

Univerzita Karlova v Praze, Filozofická fakulta
ÚSTAV ANGLICKÉHO JAZYKA A DIDAKTIKY

Název práce: Fonetické změny mezi starou a střední angličtinou (algoritmické zpracování)

Title of Thesis: Phonological Changes between Old and Middle English (An Algorithmic Approach)

Bakalářská práce / BA Thesis

Zpracoval: Daniel Marek

Vedoucí bakalářské práce: Mgr. Ondřej Tichý

Praha, září 2010

Chtěl bych poděkovat:

Mgr. Ondřejovi Tichému za skvělé vedení, trpělivost a inspiraci

Daniele Kolbe z univerzity v Trieste za sehnání fonetických materiálů [Strauss(1999)]

Své přítelkyni a rodině za trpělivost s mým nepořádkem, který během tvorby vznikl jako vedlejší produkt kreativity

Všem ostatním, co mne naučili různými lingvistickými či inforatickými dovednostem (učitelé z gymnázia, FF i MFF UK, vedoucí z KSP), které jsem upotřebil v této práci

Abstract

The purpose of this thesis was to create an automatic analyzer of phonetic changes in the historical development of English, namely between Old and Middle English. The analyzer gets an OE and ME form of a word at the input and produces a suggestion of an explanatory sequence of changes connecting these two forms as the output. As sound changes operate mainly on the spoken language, three main subtasks emerge: to create a grapheme-to-phoneme convertor for Old English, the same (although structurally more complicated) for Middle English and a core algorithm that searches for possible sequences that would explain the development of the word between its OE and ME spoken forms.

There is a certain regularity in the diachronic phonetic changes, and therefore these can be translated into a set of rules. A general framework is proposed and implemented that works with these rules in a certain formalized fashion that allows their integration within standard algorithms.

The form of the rules is largely adopted from regular expressions, with some alterations and additions, the most important being the possibility of using wildcards that are based on phonetic properties. A database of phoneme representations and phonetic properties is taken from Kirshenbaum's 1992 proposal for ASCII representation of IPA symbols.

In the case of the Old English pronunciation, the set of rules describes context-dependent variants of grapheme-to-phoneme relationships. These rules are taken in an established order and each rule is applied exactly once to rewrite the input word and pass it on to the next rule. The quality of the IPA output of this deterministic orthography-to-pronunciation translator is satisfactory, because Old English had a relatively good level of standardization and regularity of writing.

In the case of Middle English, the situation is more complex, because of a lower level

of standardization and regularity and thus the system for translation to IPA has to be reused in a more complex manner. The resulting string does not describe one suggested pronunciation, but rather a whole set of possible pronunciations which are ranked according to their probable feasibility.

An implicit graph is defined, in which the nodes are possible spoken forms and edges represent known sound changes. Such path in the graph is found, that connects the Old English pronunciation with one of the proposed Middle English pronunciations and this path is presented as a sequence possibly explaining the diachronic development. The algorithm works with various metaparameters, such as the approximate dating of the sound changes, node weight (a penalty that disadvantages certain changes from appearing in the explanation, because they are thought improbable) evaluation of the intermediate word forms etc.

The algorithm or its parts can be used in various ways, ranging from didactic purposes to the incorporation of the grapheme-to-phoneme convertor into dictionaries or corpora in order to enhance them with phonetic search capabilities etc. Parts of the system are also relatively universal and can be reused for the treatment of different historical periods or even languages.

Keywords: Old English, Middle English, phonetics, phonetic development, formalization

Abstrakt

Cílem této bakalářské práce bylo vytvořit automatický analyzátor diachronních fonetických změn v angličtině, který je konkrétně zaměřen na změny mezi staroangličtinou a střední angličtinou. Vstupem programu jsou dvě slova: jedno je staroanglické a druhé je jeho pozdější verze ve střední angličtině. Vzhledem k tomu, že tyto změny probíhají na mluvené formě jazyka, vyplývá nám ze zadání několik dílčích úkolů: vytvořit automatický převaděč ze staroanglické psané formy do odpovídajícího přepisu výslovnosti, totéž pro střední angličtinu (což je ale díky povaze tehdejšího pravopisu strukturně složitější) a nakonec algoritmus, který mezi navrženými výslovnostmi pro daná období nalezne cestu ve formě pravděpodobné sekvence fonetických změn, která by vysvětlovala vztah mezi dvěma verzemi slova, které jsme dostali na vstupu.

Ve fonetických změnách panuje určitá pravidelnost a tudíž je lze z velké části popsat pomocí sady pravidel. Tato pravidla jsou v této práci formalizována tak, aby bylo možno je později zkombinovat s funkcionalitou standardních inforatických algoritmů. Reprezentace těchto pravidel vychází ze syntaxe regulárních výrazů s určitými úpravami a přidanými funkcemi. Nejdůležitější přidanou funkcionalitou je zde podpora pro fonetické žolíky (wildcards), a systém tedy kromě možností standardních regexových speciálních znaků umožňuje např. snadno vyjádřit, že se jedná o libovolnou znělou frikatívu. Fonetické změny jsou konstruovány jako přepisovací pravidla.

Převaděč z psané do mluvené formy staroangličtiny je konstruován jako sada těchto pravidel, která se v určeném pořadí aplikují na vstupní slovo. Díky relativně pravidelnému a standardizovanému vztahu mezi psanou a

mluvenou formou ve staroangličtině, jsou zde výsledky dobré.

U střední angličtiny není pravