } ;
    mkNP2 pn = { s = pn.s ; n = pn.n ; p = P3 } ;
    mkNP3 pron = pron ;
    adjN adj n = { s = table { num => adj.s ++ n.s ! num }} ;
    mkVP v = v ;
    mkVP2 v2 np = { s = \\p,n => v2.s ! p ! n ++ np.s } ;
    advVP vp adv = { s = \\p,n => vp.s ! p ! n ++ adv.s } ;
    aSg_Det = { s = "a" ; n = Sg } ;
    aPl_Det = { s = "" ; n = Pl } ;
    theSg_Det = { s = "the" ; n = Sg } ;
    thePl_Det = { s = "the" ; n = Pl } ;
    Chomsky_PN = { s = "Chomsky" ; n = Sg } ;
    colorless_A = { s = "colorless" } ;
    green_A = { s = "green" } ;
    idea_N = { s = table { Sg => "idea" ; Pl => "ideas" }};
    sleep_V = { s = table { P1 => table { Sg => "sleep" ;
                                          Pl => "sleep" } ;
                            P3 => table { Sg => "sleeps" ;
                                          Pl => "sleep" }}} ;
    love_V2 = { s = table { P1 => table { _ => "love" } ;
                            P3 => table { Sg => "loves" ;
                                          Pl => "love" }}} ;
    furiously_Adv = { s = "furiously" } ;
    I_Pron = { s = "I" ; n = Sg ; p = P1 } ;
}
```

Listing 3.4: Concrete syntax for the abstract syntax in Listing 3.3 to demonstrate word inflection and agreement

7). Noun phrases and pronouns have the additional inherent feature of **person** which is stored in a record field `p`. The remaining categories `N`, `V`, `V2` and `VP` can be inflected, which means we define them as tables from the features they are inflected by to the result type. Nouns are in this small example only inflected by `Number`, which results in a table from `Number` to `Str`, or in GF code `Number => Str` (line 6). All verb categories and the verb phrases derived from them are inflected both by **person** and **number** which results in the table of tables `Person => Number => Str`.

Now that we defined all types we can define appropriate values of these types. For adverbs and adjectives this is straightforward, for each record field, here just `s` we need to assign a value (line 26, 27 and 36). For proper names and determiners the approach is the same but we also have to give a value for the `n` field containing the inherent number (lines 21–25). The indefinite article in plural usually does not appear on the surface. Instead we define it as the empty string. For pronouns we just add the record field `p` for the inherent **person** feature.

For the inflected categories we need to define inflection tables, for nouns only inflected by **number**, for verbs inflected by **person** and **number**. Table values start with the `table` keyword followed by a list of mappings (`=>`) from parameter values to result values. In line 26 for example the `Sg` parameter value is mapped to the string *"idea"* and `Pl` is mapped to *"ideas"*. The mapping has to be total, i.e. there must be a mapping for each possible value of the parameter. For verbs we have tables of tables, the value of the first level of mapping is again a table (line 29–33 and 33–45). As we can see in the case of `sleep_V` we can have the same forms several times in a table. We can avoid some of these redundancies by using a wildcard (`_`) matching all remaining parameter values, for example in line 33 where the **number** does not matter for verb forms in ***first person***.

After defining all the lexical rules we have to give the concrete versions of the phrasal rules. Here again we combine the parameters to form new values but this time we have to make sure that we enforce the right kind of agreement. The simple cases are now still `mkVP` but also `mkNP3` which converts a pronoun into a noun phrase. They again look like the identity function. The rule `mkNP2` which turns a proper name into a noun phrase just copies all the information from the noun and adds the record field for **person** and sets it to ***third person***. With rule `mkNP` in line 14 we have the first case where we have to enforce agreement when combining a determiner and a noun to a noun phrase. The noun is inflected by number (it contains a table from `Number` to `Str`) and the determiner has an inherent **number** feature (a record field for `Number`. We can select the **number** value from the record in the determiner by using record-projection (`.`) and use its value to select the right form from the table in the noun using table selection (`!`). The resulting string can be combined with the string from the `s` field of the determiner to form the string for the noun phrase. To complete the value for the noun phrase we copy the **number** value from the determiner and again add a ***third person*** value to the `p` record field. The same method is used in `mkS` to form sentences from a noun phrase and a verb phrase. Here the **person** and **number** feature is used to select the correct verb from. Finally we have to have a look at the two rules `ajdN` and `advVP`. These two rules modify nouns and verb phrases respectively

by adding an adjective or an adverb to an inflected word. At this point we do not know the correct word form yet, but we can use tables in one more way to solve this problem. Instead of a parameter value we can also use a variable name (line 17) which will later be bound to the concrete value and that way we can pass on information through the tree. A short form for this which even works for several levels using several variables is the \\ operator in lines 19 and 20. The expression `\\x => ...` is syntactic sugar for `table { x => ... }`.

Covering the whole syntax of GF is not feasible in the scope of this thesis so the interested reader should be referred to the book about GF (Ranta, 2011). The main points that are relevant for the rest of the thesis are the separation into abstract and concrete syntax that easily give rise to multilinguality, especially in combination with the RGL. Also important is the extensions of the context-free core with records and tables to provide means to store separate pieces of information, select values from a set of alternatives and pass information through the syntax tree, which contributes to the expressive power of GF.

3.2 From Word-based Text Editing to Translation Exercises

Ljunglöf (2011) suggested a novel method to edit texts on the word level using grammars in the background by mapping text modification on the surface onto operations on the underlying syntax trees. Even though other grammar formalisms could be used as well, by using GF, as described in the previous section, we get advantages like immediate multilinguality by using language independent abstract syntax trees. How this method can be used in language learning will be explained in the course of this section.

Language learning usually involves two different languages, one language that is taught and the other language that is used for teaching and to describe the learning outcome. Following a tradition in logic and linguistic we call the first language the object language and the second one the meta language.

One traditionally important part of learning a language is to translate between these two languages. For that reason we want to focus, among other potential exercise types, on translation exercises in our application. But instead of presenting the learner with a sentence in one language and expecting them to come up with a complete and free translation on their own, we offer the learner two sentences in the different languages and the student's task is to modify one of the sentences to make it a proper translation of the other sentence. To accomplish this task, the student uses the method that will be described now.

3.2.1 Word-based Text Modification

The method presented by Ljunglöf (2011) does not, like most other methods to modify text, work directly on the surface string, but instead provides a method to map between text editing operations on the surface and tree modification on the underlying abstract syntax tree.

The user interacts only with the surface representation by clicking on positions in the string and choosing replacements of selected parts from a menu.

The clicks are translated into node positions in the tree which then affect both the selection of a substring and the list of potential changes.

To give a concrete example we can revisit variations of the Chomskyan sentence *Colorless green ideas sleep furiously* with its syntax tree (Figure 3.1). We already showed the abstract syntax in Listing 3.1 in the previous section. Let us assume that the user selects the circled `NP` node covering the phrase *green ideas*.[4]. Then the system uses the grammar to generate a list of trees with the same category `NP` (Figure 3.2). The syntax rules that can result in `NP` are the two rules in lines 5 and 6 of Listing 3.1. As already this small example shows, the list can be infinite, so some filtering has to be applied and the final, finite, list of trees is used to suggest editing operations on the string to the user. The first tree results in the string *ideas* and is not containing any adjective which results in the deletion of the adjective that was present previously, the second tree (*colorless ideas*) contains a different adjective which results in a substitution of the corresponding substring and the third tree (*colorless green ideas*) contains an additional adjective which results in the insertion of this additional adjective. By using the grammar the system only suggests changes that lead to syntactically correct sentences. We will discuss in Section 3.6 what influence that can have on language learning.

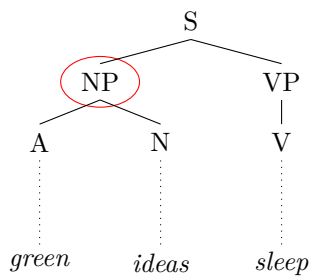


Figure 3.1: The syntax tree for the sentence *green ideas sleep* according to the abstract syntax in Listing 3.1 with the `NP` node selected by the user and highlighted

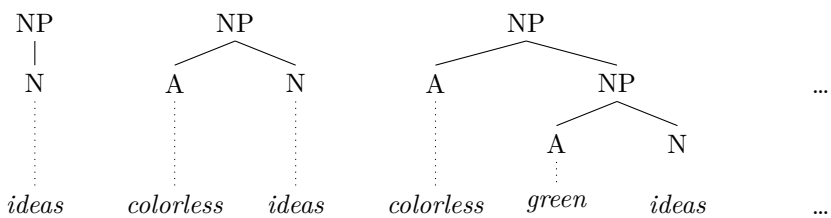


Figure 3.2: List of potential subtrees with root category `NP` that can be generated with the abstract syntax in Listing 3.1 and be used to replace the subtree selected in Figure 3.1

[4]Details how this selection works will be covered in Section 5.1

3.2.2 Translation Exercises

To extend this to a method suitable for language learning, we need two syntax trees, which then give rise to two sentences. By using multilingual grammars we can have one sentence in the meta and and one in the object language by using different concrete syntaxes of the same abstract syntax. The user can then use the method of word-based text modification to change one of the trees to make it match up with the other to solve translation exercises.

How this method works specifically in the language learning context we will show at the concrete example of our MULLE Latin app in Section 5.1 of Chapter 5.

3.3 Textbook Lessons

Now that we have discussed how we can use grammars to direct the user interaction and sketched a method for translation exercises given two abstract syntax trees, the next step is to explain how we can extract suitable grammars for teaching a language from available resources. The main type of resources we want to focus on are traditional textbooks. However, that does not mean that the general ideas cannot be used in various other settings.

3.3.1 Lesson Structure

Text books that are used in language teaching in a traditional classroom setting (e.g. Ehrling (2015); Lindauer et al. (2000)) are divided into several lessons to slowly increase the vocabulary and syntactic complexity covered, to guide the learner to an extensive understanding of the language. These lessons are usually divided into a text fragment (see Figure 3.3 for a sample fragment from Ehrling (2015)), a vocabulary list and explanations of the grammar as well as exercises to be solved on paper by the student.

```
Prima scripta Latina

[...] Imperium imperatorem habet. Imperator imperium tenet.
Caesar Augustus imperator Romanus est. Imperium Romanum tenet.
Multas  civitates externas vincit. Saepe civitates victae
provinciae deveniunt. [...]
```

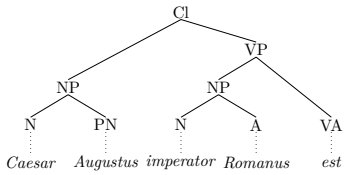
Figure 3.3: An excerpt from the first lesson in a textbook for beginner's learners of Latin on university level (Ehrling, 2015) showing short, simple sentences

We adopt the same structure for our application by creating one grammar for each lesson. To cover the same content as well as keep up the familiarity we use both the vocabulary list as well as the lesson texts to create our grammars.

3.3.2 The Grammar Extraction Process

The process to create a lesson grammar from a textbook lesson consists of three steps. These steps should be automated as much as possible, but at the

Caesar Augustus imperator Romanus est.




```
[...]
mkNP1 : N -> PN -> NP ;
mkNP2 : A -> N -> NP ;
mkVP  : VA -> NP -> VP ;
mkCl  : NP -> VP -> Cl ;
mkS   : Cl -> S ;
[...]
```

Figure 3.4: The steps to derive a grammar from a sentence 1. analyze the sentence (eng. *Caesar Augustus is the Roman emperor*), 2. create a syntax tree to cover the sentence, 3. derive a grammar from the syntax tree.

moment require some human intervention. The steps are the following:

1. Adapt a lexicon from the textbook lesson, which usually is given as an explicit vocabulary list. Some of the vocabulary is already covered in the lexical resources available for GF in the language we want to handle. Otherwise the morphological grammar rules using smart paradigms [5] can be used to extend the lexical resources.

2. Create syntax trees for all sentences in the text. This can be done manually or semi-automatically by parsing the sentences with the full RGL extended with the new lexicon. In case of several analyses the correct, i.e. desired, analysis has to be selected manually.

3. Create a grammar describing precisely the trees from the previous steps. For that the rules can be read off the inner nodes of the trees. Usually this grammar will be over-generating. Several methods can be used to reduce the grammar, e.g. by merging several rules to one.

An example of this process can be seen in Figure 3.4. The last two steps are simplified a lot by having access to the RGL. Especially in the last step, most of the rules needed in a grammar are usually already covered by the RGL and can be used without any additional effort. Sometimes minor adjustment between the general case handled in the RGL and a more specific case in the text are necessary. Thanks to the abstraction level of the RGL and modularity supported by GF the parts that are covered by the RGL can easily be shared across languages. That makes it easy to add additional languages to a lesson or exchange one language for another because the common core stays the same and only the manually overridden, language-specific parts have to be replaced.

The grammar which we get as a result from this process can be used in our application to provide the resources necessary to generate translation exercises based on the content of a syllabus and in the context of a language course. In the end each lesson is covered by one grammar which is specific to exactly the same vocabulary and syntactic complexity of this lesson. This means that the content of the exercises generated from this lesson should already be familiar to a student from the classroom. How exactly exercises can be generated from a given lesson grammar is the topic of the following section.

[5] Smart paradigms are covered in detail in Chapter 4

3.4 Exercise Generation

After the introduction of the method used for user interaction as well as the method to derive lesson-specific grammars from an existing textbook, this section focuses on how to generate proper exercises given a set of lesson grammars.

Our main interest in exercises so far have been translation exercises because they are straightforward to implement in our system and are the most common type of exercises in the concrete use case we looked at. We want to group the exercises we provide in our system into lessons. We already pointed out that for each lesson in the textbook we can create a lesson-specific grammar file. We extend this notion of a lesson to a lesson in our framework which consists of a lesson grammar and a set of exercises using the same grammar.

Each exercise consists of two abstract syntax trees which are valid within the lesson grammar. Depending on the similarity between the two trees the exercise can be easier or more difficult to solve. This gives control over the way how the students are improving their language skills. To define a lesson we have to generate a set of exercises.

Bootstrapping for the exercises in a lesson can be done manually or be partially automated by randomly generating valid trees for a grammar. To guarantee for a high level of reliability, human expertise is required at the moment. For example for each lesson it the difficulty level has to be decided which can be controlled by selecting reasonable, i.e. not too different, trees for each exercise.

The trees should not be completely unrelated, otherwise it would be too difficult and tedious to solve the task, especially in the beginning. So both trees should have a certain level of similarity. Even similar trees can be different on several levels: on the lexical level requiring to change certain aspects of the used words, or the syntactic level requiring to add, remove or substitute larger constituents or even to modify higher-level aspects of the sentence like polarity, tense, aspect, etc.

This gives some flexibility in designing the lessons from the teacher's point of view but adds complexity when trying to automatically generate suitable exercises. The method is even so general that the whole system can be seen as agnostic about both the languages to be used and the teaching approach. As long as it is provided with a grammar for each lesson which is multilingual with at least two different languages and a set of exercises is provided or generated where each exercise consists of two valid syntax trees and each syntax tree one of the languages in the grammar is assigned, the system can provide automatic translation exercises to the student.

Additional exercise types have been discussed but mostly remain for future work. The simplest possible addition would be exercises to practice vocabulary knowledge by using trees that are only different on the lexical level. One of the other exercise types could be an additional level of morphology training. By default grammars in GF take care of choosing the correct word forms and enforce agreement between parts of the sentence. These agreement features can be relaxed to also allow exercises where the user has to choose the correct word forms instead of relying on the system to provide the correct forms for them. Other exercise types could contain additional media like images or sound.

3.5 Suitable Grammars

The grammars that are intended to be used in this framework require certain properties. Some of these properties are determined by the method used for modifying the sentences and other properties are inherent of the grammars that can be derived from the textbook lessons.

The grammars derived from a textbook in the way we presented have the following properties:

- limited vocabulary, given in the explicit vocabulary list in the book
- small set of syntax rules, defined by the text fragment given in the lesson
- implicitly defined syntactic complexity, defined by the lesson's learning outcome

For the editing method it is relevant that the size of the grammar is limited so that the suggestions can be presented in a clear way. Furthermore, it might be necessary to have a surface representation of all information necessary for the user to solve the task. That means that even if some grammatical features are usually not expressed on the surface they have to be accessible on the surface in our system. In case of Latin that would be for example articles in general or pronouns in subject position. It is not yet fully clear if that is a general requirement or should only be an option in the beginning to help people learn these potentially unusual features like pronoun dropping.

These properties place the suitable grammars close to the concept of Controlled Natural Languages. A CNL is a language based on a specific natural language preserving most of the natural properties while restricting its vocabulary, syntax and/or semantics. It also has to be specifically constructed for a specific purpose (Kuhn, 2014). So CNLs can be seen as languages that are placed on the scale ranging from formal languages to full natural languages. This sounds similar to the result we get from extracting grammars from textbooks. The only concern could be the focus on explicit definitions of languages in CNLs while our lesson grammars can be seen as being defined implicitly by the text fragment. Still the concepts seem strongly related.

CNLs can be classified according to various criteria, including the intended application of the language (translation, reasoning, etc.) and the context of its creation (academic, industrial, etc.). A more relevant classification is the PENS classification that places a language in a space defined by the four dimensions Precision, Expressiveness, Naturalness and Simplicity. Each dimension is assigned a scale from 1 to 5 with 5 being the highest level possible. (Kuhn, 2014).

Considering the restrictions identified for usable grammars and the general structure of the grammar used in MULLE, this results in the PENS classification $P^4E^-N^4S^4$:

Deterministically interpretable (P^4): The grammars are fully formalized in a computational grammar formalism. Each sentence can be mapped to a finite set of abstract syntax trees

Languages with natural sentence (N^4): The sentences created by the grammar are syntactically correct according to the RGL

Languages with short description (S^4): The languages are formalized in compact grammars with limited access to external resources like the RGL and additional lexica

No classification (E^-): No formal representation besides the abstract syntax trees is used so the dimension of expressivity is not relevant.

Besides the CNL classification, we can also look at how grammars are classified in the GF ecosystem. There is a distinction between resource grammars and application grammars. The main difference is that resource grammars just describe the syntax of a language without any semantic considerations while application grammars are used for a specific application and have a strong semantic focus suitable for the intended domain. Also resource grammars aim for wider coverage than application grammars.

The grammars that are used in our system can be seen as a hybrid. They have a strong focus on syntax, but rely, where possible, on a separate resource grammar and only need to cover a small fragment of a language. They also seem to require at least some focus on semantics. The sentence originally presented by Chomsky talking about "colorless green ideas" is considered grammatically correct but does not make sense semantically. We argue that this lack of semantic coherence can be an obstacle when learning languages and for that reason it would be useful to introduce semantic restrictions. From a purely syntactic point of view, adjectives can be combined with any noun, but language use only allows certain combinations depending on features of lexical semantics. These restrictions can be added by including semantic knowledge, for example by including FrameNet-style semantics in the grammars Gruzitis and Dannélls (2017).

3.6 Teaching Concept

Before we conclude this section we still have to answer the question about the concept we intend for language teaching and learning with our application. We will describe a language learning application in Chapter 5 which has a very specific use case: It is intended to be used in combination with the traditional classroom setting. That means that a large part of language learning does not happen with our application but in the classroom and our application provides drop-in replacement for exercises traditionally solved on paper. But that only partially answers the question.

The language learning system has certain properties which shape the learning experience:

positive feedback There is no way to provoke negative feedback. The learner can try until they succeed and the system can help to guide them in the right direction.

implicit learning The learning within our application is purely implicit, the explicit part of language learning has to be provided externally, e.g. in a traditional language class. The learner also learns mostly by imitating constructions presented to them.

That means the intended audience is currently not an independent language learner but a student in a separate explicate language learning context. This context provides them with the knowledge about the necessary vocabulary for the lessons and general syntax rules. However, our system provides the student with a flexible playground in which they can explore features of the language without facing frustration from negative feedback.

Chapter 4

The Latin Resource Grammar Library

One of the foundations we built our final language learning application on is an extensive language resource in the form of a computational grammar for the Latin language. Work on this grammar already started before the current project but due to new and future requirements, extensions have been made, and work will continue as long as necessary. For that reason we have decided to include a description of this grammar including extensions made in the more recent past.

We first explain the different components of the grammar, how they are motivated and how they can be implemented in GF. That includes a description of a functional morphology for Latin, an example lexicon and a set of syntactic rules. This is followed by a discussion of current extensions and ways of evaluating the quality and coverage of the grammar.

4.1 Grammar Components

The grammar which is subject to description here is part of the Resource Grammar Library distribution. It follows the the concept of the RFL by implementing the common interface as the other languages, still the process of creating a concrete implementation for a language in the RGL can vary a lot depending on the language and certain decisions about the design approach (Ranta, 2011, p. 224).

The process described here is mostly influenced by two factors, the structure of the RGL abstract syntax and the structure of a generally available grammar book, more specifically the school grammar book by Bayer and Lindauer (1994). There is quite some overlap between the two structures, still there are some general differences. The RGL defines a set of morphological rules to derive full paradigms from a base-form lexicon, a small lexicon and general syntax rules. The Latin grammar book focuses mostly on morphology and some syntactic information, but does not cover any lexicographic information. For that reason we will postpone the discussion of the lexicon until after the description of the morphology.

4.2 Morphology

Word class	Inherent	Parametric	No. of Inflection classes
Noun	Gender	Number, Case	5
Adjective		Degree, Gender, Number, Case	3
Verb (active)		Anteriority, Tense, Number, Person	4 regular, 4 deponent
Determiner	Number	Gender, Case	

Table 4.1: Inherent and parametric features relevant for lexical categories

Latin is a strongly inflecting language, which means it belongs to the class of synthetic languages that express many syntactic features by using morphemes. Other than in agglutinative languages, another family of synthetic languages, where every feature is expressed in a separate morpheme, inflecting languages can encode several features in one suffix morpheme (Payne, 1997, pp. 27). For example in the Latin verb form *audio* (eng. *I listen*) *audi-* forms the word stem

Feature	Values
Gender	Feminine, Masculine, Neuter
Number	Singular, Plural
Case	Nominative, Genitive, Dative, Accusative, Ablative, Vocative
Degree	Positive, Comparative, Superlative
Anteriority	Anterior, Simultaneous
Tense	Present Indicative, Present Subjunctive, Imperfect Indicative, Imperfect Subjunctive, Future Indicative, Future Subjunctive
Person	1, 2, 3

Table 4.2: Domains of the finite features relevant for the Latin language

and the suffix *-o* encodes ***present tense***, ***indicative***, ***active***, ***first person***, ***singular***. The suffix *-remus* instead indicates ***imperfect***, ***subjunctive***, ***active***, ***second person***, ***plural*** where *-re-* encodes the **tense** and **mood** and *-mus* the **number** and **person**.

As a consequence, it means that many of the Latin word classes inflect along many different grammatical features. An overview of the different features for each word class can be seen in Table 4.1 and Table 4.2 gives for each feature, or parameter type, the list of all possible values. As we can see from the table, nouns are inflected by **case** and **number** and there are six values for **case** and two values for **number** which gives us $6 \times 2 = 12$ noun forms. Other word classes like verbs can appear in even more forms.

This fact confronts us with the challenge of dealing with this morphological complexity. Fortunately, Latin is a very regular language, for each lexical category there exist a set of inflection classes, i.e. collections of words of a word class that inflect in a similar way, covering the majority of the vocabulary. These classes can be formalized and applied in a general way that is called smart paradigm in GF (Détrez and Ranta, 2012).

Smart paradigms are functions that reconstruct as much morphological information as possible, that means that we want to have as little information as necessary in our lexicon from which we then generate the whole paradigm. Ideally we can use just one word form to extract all other forms as well as all other relevant grammatical information. The result of a smart paradigm is a GF table that encodes the information from Table 4.1 and Table 4.2 for the word class in question. An example for a smart paradigm for nouns that only requires one word form can be seen in Listing 4.2. Smart paradigms are one of the applications of pattern matching in GF, in this case pattern matching on strings (Ranta, 2011, p. 282f.).

4.2.1 Noun Inflection

Nouns are inflected by **number** and **case** and have an inherent **gender** which is reflected in the linearization type in GF (Listing 4.1).

For nouns we have five different inflection classes, in Latin linguistics usually called declension classes (from lat. *declinare* – eng. *to bend*), four of which are mostly regular and one class that contains all the irregular nouns.

For the nouns of the first two classes we can extrapolate the whole paradigm as well as the noun gender from just one noun form, the ***nominative singular***

```
param
  Number      = Sg  | Pl ;
  Case        = Nom | Gen  | Dat | Acc | Abl | Voc ;
  Gender      = Fem | Masc | Neutr ;
lincat
  N = {s : Number => Case => Str ; g : Gender }
```

Listing 4.1: Linearization type for nouns together with all parametric features required

```
oper
  noun : Str -> Noun = \verbum ->
    case verbum of {
      _ + "a"  => noun1 verbum ;
      _ + "us" => noun2us verbum ;
      _ + "um" => noun2um verbum ;
      _ + ( "er" | "ir" ) => noun2er verbum
                                ( (Predef.tk 2 verbum) + "ri" ) ;
      _ + "u"  => noun4u verbum ;
      _ + "es" => noun5 verbum ;
      _  => Predef.error ("3rd declinsion cannot be applied" ++
                          " to justone noun form" ++ verbum)
    } ;

  -- a-Declension
  noun1 : Str -> Noun = \mensa ->
    let
      mensae = mensa + "e" ;
      mensis = init mensa + "is" ;
    in
    mkNoun
      mensa (mensa +"m") mensae mensae mensa mensa
      mensae (mensa + "s") (mensa + "rum") mensis
      Fem ;

...

  mkNoun : (sn,sa,sg,sd,sab,sv,pn,pa,pg,pd : Str)
             -> Gender -> Noun =
    \sn,sa,sg,sd,sab,sv,pn,pa,pg,pd,g -> {
      s = table {
        Sg => table {
          Nom => sn ; Acc => sa ; Gen => sg ; Dat => sd ;
          Abl => sab ; Voc => sv
          } ;
        Pl => table {
          Nom | Voc => pn ; Acc => pa ; Gen => pg ;
          Dat | Abl => pd
          }
        } ;
      g = g
    } ;
```

Listing 4.2: Smart paradigm for nouns to identify declension classes according to noun suffixes. The function `init` drops the last character, `Predef.tk` drops a suffix of given length. `noun1` is an example for the implementation of a declension class in GF. `mkNoun` is a function that fills the inflection table to form a noun value

```
oper
  noun_ngg : Str -> Str -> Gender -> Noun = \verbum,verbi,g ->
    let s : Noun = case <verbum,verbi> of {
      <_    + "a"            , _ + "ae"> => noun1 verbum ;
      <_    + "us"           , _ + "i" > => noun2us verbum ;
      <_    + "um"           , _ + "i" > => noun2um verbum ;
      <_    + ( "er" | "ir" ) , _ + "i" > => noun2er verbum verbi ;
      <_    + "us"           , _ + "us"> => noun4us verbum ;
      <_    + "u"            , _ + "us"> => noun4u verbum ;
      <_    + "es"           , _ + "ei"> => noun5 verbum ;
      _  => noun3 verbum verbi g
      }

  -- Consonant declension
  noun3c : Str -> Str -> Gender -> Noun = \rex,regis,g ->
    let
      reg : Str = Predef.tk 2 regis ;
      regemes : Str * Str = case g of {
        Masc | Fem => < reg + "em" , reg + "es" > ;
        Neutr => < rex , reg + "a" >
        } ;
    in
    mkNoun
      rex regemes.p1 ( reg + "is" ) ( reg + "i" ) ( reg + "e" ) rex
      regemes.p2 regemes.p2 ( reg + "um" ) ( reg + "ibus" )
      g ;
```

Listing 4.3: Smart paradigm for nouns to identify declension classes according to noun suffixes given two noun forms and the noun gender to handle cases that cannot be handled by the smart paradigm in Listing 4.2. `noun3c` is the implementation of the third declension class for noun stems ending in a consonant.

form, which we will call the base form (Listing 4.2). From this we can easily extract the noun stem and by attaching the correct suffix for **number** and **case** we get the full paradigm (e.g. lines 15–23). From Table 4.2 we can see that we have to include two values for **number** and six for **case**. We put the strings into the correct shape with the function in lines 27–37 by putting the strings into the correct position of the two tables.

In the general case the same approach works for nouns of the fourth and fifth declension. Only in some cases the base form has the same suffix as nouns from the second class but they can be distinguished by looking at another noun form like the ***genitive singular*** e.g. lat. *domus* (***nom. sg.*** – eng. *house*), *domi* (***gen. sg.***) (second class) and lat. *casus* (***nom. sg.*** – eng. *case*), *casus* (***gen. sg.***) (fourth class).

The third declension class is the least regular one, in the sense that it is a collection of many nouns that inflect in different ways but each of them is quite regular in itself. That means that we can still derive the full paradigm from the two base forms we already used before plus the noun gender (Listing 4.3).

The gender information for most of the classes, except for the third one, can be derived easily because they mostly contain nouns of a certain gender that can easily be identified from the base form, feminine for the first class and masculine or neuter for the second class. But the third class is not only a

collection of many different nouns with different ways of inflection but also of different genders, which has to be added to the lexicon entry of the noun.

4.2.2 Adjective Inflection

```
1 param
2   Degree = Posit | Compar | Superl ;
3   Agr = Ag Gender Number Case ;
4 lincat
5   A = {
6     s : Degree => Agr => Str ;
7     adv : Adverb ;
8   } ;
```

Listing 4.4: Linearization type for adjectives together with all parametric features required, the parameter **Agr** uses the constructor **Ag** to construct new parametric values by combining several other parameter values

The inflection of adjectives is related to the noun inflection because adjective forms in most cases mirror the forms of the nouns they agree with, including the suffixes. The major difference is that adjectives inflect along two more features, gender and comparison level (or degree). On the other hand there are only three inflection classes for adjectives. The linearization type for adjectives can be seen in Listing 4.4. The parameter type **Agr** in line 3 looks different from the parameter types we have seen so far. It uses the constructor **Ag** to form a new parameter value from one **Gender**, one **Number** and one **Case** value. By combining finite types we again get a new finite type but also gain advantages e.g. when using pattern matching on these values.

The first and second declension classes, which cover the majority of adjectives, inflect in a very similar way to the noun inflection of the first and second declension classes. Because adjectives also inflect along the gender, the masculine and neuter forms correspond to the noun forms of the second declension while the feminine forms match the forms of nouns of the first declension class. Only in some cases the ***masculine nominative singular*** form is different (Bayer and Lindauer, 1994, p. 39) but the rest of the paradigm uses the same inflection as the noun declension. Only in the third inflection class we can find adjectives which have either three different forms, one for each gender, only two forms because masculine and feminine share forms, or even only one form independent of gender (Bayer and Lindauer, 1994, p. 38).

From the implementation point of view, we already encoded most of the suffixes used in adjective inflection in the noun morphology, which can be reused with just minor modifications. That works without any bigger problems for adjectives of the first two declensions.

Some additional complexity is added by adjective that do not have separate forms for the comparison levels but instead use the adverbs *magis* (eng. *more*) and *maxime* (eng. *most*) (Bayer and Lindauer, 1994, p. 44). This fact leads to a lot of "missing" forms in the paradigm and additional we need to keep track which adjectives form comparison levels in which way, e.g. in additional record fields.

4.2.3 Verb Inflection

```
param
  Person      = P1 | P2 | P3 ;
  VActForm  = VAct VAnter VTense Number Person ;
  -- For passive no anteriority because perfect forms are
  -- built using participle
  VPassForm = VPass VTense Number Person ;
  VInfForm  = VInfActPres | VInfActPerf Gender
            | VInfActFut Gender | VInfPassPres
            | VInfPassPerf Gender | VinfPassFut ;
  VImpForm  = VImp1 Number | VImp2 Number Person ;
  VGerund   = VGenAcc | VGenGen |VGenDat | VGenAbl ;
  VSupine   = VSupAcc | VSupAbl ;
  VPartForm = VActPres | VActFut | VPassPerf ;
  VAnter = VAnt | VSim ;
  VTense = VPres VMood | VImpf VMood | VFut ;
  VMood  = VInd | VConj ;
lincat
  V = {
    -- active verb forms
    act   : VActForm => Str ;
    -- passive verb forms
    pass  : VPassForm => Str ;
    -- infinitive verb forms (nominal verb forms)
    inf   : VInfForm => Str ;
    -- imperative verb forms
    imp   : VImpForm => Str ;
    -- gerund verb forms (nominal verb forms)
    ger   : VGerund => Str ;
    -- gerundive verb forms (adjectival verb forms)
    geriv : Agr => Str ;
    -- supine verb forms (nominal verb forms)
    sup   : VSupine => Str ;
    -- participle verb forms (adjectival verb forms)
    part  : VPartForm => Agr => Str ;
  } ;
```

Listing 4.5: Linearization type for verbs together with all parametric features required, parameters like **`VActForm`** use constructor like **`VAct`** to construct new parametric values by combining several other parameter values

Verb inflection in Latin can be seen as one of the biggest morphological challenges. Especially according to traditional grammar books, verbs do not only respect the features in Table 4.1, but besides these common verb forms, also form both nominal and adjectival forms. These are words derived from verbs that can be used in the same way as nouns or adjectives and express e.g. necessary actions (Bayer and Lindauer, 1994, pp. 172–190). It is open for discussion if these forms should be treated as verb forms or as separate lexical items.

The most practical way is to split the verb paradigm into different groups of verb forms like finite verb forms, infinite verb forms, adjectival verb forms and nominal verb forms which are stored in separate record fields (Listing 4.5). All these forms can be derived from base forms that are closely related to the word stems. In Latin traditionally three word stems are assumed for verbs, the

present stem, the ***perfect*** stem and the ***participle*** stem (Bayer and Lindauer, 1994, p. 68) from which all other forms can be derived. All of them can be derived from a base from like the ***present active infinitive*** which we use in our grammar or alternatively the ***first person present indicative active*** form used in other approaches.

Latin verbs are inflected according to one of four conjugation classes (from lat. *coniugare* – eng. *to connect, join*) of which the first, second and fourth can be considered regular, while the third class is an irregular class that covers several subclasses again. For the regular conjugation classes all forms can be derived from the one base form, for the irregular forms, some more forms have to be given explicitly: the ***first person singular indicative active*** forms in both ***present*** and ***perfect*** **tense** as well as the ***perfect passive*** participle.

```
1  oper
2    verb1 : Str -> Verb = \laudare ->
3      let
4        lauda = Predef.tk 2 laudare ;
5        laud = init lauda ;
6        laudav = lauda + "v" ;
7        pres_stem = lauda ;
8        pres_ind_base = lauda ;
9        pres_conj_base = laud + "e" ;
10       impf_ind_base = lauda + "ba" ;
11       impf_conj_base = lauda + "re" ;
12       fut_I_base = lauda + "bi" ;
13       imp_base = lauda ;
14       perf_stem = laudav ;
15       perf_ind_base = laudav ;
16       perf_conj_base = laudav + "eri" ;
17       pqperf_ind_base = laudav + "era" ;
18       pqperf_conj_base = laudav + "isse" ;
19       fut_II_base = laudav + "eri" ;
20       part_stem = lauda + "t" ;
21     in
22     mkVerb laudare pres_stem pres_ind_base pres_conj_base
23       impf_ind_base impf_conj_base fut_I_base imp_base
24       perf_stem perf_ind_base perf_conj_base
25       pqperf_ind_base pqperf_conj_base
26       fut_II_base part_stem ;
```

Listing 4.6: Function to create the full verb paradigm for a verb of the first conjugation class. The function `mkVerb` appends the correct suffixes to the base forms and fills the inflection tables

The approach to verb inflection is fundamentally the same as for nouns and adjectives. From a set of base forms all other forms can be derived by attaching the correct suffix to match all grammatical features that can be seen in Table 4.1. The biggest difference is that for the verb inflection more base forms are used. These base forms include the three stems, but also twelve additional base forms that can be easily derived from the stems that include already the morphemes for **tense**, **aspect** and **mood** but are missing the suffix for **person** (Listing 4.6). That way the base forms can easily be combined with a fixed set of suffixes.

For the nominal and adjectival verb forms we can determine the necessary

base form and then use the same methods we used before for nouns and adjectives to derive all the forms, i.e. reuse the functions that attache the correct suffixes.

An exception to this simple approach are the deponent verbs (lat. *deponere* – eng. *to lay away*). These verbs use their passive forms instead of active forms and for that reason do not have separate passive forms. However, these verbs in themselves regular again, but are a good example for words with only partially defined paradigms.

4.2.4 Other Classes

Other classes that should be mentioned here are pronouns of various kinds including personal pronouns, possessive pronouns, demonstrative pronouns but also quantifiers, prepositions and adverbs.

The most common determiners in the traditional sense, the definite and indefinite article, do not exist in Latin, but in some situations similar things can be expressed with pronouns and quantifiers.

Most words in these other classes are either irregular or they are not very common in the vocabulary of the language so that it is not very useful to define smart paradigms but instead it is easier to just list all forms explicitly.

Finally adverbs and prepositions in most cases do not inflect at all with the exception of some adverbs which have different forms according to the comparison level.

4.3 Lexicon

```
1  lin
2    airplane_N = mkN "aeroplanum" ; -- Modern word
3    apartment_N = mkN "domicilium" ;  -- noun 2nd decl
4    art_N = mkN "ars" "artis" feminine ; -- Noun 3rd decl
5    bad_A = mkA "malus" ; -- Simple adjective
6    bank_N = mkN "argentaria" ; -- Homonymous Noun
7    computer_N = mkN "computatrum" ; -- Modern word
8    country_N = mkN "terra" ; -- -ae f. -- Noun 1st decl
9    camera_N = ResLat.useCNasN
10                (AdjCN (PositA (mkA "photographicus") )
11                  (UseN (mkN "machina" ) ) ) ; -- Paraphrase
12   come_V = mkV "venire" ; -- Verb 4th conj
13   die_V = mkV "mori" "mortuus" "morturus"; -- Deponent verb
14   house_N = mkN "domus" "domus" feminine ; -- Noun 4th decl
15   live_V = mkV "vivere" "vivo" "vixi" "victurus" ; -- Verb 3rd conj
16   love_V2 = mkV2 "amare" ; -- Verb 1st conj
```

Listing 4.7: Various entries from the RGL lexicon for Latin

In GF it is a convention that grammar rules are either classified as lexical or phrasal rules (Ranta, 2011, pp. 98). Lexical rules are the basic constituents and always appear at the leaves of the syntax trees while phrasal rules combine their parameters to a new part and appear inside the tree.

Lexical categories are categories that only appear as the result type of lexical rules. Lexical categories again can be divided into open and closed categories (Ranta, 2011, p. 99).

Closed categories are word classes where it is possible to exhaustively list all members, i.e. classes that are not productive. Usual candidates are determiners, quantifiers, prepositions and conjunctions. There are only few words of a language in the closed categories compared to the open categories.

More interesting are the open classes which are still productive because new words of these classes can be added to a language at any time. Usual open categories are nouns, adjectives and verbs. The vocabulary of the open categories is listed in the lexicon.

The lexicon is an essential part of a GF grammar to test the other parts of the grammar. For that reason the RGL defines a minimal lexicon with about 350 dictionary entries. These entries have been selected to include both the Swadesh list of 207 basic concepts as well as an additional variety of modern concepts (Ranta, 2011, p. 233).

Lexicographic work tends to involve common challenges like homonymy. Other challenges appear in the multilingual context of GF or in the context of historic languages.

The most commonly known problem is homonymy, i.e. when the same string denotes two different semantic concepts in the language. A prime example is the word *bank* which is homonymous in several languages, in English it can denote the place to store money and the river bank. One reason why we have to deal with homonymy in our task is that the RGL abstract syntax defines the lexicon as a set of abstract identifier consisting of the English base form and the lexical category. The identifier in question is `bank_N` (Line 6 in Listing 4.7). That means we have to decide which concept in Latin we want to assign to this identifier, either *argentaria* as the place for money or *ripa* as the location at the river.

An easy but misleading approach would be to define both meanings as variants of the same concept. But that would allow us to map *ripa* to the identifier `bank_N` and the identifier `bank_N` back to `argentaria` and as a consequence *ripa* = `bank_N` = *argentaria*, and consequently *ripa* = *argentaria*, which we want to avoid. It would also lead to serious problems in connection with multilinguality.

A better approach is to select one of the two meanings, e.g. the more common one, and add other abstract identifiers for the other meanings. So we can define `bank_N` as *argentaria* and `bank2_N` as *ripa* so we get *ripa* = `bank2_N` $\neq$ `bank_N` = *argentaria*.

Another problem we encounter in the multilingual setting of the RGL is that we need to assign a translation to a word that does not as such exist in the language. The usual approach would be to paraphrase the missing word (Line 9–10). The problem with a paraphrase usually is that it belongs to a phrasal category while the lexicon entry requires a lexical category. That means we need to translate between the type of a lexical category and the type of a phrasal category.

Finally, in this very specific case of a historic language, another challenge was the presence of modern concepts in the lexicon which are not easy to translate into Latin (e.g. Line 2). Even though the Vatican, the only country

where Latin is still the official language, provides an official translation list for modern concepts between Latin and Italian, this list is not complete either and requires the translation via Italian. The best solution we encountered is to use information provided by the Latin Wikipedia which, thanks to a crowd-sourcing approach, is able to provide more than 100 000 pages (Vicipaedia, 2018) that are linked to their counterparts in various languages and provide a rich resource for translating lexical concepts.

Besides these very specific problems, the general challenge for the creation of the lexicon is to spell out all the necessary information that cannot be derived otherwise, e.g. by using the smart paradigms.

4.4 Grammar rules

After looking at the lexical entities, the smallest components we are concerned with in our grammars, now we can combine them to larger phrases up to the topmost level of utterances, i.e. sentences and phrases in which all grammatical features are fixed to form plain strings.

The most important part of this task is to decide which parts can already be put together and which parts have to be kept available and flexible for later modification.

To keep up the flexibility we can use the tables and records. With tables we can pass features on through the syntax trees and with records we can keep parts of the phrases separate. So records can be used to model discontinuous constituents as well as to store temporary information influencing the construction of phrases.

Which rules have to be present in a grammar is again defined by the RGL abstract syntax. These rules range from common and generic to very specific. This section focuses on the most commonly encountered rules and how they can be implemented for Latin.

In Listing 4.8 a definition of basic phrase types with the additionally necessary parameter types is given. Most of the parameter types used in the phrasal types are the same as used for nouns and verbs, some additional ones are reuqired and some are different and seem redundant, like e.g. **`VTense`** and **`Tense`** but are required because the tense system in the RGL is different from the tense system used in the grammar book. However, a simple mapping between the tense systems is possible.

4.4.1 Noun Phrases

In the lexicon there are the two lexical categories of nouns and adjectives which can be used to form noun phrases (**`NP`**). As an intermediate category the category of common nouns (**`CN`**) is introduced. They can be formed directly from nouns and can be modified by adjectives.

Adjectives can be transformed into adjectival phrases (**`AP`**) by selecting the desired comparison level (positive, comparative or superlative). These adjectival phrases can then be combined with a **`CN`** to form a **`CN`** again. Because adjectives can appear before or after the noun in Latin we keep track of the position by putting the APs in the separate record fields `preap` and `postap` (Line 13 of Listing 4.8).

```
1  param
2    Anteriority = Simul | Anter ;
3    Tense       = Pres | Past | Fut | Cond ;
4    Polarity    = Pos | Neg ;
5    VQForm      = VQTrue | VQFalse ;
6    Order       = SVO | VSO | VOS | OSV | OVS | SOV ;
7  lincat
8    AP =   { s : Agr => Str } ;
9    CN =   { s : Number => Case => Str ; g : Gender ;
10            preap : AP ; postap : AP } ;
11   NP =   { s : Case => Str ; g : Gender ; n : Number ;
12            p : Person } ;
13   VP =   { s : VActForm => VQForm => Str ;
14            obj : Str ; adj : Agr => Str } ;
15   Prep = { s : Str ; c : Case ; isPost : Bool } ;
16   -- extend verb phrase by adding a preposition
17   VPSlash = VP ** {c : Prep} ;
18   Cl =   { s : Tense => Anteriority => Polarity => VQForm
19                  => Order => Str } ;
20   S =    { s : Str } ;
```

Listing 4.8: Linearization types for adjective phrases (`AP`s), common nouns (`CN`s), noun phrase (`NP`s), verb phrases (`VP`s), clauses (`Cl`s) and sentences (`S`) together with all additional parametric features required

By combining a `CN` with a determiner we can create an `NP`. Latin itself does not use the determiners commonly found in modern languages like the definite and indefinite article. However, because the **number** of a `CN` is not fixed and has to be determined by a different constituent, we need to employ these articles to determine the **number** feature of the noun phrase even though they do not have a representation on the surface. Furthermore, some other quantifiers like *omnis* (eng. *all*) which can be used as determiners. The **number** value of the determiner dominates the **number** value of the noun and the adjective forms, the noun **gender** dominates the adjective **gender** value, the **case** is still undecided, and the **person** that is used in the verb agreement is either ***third person*** by default, or if the noun phrase is constructed from a pronoun, adopted from it. That leads to the type given in Listing 4.8 on line 15 for `NP`s.

4.4.2 Verb Phrases

Intransitive verbs can be used as verb phrases (`VP`) directly and transitive verbs can be combined with a direct object into a `VP`. Also adjective phrases can be used together with the copula *esse* to form a `VP`.

To create verb phrases from intransitive verbs most of the information contained in the verb can be ignored and only the ***finite active*** verb forms are used. That means the verb forms are inflected by **tense**, **anteriority**, **number** and **person**. The tense system with the separation into **tense** and **anteriority** is based on the work of Reichenbach (1947). As an additional inflection feature the parameter `VQForm` is added which controls the use of the question suffix *-ne*. Record fields in the `VP` for direct objects and adjective phrases are just left empty.

In the case of transitive verbs an intermediate slash category is used. A slash

category (C_1/C_2) expresses the fact that the category is missing a complement C_2 to form a value of category C_1, a concept used in many modern grammar formalisms like Combinatory Categorial Grammars and GPSG with slightly different semantics (Wood, 1993, p. 107).

A transitive verb can be seen as a category which combined with a noun phrase to form a verb phrase, similar to a slash category `VP/NP`. For that reason it can also be transformed to the `VPSlash` category in GF which then can be combined with the object. In this case again the ***finite active*** verb form is stored in the `VP` the same way as the intransitive verbs. Additionally the noun phrase is stored as the direct object. The most common grammatical **case** for the direct object is the ***accusative***, but this can be overridden by information stored in the transitive verb.

Finally, for predicative expression, i.e. where an adjective is used as a complement to a noun, the verb field is filled with the required forms from the copula and the adjective phrase is stored in the `adj` field. Otherwise this field remains empty.

4.4.3 Clauses and Sentences

To finally form sentences we can combine a subject noun phrase with a complete verb phrase to a clause (`Cl`), which contains most of the information of the sentence but several grammatical features like **tense**, **anteriority**, **polarity**, the word order as well as if it is a proposition or a question, are not decided yet. These features are encoded in a table again. The rule to create a sentence is called `PredVP` which can be seen in Listing 4.9. On this level we have to deal with a high level of complexity and we do not want to go too much into the detail but basically we have three main record fields, the subject noun phrase in the `s` field, the predicate verb phrase in the `v` field and the object noun phrase in the `o` field. In addition we have one field for negation particles (`neg`) and one for adverbial modifiers (`adv`). We allow adverbs in six different positions, both between the phrases and within phrases. All this is necessary to deal with the flexibility in word order we encounter in Latin.

To form a sentence from a clause the parameter values we just named have to be finally decided. Depending on theses decisions all the parts are assembled. These involve the subject noun phrase, the verb, potentially direct object or predicative adjectives, negation particles, adverbs and so on. Some of these are already decided on in the clause, others are postponed until the formation of a sentence.

For example for Subject-Object-Verb word order we start with the subject noun phrase in ***nominative case*** which can be followed by a negation particle. In a predicative setting an adjective phrase will be followed by the verb form agreeing with the subject noun phrase in **number** and **person**, in case of a transitive verb, instead the direct object is placed before the verb. This provides us with complete sentences and concludes the description of syntactic rules.

```
lin
  PredVP np vp = -- NP -> VP -> Cl
  let
    -- combines adverbs from noun phrase and verb phrase
    adv  = np.adv ++ vp.adv ;
    -- helper functions to either place the adverb in the designated position
    -- or an empty string instead
    pres   : AdvPos -> Str = \ap -> case ap of { PreS => adv ; _ => [] } ;
    prev   : AdvPos -> Str = \ap -> case ap of { PreV => adv ; _ => [] } ;
    preo   : AdvPos -> Str = \ap -> case ap of { PreO => adv ; _ => [] } ;
    preneg : AdvPos -> Str = \ap -> case ap of { PreNeg => adv ; _ => [] } ;
    ins    : AdvPos -> Str = \ap -> case ap of { InS => adv ; _ => [] } ;
    inv    : AdvPos -> Str = \ap -> case ap of { InV => adv ; _ => [] }
  in
  {
    -- subject part of the clause:
    -- ap is the adverb position in the clause
    s = \\ap =>
      -- adverbs can be placed in the beginning of the clause
      pres ap ++
      -- the determiner, if any
      np.det.s ! np.g ! Nom ++
      -- adjectives which come before the subject noun, agreeing with it
      np.preap.s ! (Ag np.g np.n Nom) ++
      -- adverbs can be placed within the subject noun phrase
      ins ap ++
      -- the noun of the subject noun phrase in nominative
      np.s ! Nom ++
      -- adjectives which come after the subject noun, agreeing with it
      np.postap .s ! (Ag np.g np.n Nom) ++
      -- second part of split determiners
      np.det.sp ! np.g ! Nom ;

    -- verb part of the clause:
    -- tense and anter(ority) for the verb tense
    -- vqf is the VQForm parameter which defines if the ordinary
    -- verbform or the question form with suffix "-ne" will be used
    -- ap is the adverb position in the clause
    v = \\tense,anter,vqf,ap =>
      -- adverbs can be placed in the before the verb phrase
      prev ap ++
      -- verb phrase complement, e.g. predicative expression, agreeing
      -- with the subject
      vp.compl ! Ag np.g np.n Nom ++
      -- adverbs can be placed within the verb phrase
      inv ap ++
      -- verb form with conversion between different forms of tense and aspect
      vp.s ! VAct (toVAnter anter) (toVTense tense) np.n np.p ! vqf;

    -- object part of the clause, it only depends on the adverb position
    o = \\ap => preo ap ++ vp.obj ;

    -- optional negation particle, adverbs can be placed before the negation
    neg = \\pol,ap => preneg ap ++ negation pol ;

    -- after combining the clause the adverb is empty again
    adv = ""
  } ;
```

Listing 4.9: The PredVP rule to combine a noun phrase and a verb phrase to a clause

4.4.4 Free Word Order

One commonly named characteristics of Latin is the free word order, a big challenge for every endeavor to formalize this language. Because lots of syntactic information is morphologically encoded it is not that important to have a strict order of constituents and their sub-parts in a sentence. This allows flexibility on several levels.

On the top-level the main constituents like subject noun phrase, verb and direct object can be combined in an almost arbitrary way. Given these three parts, six combinations are theoretically possible. However, an empirical evaluation on a Latin treebank by Bamman and Crane (2006) shows that classical authors prefer verbs in the final position while later authors prefer verbs in second position.

To deal with this general order of constituents a new parameter type (Line 12 in Listing 4.8) can be defined which defines the order on the clause level and provides access to all possible combinations in a table. As a default value the word order subject-object-verb which is most commonly used by Caesar is assumed.

The large quantity of morphological information also allows larger distances between sub-parts of larger constituents like noun or verb phrases because the agreement in morphological features provides sufficient information to identify the parts belonging together. That allows that particles including adverbs can be placed in almost any position in a sentence, even within larger phrases. To allow this it is necessary to change the phrase categories into split or discontinuous categories, i.e. categories that do not just use one record field to store the surface representation but instead split it into parts that are stored in different record fields. That way the assembly of phrases can be delayed to a later point when all parts, including particles, of a sentence are known and can be put in the right place.

As these two examples show, it is possible to solve challenges in handling natural languages including discontinuous constituents with the help of the two constructs of tables and records in a clear way.

4.5 Status and Extensions

The RGL abstract syntax defines 1119 abstract function, consisting of both lexical and phrasal rules. The Latin RGL grammar currently implements 847 of these with 272 rules missing. The implementation of the missing rules remains as future work. The task to figure out which of the missing rules are more relevant can be seen as part of the general future task of evaluating the grammar.

Besides the work on completing the coverage of the abstract syntax, adding external resources is necessary to increase the usability of the grammar. In that direction we can report some success: The Latin dictionary provided by William Whitacker's Words program (Whitaker, 2006), containing 39225 entries could in large parts be translated into a GF wide-coverage dictionary containing 33355 entries. Some of the conversion was challenging due to a large amount of Greek loan words, the support for which had to be added to the grammar.

The grammar in the form described above covers basic and general concepts that are necessary to handle language complexities posed by the Latin language. It is also sufficient for the current use in the MULLE Latin learning application. However, it can be extended by either adding more syntax rules or by further extending the lexical coverage.

Chapter 5

Latin Language Learning

In Chapter 3 we introduced the general ideas and concepts for a modern language learning application. These ideas led to the implementation of a web application that can be used in a Latin language class. The system in the current state is built with Swedish as the meta language and Latin as the object language, so where necessary we will gloss the examples with an English translation.

In this chapter we will first revisit the concept of grammar-based text modification on the word level and show using a concrete example how the user interaction for translation exercises works. After that we will discuss concepts of gamification to improve the user experience and we conclude the chapter with the description of an experiment to evaluate our system. In this evaluation the main focus is on the change of learner motivation and attitude.

5.1 User Interaction

When the user logs in into the MULLE web interface they are presented with the lesson view for the course (Figure 5.1) which gives an overview over the total progress. All the lessons from the course are listed but some of the lessons can be disabled, for example if they depend on other lessons which have to be solved first. Also already finished lessons are marked in green and can usually be repeated to improve the results. The author of a course can also create special lessons which are not repeatable. This can be used, e.g. for placement tests or intermediate tests to assess the learning progress. The lesson screen also gives some general statistics including the amount of exercises in a lesson and the current score. When clicking on one of the lessons the user is transferred to the exercise view for this lesson.

The exercise view is the most relevant one because it is what a student will be confronted with most of the time. Examples for the exercise view can be seen in Figure 5.4 to 5.13. At the top of the screen some statistics is shown, like the time spent, the number of clicks used, the number of exercises already solved in the lesson and as well as the number of remaining exercises.

In the middle of the screen the user task is placed. Two sentences in different languages are shown to the user and their task is to use the method of word-based text editing to change the sentence at the bottom to make it a translation of the sentence at the top. The parts that are already matched are highlighted in different colors. The matching phrases are determined by matching subtrees in the two syntax trees. To increase the difficulty level these highlights can be disabled.

Now we walk through a concrete example to show how our input method can be used to solve a translation task. For the start we need a grammar (Listing 5.1) and two valid syntax trees within the grammar (Figure 5.2 and Figure 5.3). When looking at both trees, it is possible to see which steps are necessary to make them the same. For the user who only sees the sentence on the surface this can be a lot more challenging. We will now show one possible sequence of steps to solve the exercise, highlighting the changes both in the syntax tree and on the surface.

To simplify the presentation of the example, we only show the categories and not the functions in the syntax trees and we will use the category `S` as

Figure 5.1: The lesson screen, the first screen the user sees after logging in, shows the list of lessons with color-coded status (red locked, blue unfinished, green finished) and general information including the current score

the topmost or start category. Furthermore we will ignore the *(definite)* node in the tree, the function of which is to determine which function was used to create the noun phrase. This is necessary because Latin does not express the definite or indefinite article which usually defines the **number** feature of the noun. However, this detail does not have an influence on our example.

5.1.1 Solving an Exercise

In the exercise view the user can click on any word in the bottom sentence. The click then is mapped to the node in the tree which introduced the corresponding word. For example the user can click on the word *imperator* which is translated to the left-most node at the bottom of the tree. The systems then generates all subtrees the same root category as the category pointed to. In this case the root category is `N` and all trees with the root category are enumerated and converted to a list of suggestions. Because it does not make any sense to replace a word with itself, this option is filtered out. The menu after the first click can be seen in Figure 5.5 on the right while on the left the abstract syntax tree can be seen with the highlighted node. The next click on the same position moves the pointer up the tree to the parent of the currently selected node. In this example the pointer is moved up to the `CN` node above the previously selected `N` node (Figure 5.6, left). The menu still does not contain the entry that would bring the user closer to the goal. So they can continue clicking and whenever the user clicks in the same position, the pointer is moved further up in the tree until it arrives in the root node.

Clicking on a different word sets the pointer to the node which introduces this word instead of moving the pointer further up. This helps when the user

accidentally clicked on the wrong word and allows them to explore several paths to the way to the solution.

In our example, the next click will lead to the pointer pointing to the `NP` node (Figure 5.7) where we find the correct subject noun phrase *Gallia* (eng. *Gaul* **nominative singular**) in the suggestion list. After clicking on the suggestion both the tree and the sentence are updated (Figure 5.8).

As a next step the user can change the verb phrase, starting by a click on the verb *vincit* (eng. *conquer* **third person singular**, Figure 5.9). This again moves the pointer to the node which is responsible for introducing the word, here the `V2` node. With one more click the pointer ends up pointing to the `VP` (Figure 5.10) where we can select *victus est* (eng. *be conquered* **third person singular**) from the menu which replaces the whole subtree and also the whole verb phrase *Galliam vincit* (eng. *conquer Gaul* **third person singular**) with the correct one (Figure 5.11).

The final step to make the trees the same is to add the adverb. According to the grammar and the syntax tree the adverb has to be attached on the `S` node at the top of the tree. But the user has no access to this information, so in theory it would not matter on which word the user starts clicking until they end up at the top of the tree. However, starting at the verb to add an adverb seems reasonable. After several clicks on the verb *est* (eng. *be* **third person singular**, Figure 5.12) the menu contains the option to add the desired adverb.

After selecting this option the abstract trees are the same (Figure 5.13) and the system congratulates the user to the success and gives the final statistics including the time spent and the amount of clicks necessary. This concludes our example of a user interaction to solve a translation exercise using the method for text modification we propose.

```
abstract PrimaRules = Cat, Conjunction ** {
cat CS ;
fun
  useA : A -> AP ;
  simpleCl : NP -> VP -> Cl ;
  usePN : PN -> NP ;
  usePron : Pron -> NP ;
  useCNdefsg : CN -> NP ;
  useCNindefsg : CN -> NP ;
  useCNindefpl : CN -> NP ;
  complexNP : Det -> CN -> NP ;
  conjNP : NP -> NP -> ListNP ;
  extConjNP : ListNP -> NP -> ListNP ;
  useConjNP : Conj -> ListNP -> NP ;
  useN : N -> CN ;
  attribCN : AP -> CN -> CN ;
  apposCNdefsg : CN -> PN -> NP ;
  useCl : Cl -> S ;
  advS : Adv -> S -> S ;
  intransV : V -> VP ;
  transV : V2 -> NP -> VP ;
  complVA : VA -> AP -> VP ;
  useS : S -> CS ;
}

abstract PrimaLex = Cat ** {
fun
  copula_VA : VA ;
  copula_V2 : V2 ;
  -- Vocabulary p11      -- More vocabulary p19
  imperium_N : N ;       puella_N : N ;
  Romanus_A : A ;        laetus_A : A ;
  magnus_A : A ;         amicus_N : N ;
  imperator_N : N ;      anxius_A : A ;
  habere_V2 : V2 ;       vinum_N : N ;
  tenere_V2 : V2 ;       bonus_A : A ;
  multus_Det : Det ;     pater_N : N ;
  civitas_N : N ;        felix_A : A ;
  externus_A : A ;       coniux_N : N ;
  vincere_V2 : V2 ;      sapiens_A : A ;
  victus_A : A ;         numen_N : N ;
  saepe_Adv : Adv ;      ingens_A : A ;
  provincia_N : N ;      -- Not in vocabulary list but in text
  devenire_V2 : V2 ;     Augustus_PN : PN ;
  Gallia_PN : PN ;       Caesar_N : N ;
  Africa_PN : PN ;       he_PP : Pron ;
  Germanus_N : N ;       and_Conj : Conj ;
  hostis_N : N ;
  dicere_V : V ;
}
```

Listing 5.1: Abstract syntax for the first lesson, containing both the syntax rules extracted from the lesson text and the lexicon given as a vocabulary list in the lesson

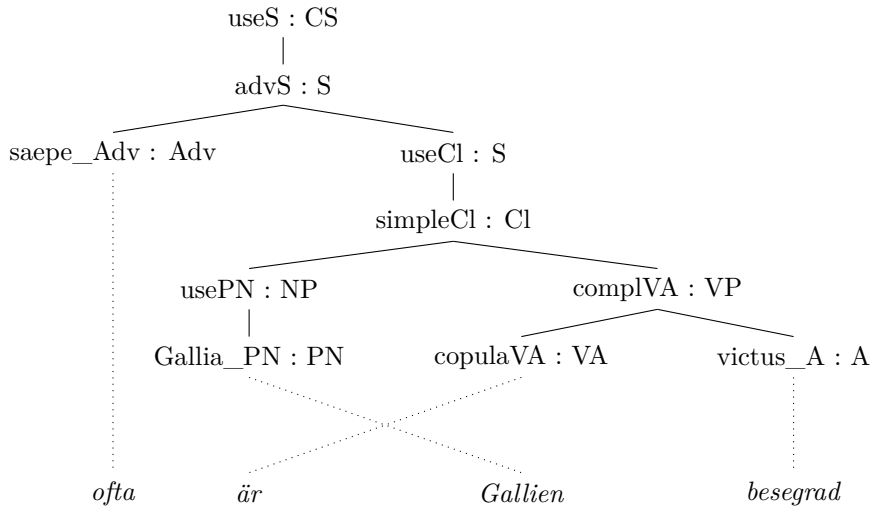


Figure 5.2: The first abstract syntax tree with a linearization in Swedish (eng. *often Gaul is conquered*

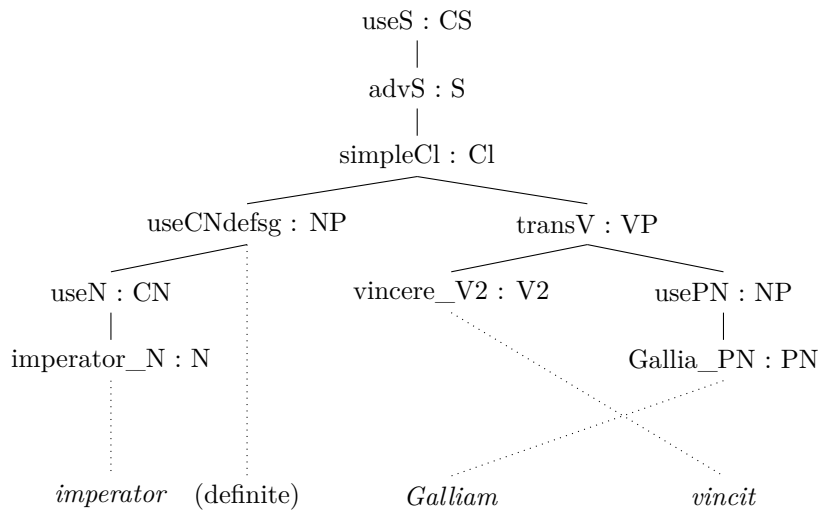


Figure 5.3: The second abstract syntax tree with a linearization in Latin (eng. *the emperor conquers gaul*

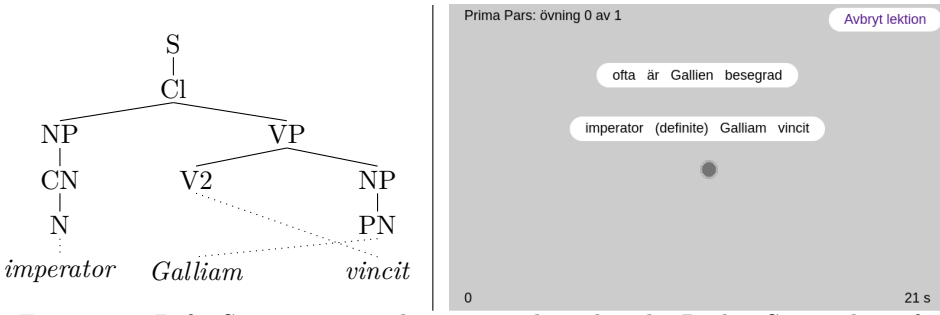


Figure 5.4: Left: Syntax tree without any selected node, Right: Screenshot of the system at the start before any click

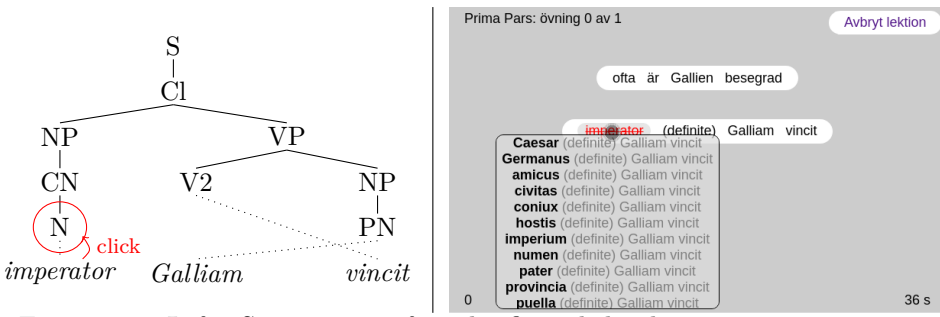


Figure 5.5: Left: Syntax tree after the first click, the pointer pointing to the `N` node on the left of the tree, Right: Screenshot after the first click on *imperator*, showing the suggestion list including all nouns from the lexicon except *imperator*. All options lead to a substitution

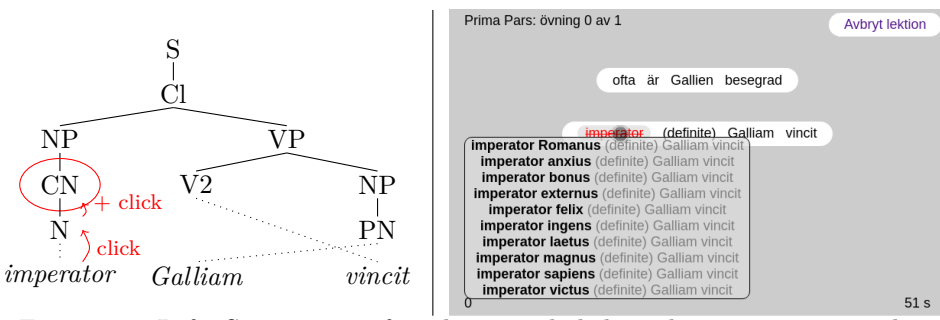


Figure 5.6: Left: Syntax tree after the second click in the same position, the pointer now pointing to the `CN` node above the `N` node, Right: Screenshot after the second click on *imperator*, showing the suggestion list containing *imperator* modified by all possible adjectives. All possible operations are equivalent to the insertion of an adjective

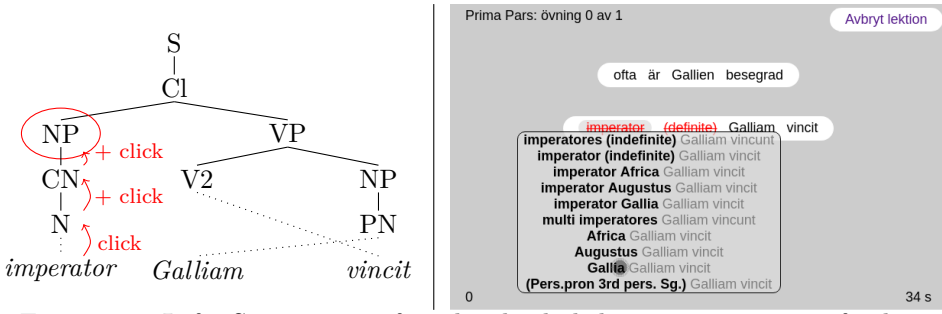


Figure 5.7: Left: Syntax tree after the third click, pointer going up further through the tree now pointing to the **NP** node, Right: Screenshot after the third click on *imperator*, showing the suggestion list for noun phrases: the option to change to plural or to indefinite form of *imperator*, using *imperator* as a title for proper names, changing the determiner to the quantifier *multi* (eng. *many*) or replacing the whole noun phrase by a proper name or pronoun

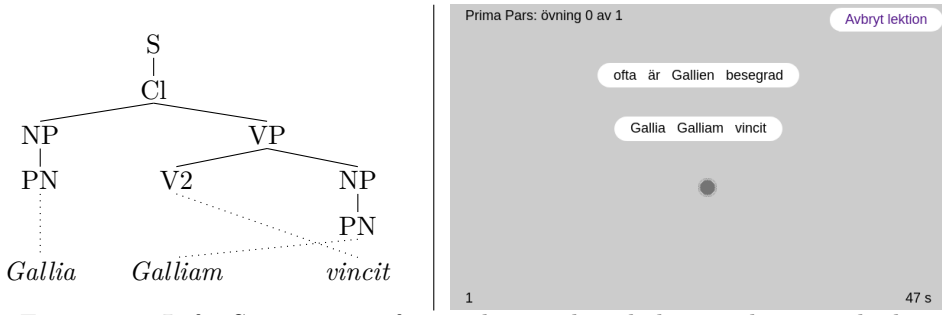


Figure 5.8: Left: Syntax tree after replacing the whole **CN** subtree with the **PN** subtree, Right: Screenshot after updating the tree and the sentence on the surface

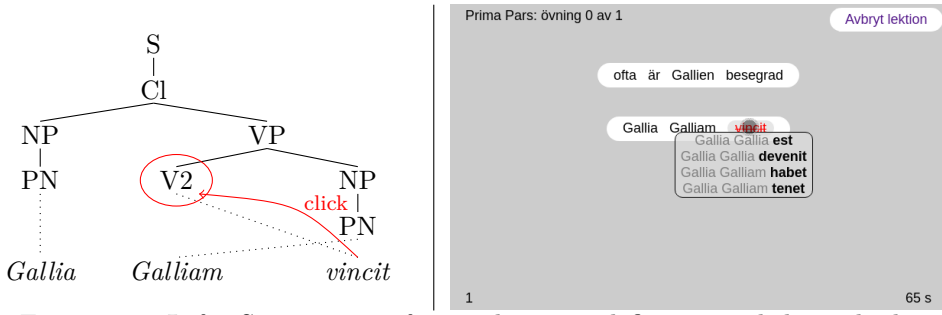


Figure 5.9: Left: Syntax tree after updating and first new click, with the pointer pointing to the **V2** node, Right: Screenshot after first click on *vincit* (eng. *conquer* **third person singular**), showing a list of transitive verbs to replace it with

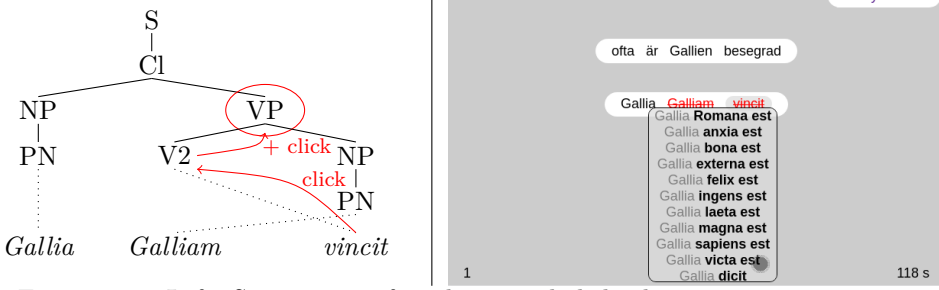


Figure 5.10: Left: Syntax tree after the second click, the pointer now pointing to the `VP` node above the `V2` node, Right: Screenshot after the second click on *vincit*, with the suggestion list showing various types of verb phrases like predicative expressions and intransitive verbs including the intended option *victus est*

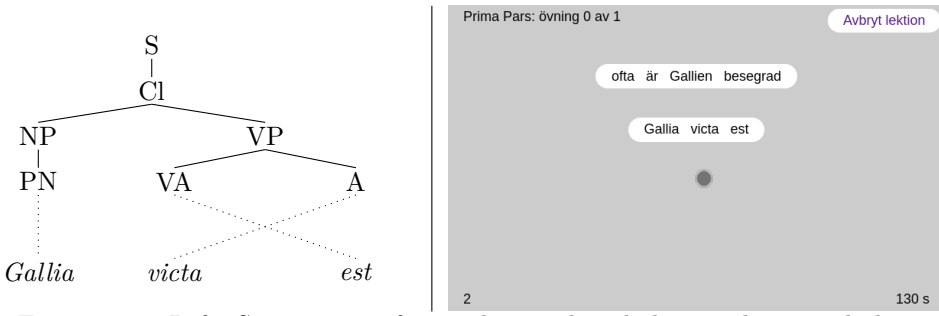


Figure 5.11: Left: Syntax tree after replacing the whole `VP` subtree including the `V2` node with a new `VP` subtree, Right: Screenshot after updating the tree and the sentence on the surface

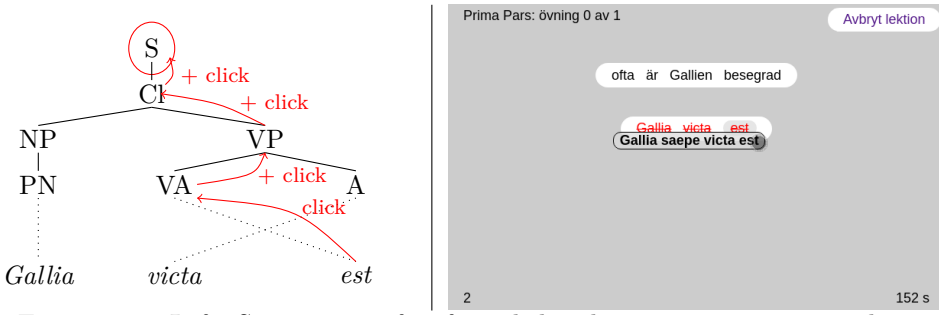


Figure 5.12: Left: Syntax tree after four clicks, the pointer pointing to the `S` node at the root of the tree, Right: Screenshot after four clicks on *est* (eng. *be* **third person singular**), showing the option to add the adverb *saepe* (eng. *often*) which makes the sentence a proper translation of the sentence at the top

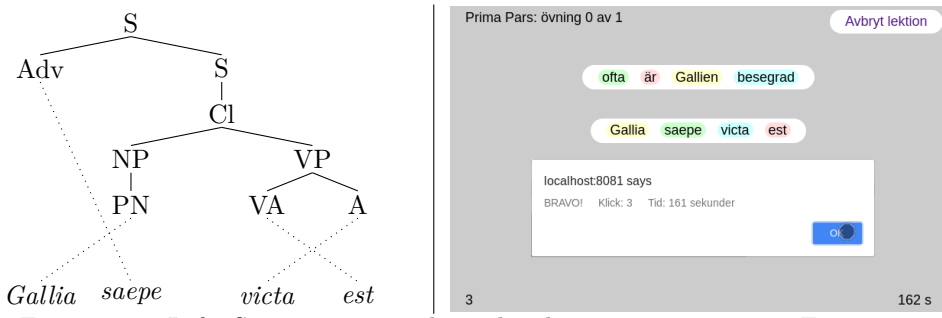


Figure 5.13: Left: Syntax tree matching the abstract syntax tree in Figure 5.2, Right: The result screen congratulating the user and showing the time and clicks spent on the exercise

5.2 Improving the User Experience

One important aspect of modern language learning applications is the concept of gamification (Deterding et al., 2011). It is the idea to enrich a serious task with certain elements of a game to improve the motivation to solve this task.

Several approaches to gamification have been suggested including GameFlow by Sweetser and Wyeth (2005) and MICE by Lafourcade (as described in Fort et al. (2014, Section 4)). The GameFlow approach translates the more general Flow concept (Csikszentmihalyi, 1990) to computer games. MICE on the other hand was developed in the context of Games with a Purpose, a method to include the general public into academic research (Von Ahn, 2006).

Both GameFlow and MICE list a set of criteria that help to make a task more game-like and involving.

For GameFlow factors for games are: **Concentration**, **Challenge**, **Player skills**, **Control**, **Clear goals**, **Feedback**, **Immersion** and **Social interaction**. MICE is based on and an acronym for: **Money and reward**, **Ideology and interest**, **Constraint and retention** and **Ego and community**.

Of these different factors we were able to integrate the following:

Concentration Minimize the distraction from the task

Challenge Provide a scoring schema

Control Provide an intuitive way to solve the task

Clear Goals Adopt a lesson structure

Immediate Feedback Include a coloring scheme to demonstrate progress

These already cover the majority of the eight parts of the GameFlow concept and in comparison with the MICE method, we provide some kind of **reward**, we support the players **interest** to learn a language, we give **feedback**.

Especially the lesson structure is essential to our task because it makes the learning progress explicit. After finishing a set of exercises a lesson is considered finished and the student can move on to the next lesson. That way it is always clear where on the journey to learn the language the student is at the moment.

Independent variables	Dependent variables
Use of application (yes/no)	
Previous knowledge (learner level)	Learning outcome (translation speed)
Learner motivation	Learner motivation (difference to start)
Use of application (usage time)	
Dislike of computers	

Table 5.1: Variables identified for our setup

More aspects of gamification are possible, especially including social aspects to both increase competition but also collaboration, but have not yet been included. These should be included at a later point because social aspects are an important part both of GameFlow and MICE.

5.3 Evaluation

To both justify our work in the scientific context as well as to collect feedback for improvement, we designed an experiment to evaluate our language learning application integrated into a classroom setting. The main question to be researched in this experiment is on the one hand to show a positive change in the learning outcome but more importantly on the other hand that the learner attitude can be influenced in a positive way.

Language teachers, especially for ancient languages, point out that anxiety among students can be a serious threat to successful learning (Takahashi and Takahashi, 2015). By providing a game-like approach we intend to target this problem.

5.3.1 Experimental Design

In this section we describe the setup for an experiment to evaluate the MULLE Latin learning application. The experiment is both within-subject and between-subject and focuses within the language learner on the change in language skills and learner's attitude over the time of the experiment and compares the results from a treatment group with one or more control groups.

We describe all relevant aspects for a valid experiment: the identified variables, the general setup, the sampling of candidates and the evaluation of relevant factors. Consequently we also describe the pilot for this experiment.

5.3.1.1 Variables

We identified independent and dependent variables relevant to the posed questions. Both the task of language learning and the classroom setting provide many potential aspects to an experiment which lead to various kinds of variables. To only have the relevant variables influence the outcome of the experiment it is necessary to control for all the other variables. The set of identified variables can be seen in Table 5.1.

The most informative independent variable is the use of our application compared to just the traditional teaching approach. Other variables are also possible including the choice of user input and user interface.

After identifying the variables they have to be operationalised. Both dependent variables are of interest but difficult to measure. The change in learner motivation is very subjective and for a change in learning outcomes a longitudinal study is required. We approximate a reliable measure for the change in learning outcome by using timed placement test both in the beginning and end of the experiment and measure a change in speed to solve the task. For the learner motivation both a self-assessment in a user survey and individual interviews can be conducted.

5.3.1.2 Setup

Other language learning systems like Duolingo have been evaluated over a longer period in a closed setting under strongly controlled conditions and supervision (Vesselinov and Grego, 2012). Instead we try to increase the openness and reduce the time effort.

The experiment can be conducted online and the expected runtime is four weeks. Within this time four different lessons will be made accessible, one by one, week by week. Each participant is provided with anonymous user credentials that help to identify a user within the system without connecting them directly to private and sensitive information.

When logged in for the first time the user is asked to answer a user survey in the form of the questionnaire seen in Figure 5.14. These answers can be uses to control for background variables. The questionnaire also asserts the previous knowledge and motivation in the beginning by self-assessment of the user. The majority of the questions are closed questions with five possible answers in a Likert scale ranging from "Very good" to "Very bad" or similar categories. Alternatively an even number of choices can be used to enforce a clear decision (Allen and Seaman, 2007). The questions are designed in a way that several questions cover the same variables and cross-check for inconsistent answers.

After this survey the user takes a simple, timed placement test in which a set of translation exercises from all four lessons are to be solved. Each exercise also contains a timeout so that the result is either the time used to solve the exercise or the fact that the user failed to do so. This test contributes to the measure of previous knowledge and gives the base relative to which the change in learning outcome can be measured.

After this introductory test the user has access to the system for the whole time of the experiment to practice their translation skills. In the meantime more data like the intensity of the system use is kept track of. This can be used to correlate it to the other results and to contribute to individual interviews by using stimulated recall (Fox-Turnbull, 2009).

At the end of the experiment a variation of the first step is repeated. This again involves a modified version of the user survey (Figure 5.15) as well as a second placement test. Finally, all the data gathered in the experiment has to be analyzed.

5.3.1.3 Sampling

Depending on the location and the languages involved, the sampling question, i.e. the selection of a suitable group of participants, can either be solved by using the whole population, for example beginner Latin learners are not that

common. In case of a large population more elaborate sampling methods would be required.

The participants have to be split into several groups, one treatment group that gets access to our application and one or more control groups. In the simplest setup just one control group exists which follows the traditional teaching approaches, in a more advanced setup also control groups that user different language learning tools or some form of "placebo" can be introduced.

The separation into groups is a delicate matter and can either be solved in a fair way by randomizing the assignment or several language classes on the same level take the experiment at the same time and each complete class gets assigned one of the groups. Random sampling however has the disadvantage that it does not guarantee for representative sampling. One way of accounting for these problems would be to shuffle the group assignments after some time.

5.3.1.4 Evaluation

For a meaningful experiment we have to guarantee for its suitability. This includes validity and reliability as well as replicability. Validity again can be separated into internal and external validity (Bryman, 2012, p. 69).

For internal validity a causal relationship between the change in learning outcome as well as learner attitude and the use of the application has to be shown. To do that, we compare the results in the treatment group with the results in the control groups. Furthermore we have to control for the background variables and check the plausibility of the student's self-assessment.

External validity, the generalizability from this experiment to the general case, depends on the independence of the sample of participants. This can be solved by using reasonable sampling methods or, in the worst case, by using the whole population.

The feature of reliability depends on the measures selected for the variables. A measure has to be chosen in a way that repeated measurements do not result in a change in the result. This itself depends on stability and internal reliability.

Following up on this, stability is hard to accomplish outside the natural sciences, where for example the temperature should not be influenced significantly by measuring it. In experiments involving humans, stability is much more of a challenge. Some measures are by their nature more stable than others: Translation speed might vary, but only within a certain range, while subjective self-assessment can vary a lot between two measurements depending on external factors. So the challenge is to find a balance between stability and applicability. The subjective features are measured in parallel questions with multiple-indicator measure in the form of Likert-style question answering. In combination with Cronbach α we can test for reliability. However, we also combine them with more empirical and stable features like the placement tests.

By designing the experiment as objective as possible we aim for a high level of reproducibility.

5.3.2 Pilot

Based on these design ideas for a complete experimental evaluation we conducted a pilot in collaboration with a Latin teacher in an introductory course for Latin on the university level.

In the beginning 6 students out of the total number of 10 students in the course agreed to participate and answered the first questionnaire. Over the next two weeks the students had access to a first lesson but due to a mismatch between the material in the application and the syllabus none of the students used this opportunity. After two weeks a second lesson was enabled and one student started using the system. After four weeks, at the end of the course, only 4 out of the 10 students were still in class, two of which were candidates for the experiment and also answered the second questionnaire.

In retrospect we can identify several points that have to be improved for a full experiment:

- A too short preparation time led to technical problems
- A mismatch between the teaching material and the syllabus lead to a certain level of reluctance among the teacher and the students
- The complete population we could get candidates from was too small for significant results
- Among students of history and classical antiquity there is an increased aversion against using computers

Some points, like the aversion against computers, are hard to solve, but it seems imperative to get access to a larger group of participants and to match the teaching material with the syllabus. Both and some of the other problems can be solved with a sufficient amount of preparation.

Mandatory information:

Did you attend Latin classes before?	Yes	☐	No	☐	
If you answered yes in the previous question, where?	Gymnasieutbildning			☐	
	Grundläggande högskoleutbildning			☐	
	Private			☐	
	Other _____			☐	
	Very Good	Good	Neutral	Bad	Very Bad
How would you rate your previous knowledge about Latin?	☐	☐	☐	☐	☐
How would you rate your skills in translating Latin?	☐	☐	☐	☐	☐
How would you rate your skills in reading Latin?	☐	☐	☐	☐	☐
How would you rate your motivation to learn Latin?	☐	☐	☐	☐	☐
How would you rate your joy in translating Latin?	☐	☐	☐	☐	☐
How would you rate your joy in reading Latin?	☐	☐	☐	☐	☐
How would you rate your computer skills?	☐	☐	☐	☐	☐
	Completely Agree	Agree	Neutral	Disagree	Strongly Disagree
I enjoy learning languages	☐	☐	☐	☐	☐
I would like to learn more languages	☐	☐	☐	☐	☐
I enjoy working with the computer	☐	☐	☐	☐	☐
I use(d) some software to learn or practice a language	☐	☐	☐	☐	☐
I think computers can be useful in learning Latin	☐	☐	☐	☐	☐
I think computers make the life harder	☐	☐	☐	☐	☐
I think I know all the languages I need	☐	☐	☐	☐	☐
For what reason do you attend this Latin class?	_____				
What other languages do/-did you study?	_____				

Voluntary information:

	0 - 18	19 - 30	31 - 49	50 - 69	70 +
Age	☐	☐	☐	☐	☐
	Male	Female	Other	Not specified	
Gender	☐	☐	☐	☐	
First Language(s)	_____				

Figure 5.14: Questionnaire for user survey in the beginning

Mandatory information:

How would you rate your knowledge about Latin?	☐	☐	☐	☐	☐
How would you rate your skills in translating Latin?	☐	☐	☐	☐	☐
How would you rate your skills in reading Latin?	☐	☐	☐	☐	☐
How would you rate your motivation to learn Latin?	☐	☐	☐	☐	☐
How would you rate your joy in translating Latin?	☐	☐	☐	☐	☐
How would you rate your joy in reading Latin?	☐	☐	☐	☐	☐
	Very Much	A Lot	Some	A Little	Not At All
How do you like the idea of such an application	☐	☐	☐	☐	☐
How much did you use the application	☐	☐	☐	☐	☐
If you used the application, what did you like?	_____				
If you didn't like the application, why?	_____				
Other comments?	_____				
	_____				

Figure 5.15: Questionnaire for user survey in the end

Chapter 6

Summary

To conclude this thesis we present a summary, both of the work we did as well as additional topics that seem relevant for future exploration. In the conclusion we show how we were able to address the research objectives posed and presented in the introduction and move on from there to the list of relevant and interesting topics that can be the foundation for future work.

We first revisit the work we have done. After that we move on to the large potential of future work. That includes both theoretic and practical work in our approach to language learning. Furthermore, work on the evaluation of the Latin RGL will be a relevant future contribution.

6.1 Conclusion

We started the thesis with a list of research questions and the motivation behind it. The main aspects of this work are the general description of a language learning framework and a concrete implementation of these ideas for the specific use case of teaching Latin.

This framework should have properties including being suitable for low-resourced languages and providing a high degree of control and confidence. In Chapter 3 we presented a language learning framework based on grammar-backed and word-based sentence modification in combination with formal grammars closely related to the concept of Controlled Natural Languages. Using fully formalized grammars for language learning guarantees the intended properties.

As a necessary prerequisite for the final application we describe the Latin grammar as part of the Resource Grammar library in Chapter 4, which provides an indispensable resource for the development of our application-specific grammars.

Based on these general ideas and in combination with the Latin grammar as a resource we developed a concrete language learning application for Latin that can be used in combination with the traditional closed classroom setting. Important aspects including an experimental evaluation of the system have been presented in Chapter 5.

Finally we can conclude that we addressed all research objectives we presented in the beginning. However, some very important points could not yet be fully addressed, like the full evaluation which stays with high priority on the list of topics for future research.

6.2 Future Work

Besides the most relevant task of conducting a full evaluation of our application, we identified two main areas in which future work seems fruitful. That is on the one hand work that is more theoretical in nature and on the other hand work that directly improves the learning experience using MULLE. On a side track the work on the Latin grammar will continue and a complete evaluation of its quality will be required at some point.

All in all, even though we addressed all research objectives we presented in the introduction, the project still has plenty of potential to grow in many different directions.

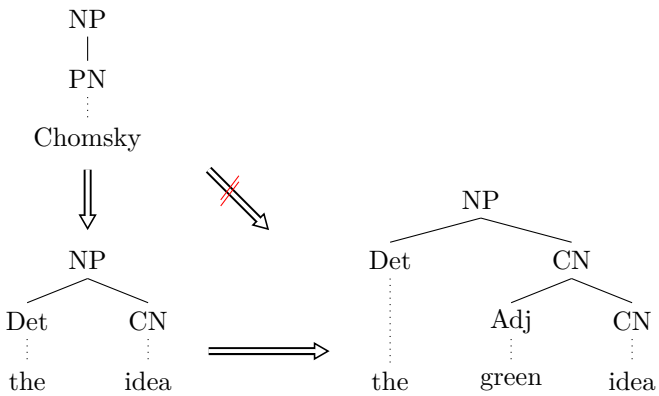


Figure 6.1: The changes that lead to the tree on the bottom right should not be part of the suggestions because it can be generated in the two separate steps shown

6.2.1 Grammar Extraction

In the description of how we can extract grammars from a textbook lesson in Section 3.3.2 we presented three steps from a text fragment to a lesson grammar. The first step always requires some manual work to select the lesson vocabulary and the second step works automatic given the resource grammar and the vocabulary list. However, the third step seems to have potential for future development. There are several different ways to extract grammar rules from a set of syntax trees. Depending on the fine-grainedness of the grammar rules the grammar can either be over-generating or over-specific. Here an automatic process would be desirable that can find a good balance between these two extremes.

6.2.2 Tree Generation

More theoretical work will focus on the grammar-based text modification. The main problem with this method is the computational complexity of creating candidate trees to create the suggestion menus and editing options. For the small grammar sizes that were involved so far the problem could still be tackled with rather simple methods. However, a well-founded algebra of tree operations that prohibits the generation of unnecessary trees in the first place would be desirable. The main group of unwanted trees are trees that can be created in a "simpler" way, i.e. we want to exclude trees from generation that are equivalent to trees that can be created by combining several simple steps. An example can be seen in Figure 6.1, where we do not want to suggest the change from the tree on the top to the tree on the bottom right because we can go via the tree at the bottom left.

A different perspective on the tree generation would be to look at it as a modified version of chart parsing, where as many nodes of the previous tree as possible should remain in the chart while adding and changing nodes to generate new trees. It also remains to be seen in the future how these different perspectives are related to each other.

6.2.3 Exercise Types

The more practical aspects remaining for future work are manifold. One direction involves the creation of additional exercise types. Translation exercises may form a good foundation for language learning, especially for languages where the primary learning outcome is translation competence. So especially for Latin that seems sufficient. But also Latin as a strongly inflecting language requires a focus on morphology. As we already alluded in Chapter 3 it is possible to relax the grammatical restrictions on agreement between constituents in a sentence to allow exercising morphological knowledge.

Another interesting form of language learning exercises would be a more multi-modal and multi-medial exercise combining both text and images or text and sound recordings, for example as an image description task or for listening exercises. Even though a large focus of GF is on grammars for natural language text, it is by far not limited to that. Several examples of grammars for graphics and other media can be found, e.g. a grammar to generate fractals (Ranta, 2011, p. 196). A similar approach could be used to either generate images including spatial relationships or to generate queries to find images with the desired content.

6.2.4 Exercise Generation

Independent of the type of exercises, the task of exercise generation has to be explored. To provide high flexibility and reliability without repetition requires a way to automatically generate exercises in a "good" way. At the moment a semi-supervised method, where a teacher selects a set of suitable exercises from which we can generate more suitable exercises, seems most practical. The problem even seems related to the selection of good examples in the creation of lexica (Kilgarriff et al., 2008) even though there are most likely differences between good examples for a lexicon and good language learning exercises.

6.2.5 Additional Languages

The broadest field of potential extensions is the addition of languages as well as going from a closed classroom-related setting to a general and open online setting. The first part might be especially useful for a wide-scale evaluation because for most languages a larger population of language learners is available than for Latin.

6.2.6 User Interaction

Some aspects of the user interaction are not yet very intuitive. Currently several improvements are work in progress that help remedy these problems. The first extension of the approach presented here is to represent the current state of the system not with exactly one abstract syntax tree. Instead we use a set of syntax trees which give rise to the same linearization as a representation and operate on all of them in parallel. This also changes the goal from matching two trees to selecting a string which gives rise to a set of trees, one of which is the same as the tree given as the goal.

A major challenge in the context of user interaction is the handling of insertions. At the moment insertion is only possible on the level where the syntax tree directly licenses it. This leads to the awkward situation in the last step of the description in Section 5.1.1. Here the user would require some knowledge about the syntax tree and the grammar to understand the behavior of the system. An improved semantics of insertions that does not purely depend on the syntax tree and the selected node could avoid this problem.

Finally, the challenge of presenting the suggestions to the user remains. That involves on the one hand a good organization of the options within the menu but also filtering out all unwanted options. That can involve several cases: replacing a phrase by itself, replacement like the ones described in Section 6.2.2 which can be split into simpler parts but also suggestions that have already been presented further down in the tree.

6.2.7 Additional Subjects

A bit more remote from the current work on Computer-Assisted Language learning but also a promising field for future work would be to broaden our perspective beyond natural languages and include work on teaching and learning more formal types of languages like the language of mathematics, programming languages or logic proof systems like natural deduction. There is already some previous work using various approaches (Caprotti and Seppälä, 2006; Gerdes et al., 2017; Villadsen et al., 2017), but it would be tempting to transform them into the MUSTE framework to make it a more general authoring tool for educational applications.

6.2.8 Grammar Testing for the RGL

As we continue working with Latin for the moment, the Latin resource grammar also stays in the focus for future work and provides potential for extensions. To speed up the process of extending the coverage of the abstract syntax, an evaluation of the current status should be a first step. We can imagine methods for evaluating the grammar on several levels. Now that we added additional lexical resources like Whitacker's Words dictionary (Whitaker, 2006) we can compare the implementation of the morphology to the state-of-the-art (Springmann et al., 2016).

Another promising approach to test GF Grammars in general is *gftest* (Listenmaa and Claessen, 2018), a general test tool that adopts methods for testing software to natural language grammars.

Finally, the access to corpora can help to see which of the missing constructions are most common. This helps focusing the work on the more relevant parts and can be supported by *ud2gf* (Ranta and Kolachina, 2017), a tool to convert dependency trees from the Universal Dependency treebank into GF abstract syntax trees. In combination with the inverse direction *gf2ud* (Kolachina and Ranta, 2016) it is also possible to not only evaluate the ratio of the treebank that is covered by the Latin RGL but also the precision by comparing it back to the annotated treebank.

Bibliography

Elnaz Abolahrar. Multilingual Grammar-Based Language Training: Computational Methods and Tools. Master's thesis, Chalmers University of Technology, 2011.

Geert Adriaens and Dirk Schreors. From COGRAM to ALCOGRAM: Toward a Controlled English Grammar Checker. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 595–601, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. URL `https://doi.org/10.3115/992133.992163`.

I. Elaine Allen and Christopher A. Seaman. Likert Scales and Data Analyses, 2007. URL `http://asq.org/quality-progress/2007/07/statistics/likert-scales-and-data-analyses.html`. Accessed 08.08.2017.

David Bamman and Gregory Crane. The Design and Use of a Latin Dependency Treebank. In Jan Hajic and Joakim Nivre, editors, *Proceedings of the Fifth International Treebanks and Linguistic Theories Conference*, pages 67–78, Prag, 2006. Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University. URL `http://hdl.handle.net/10427/42684`.

Stephen Bax. CALL — Past, Present and Future. *System*, 31(1):13 – 28, 2003. ISSN 0346-251X. URL `http://www.sciencedirect.com/science/article/pii/S0346251X02000714`.

Karl Bayer and Josef Lindauer, editors. *Lateinische Grammatik*. C.C. Buchners Verlag, J. Lindauer Verlag, R. Oldenburg Verlag, 2. Edition, auf der Grundlage der Lateinischen Schulgrammatik von Landgraf-Leitschuh neu bearbeitete edition, 1994.

John V. Becker, Daniek E. Hinton, and Hugh G. Anderson JR. Braille Computer Monitor, March 2004. URL `http://www.freepatentsonline.com/6700553.html`. US Patent 6700553.

Alan Bryman. *Social Research Methods*. Oxford University Press, Great Clarendon Street, Oxford, 4th edition, 2012.

Olga Caprotti and Mika Seppälä. Multilingual Delivery of Online Tests in Mathematics. In *Proceedings of Online Educa*, Berlin, 2006.

Noam Chomsky. *Syntactic Structures*. Mouton de Gruyter, Berlin, New York, 1957. Reprint 2002.

Noam Chomsky. *Lectures on Government and Binding. The Pisa Lectures*. Foris Publication, Dordrecht- Holland/Providence Rl- U.S.A., 1988.

Octavian Ciobanu. An Evaluation of the Ergonomics of the QWERTY Keyboards. In *Engineering Solutions and Technologies in Manufacturing*, volume 657 of *Applied Mechanics and Materials*, pages 1051–1055. Trans Tech Publications, 11 2014. URL https://www.scientific.net/AMM.657.1051.

Computer History Museum. PLATO@50: PLATO Computer Learning System 50th Anniversary, June 2010. URL https://www.youtube.com/watch?v=THoxsBw-UmM. Accessed 08.07.2018.

Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper & Row, New York, 1990.

Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15, New York, 2011. ACM.

Grégoire Détrez and Aarne Ranta. Smart Paradigms and the Predictability and Complexity of Inflectional Morphology. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 645–653, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2380816.2380895.

Sara Ehrling. *Lingua Latina novo modo – En nybörjarbok i latin för universitetsbruk*. University of Gothenburg, 2015.

Torsten Felzer, Ian Scott MacKenzie, and Stephan Rinderknecht. Efficient Computer Operation for Users with a Neuromuscular Disease with OnScreen-DualScribe. *Journal of Interaction Science*, 2(2), 2014.

Karën Fort, Bruno Guillaume, and Hadrien Chastant. Creating Zombilingo, a Game with a Purpose for Dependency Syntax Annotation. In *Proceedings of the First International Workshop on Gamification for Information Retrieval*, GamifIR '14, pages 2–6, New York, 2014. ACM. URL http://doi.acm.org/10.1145/2594776.2594777.

Wendy Fox-Turnbull. Stimulated Recall using Autophotography. a Method for Investigating Technology Education. *Strengthening the position of technology education in the curriculum*, 2009.

Ignacio Garcia. Learning a Language for free while Translating the Web. does Duolingo work? *International Journal of English Linguistics*, 3(1):19, 2013. URL https://doi.org/10.5539/ijel.v3n1p19.

Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas van Binsbergen. Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. *International Journal of Artificial Intelligence in Education*, 27(1): 65–100, Mar 2017. URL https://doi.org/10.1007/s40593-015-0080-x.

Gregory R. Crane, editor. Perseus Digital Library, 2018. URL http://www.perseus.tufts.edu. Accessed 09.06.2018.

Normunds Gruzitis and Dana Dannélls. A multilingual FrameNet-based Grammar and Lexicon for Controlled Natural Language. *Language Resources and Evaluation*, 51(1):37–66, Mar 2017. ISSN 1574-0218. URL https://doi.org/10.1007/s10579-015-9321-8.

André Kenji Horie. Rewriting Duolingo's Engine in Scala. http://making.duolingo.com/rewriting-duolingos-engine-in-scala, January 2017. Accessed 04.04.2018.

Peter Høeg. *Miss Smilla's feeling for snow*. Harvill, London, 1993a. Translation by F. David.

Peter Høeg. *Fröken Smillas känsla för snö*. Nordstedts, Stockholm, 1993b. Reprint 1997, Translation Ann-Mari Seeberg.

Peter Høeg. *Frøken Smillas fornemmelse for sne*. Rosinante, København, 6. paperback edition, 2006.

Hasan Kaya and Gülşen Eryiğit. Using Finite State Transducers for Helping Foreign Language Learning. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 94–98, 2015.

Adam Kilgarriff, Miloš Husák, Katy McAdam, Michael Rundell, and Pavel Rychlý. GDEX: Automatically Finding Good Dictionary Examples in a Corpus. In Elisenda Bernal and Janet DeCesaris, editors, *Proceedings of the 13th EURALEX International Congress*, pages 425–432, Barcelona, Spain, jul 2008. Institut Universitari de Linguistica Aplicada, Universitat Pompeu Fabra.

Martin T. King, Dale L. Grover, Clifford A. Kushler, and Cheryl A. Grunbock. Reduced Keyboard Disambiguating System, January 2000. URL http://www.freepatentsonline.com/6011554.html. US Patent 6011554.

Prasanth Kolachina and Aarne Ranta. From Abstract Syntax to Universal Dependencies. In *Linguistic Issues in Language Technology (LiLT)*, volume 13. CSLI, Stanford, August 2016. URL http://csli-lilt.stanford.edu/ojs/index.php/LiLT/article/download/71/73.

Tobias Kuhn. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170, March 2014.

Anuj Kumar, Tim Paek, and Bongshin Lee. Voice Typing: A New Speech Interaction Model for Dictation on Touchscreen Devices. In *Proceedings of CHI 2012, SIGCHI Conference on Human Factors in Computing Systems*, Austin, Texas, 2012.

Clifford A. Kushler and Randal J. Marsden. System and Method for Continuous Stroke Word-based Text Input, July 2004. URL http://www.freepatentsonline.com/y2004/0140956.html. US Patent 20040140956.

Herbert Lange. Erstellung einer Grammatik für das Lateinische im "Grammatical Framework". Master's thesis, Ludwig-Maximilians-Universität, Munich, 2013.

Herbert Lange. Implementation of a Latin Grammar in Grammatical Framework. In *DATeCH2017*, Göttingen, Germany, 2017.

Herbert Lange and Peter Ljunglöf. MULLE: A Grammar-based Latin Language Learning Tool to Supplement the Classroom Setting. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA '18)*, pages 108–112, Melbourn. Australia, 2018a. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W18-3715`.

Herbert Lange and Peter Ljunglöf. Putting Control into Language Learning. In *SIGCNL 2018*, Maynooth, Ireland, 2018b.

Michael Levy. *Computer-Assisted Language Learning. Context and Conceptualization.* Claredon Paperback, Oxford, 1997.

Josef Lindauer, Klaus Westphalen, and Bernd Kreiler. *Roma, Ausgabe C für Bayern, Bd.1.* C.C. Buchner, 2000.

Inari Listenmaa and Koen Claessen. Automatic Test Suite Generation for PMCFG Grammars. In *Proceedings of the Fifth Workshop on Natural Language and Computer Science*, Oxford, 2018. URL `https://doi.org/10.29007/3p48`.

Peter Ljunglöf. *Expressivity and Complexity of the Grammatical Framework.* PhD thesis, Göteborg University, 2004.

Peter Ljunglöf. Editing Syntax Trees on the Surface. In *Nodalida'11: 18th Nordic Conference of Computational Linguistics*, Rīga, Latvia, 2011.

Lisa N. Michaud. King Alfred: A Translation Environment for Learners of Anglo-Saxon English. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 19–26, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W08-0903`.

Richard Montague. English as a formal language. In Richmond H. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague*, pages 188–221, New Haven and London, 1974. Yale University Press.

Maria Moritz, Barbara Pavlek, Greta Franzini, and Gregory Crane. Sentence Shortening via Morpho-Syntactic Annotated Data in Historical Language Learning. *Journal on Computing and Cultural Heritage (JOCCH)*, 9(1):3, 2016.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth*

International Conference on Language Resources and Evaluation (LREC 2016), 2016. URL http://www.lrec-conf.org/proceedings/lrec2016/pdf/348_Paper.pdf.

Charles Kay Ogden. *Basic English: A General Introduction with Rules and Grammar*. Paul Treber, London, 1930.

Thomas Edward Payne. *Describing Morphosyntax : A Guide for Field Linguists*. Cambridge University Press, Cambridge, 1997.

Harm Pinkster. *Latijnse syntaxis en semantiek*. Grüner, Amsterdam, 1984. Revised 1988 Lateinische Syntax und Semantik, Tübingen: Francke, 1990 Latin Syntax and Semantics, London: Routledge.

Harm Pinkster. *The Oxford Latin Syntax. Volume I: The Simple Clause*. Oxford University Press, Oxford, 2015.

Aarne Ranta. The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2(2), December 2009. URL https://journals.linguisticsociety.org/elanguage/lilt/article/view/214/158.html.

Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011.

Aarne Ranta and Prasanth Kolachina. From Universal Dependencies to Abstract Syntax. In *Proceedings of the 1st Workshop on Universal Dependencies*. Linköping University Electronic Press, May 2017.

Aarne Ranta, Ramona Enache, and Grégoire Détrez. Controlled Language for Everyday Use: The MOLTO Phrasebook. In Michael Rosner and Norbert E. Fuchs, editors, *Controlled Natural Language*, pages 115–136, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Hanumant Redkar, Sandhya Singh, Meenakshi Somasundaram, Dhara Gorasia, Malhar Kulkarni, and Pushpak Bhattacharyya. Hindi Shabdamitra: A WordNet based E-Learning Tool for Language Learning and Teaching. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 23–28. Asian Federation of Natural Language Processing, 2017. URL http://aclweb.org/anthology/W17-5904.

Hans Reichenbach. *Elements of Symbolic Logic*. The Macmillan Company, New York, 1947.

Sandra J. Savignon. Communicative Language Teaching. *Theory Into Practice*, 26(4):235–242, 1987. URL https://doi.org/10.1080/00405848709543281.

Asad Sayeed and Stan Szpakowicz. Developing a Minimalist Parser for Free Word Order Languages with Discontinuous Constituency. In Rafael Muñoz José Luis Vicedo, Patricio Martínez-Barco and Maximiliano Saiz, editors, *Advances in Natural Language Processing, 4th international conference (EsTAL)*, Lecture Notes in Computer Science 3230. Springer-Verlag, 2004.

Uwe Springmann, Helmut Schmid, and Dietmar Najock. LatMor: A Latin Finite-State Morphology Encoding Vowel Quantity. *Open Linguistics*, 2(1): 386–392, 2016. URL https://doi.org/10.1515/opli-2016-0019.

Mark Steedman. Combinatory Categorial Grammar. An Introduction. Draft August 25th, 2016, 2016.

Penelope Sweetser and Peta Wyeth. GameFlow: A Model for Evaluating Player Enjoyment in Games. *Computers in Entertainment (CIE)*, 3(3):3–3, July 2005. URL https://dl.acm.org/citation.cfm?id=1077253.

Ayumi Takahashi and Hideki Takahashi. Anxiety and Self-Confidence in Ancient Language Studies. *Niigata University Language and Culture Research Department Bulletin*, 8 2015.

Roumen Vesselinov and John Grego. Duolingo Effectiveness Study. Technical report, 12 2012. URL http://static.duolingo.com/s3/DuolingoReport_Final.pdf. Accessed 16.10.2017.

Vicipaedia. Libera Enceclopaedia, 2018. URL https://la.wikipedia.org/wiki/Vicipaedia:Pagina_prima. Accessed 09.07.2018.

Jørgen Villadsen, Alexander Birch Jensen, and Anders Schlichtkrull. NaDeA: A Natural Deduction Assistant with a Formalization in Isabelle. *IfCoLog Journal of Logics and their Applications*, 4(1):55–82, 2017.

Elena Volodina, Ildikó Pilán, Lars Borin, and Therese Lindström Tiedemann. A Flexible Language Learning Platform Based on Language Resources and Web Services. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

Luis Von Ahn. Games with a Purpose. *IEEE Computer*, 39(6):92–94, 2006.

David J. Ward, Alan F. Blackwell Y, and David J. C. Mackay Z. Dasher: A Gesture-Driven Data Entry Interface for Mobile Computing Human-Computer Interaction. *Human-Computer Interaction*, page 228, 2002.

Mark Warschauer. Technological Change and the Future of CALL. In S. Fotos & C. Brown, editor, *New Perspectives on CALL for Second and Foreign Language Classrooms*, chapter Technological change and the future of CALL., pages 15–25. Lawrence Erlbaum Associates., Mahwah, NJ, 2004.

William Whitaker. Words. Latin-English Dictionary Program, 2006. URL http://archives.nd.edu/whitaker/words.htm. Accessed 15.7.2018.

Mary McGee Wood. *Categorial Grammar*. Linguistic Theory Guides edited by Richard Hudson. Routledge, London and New York, 1993.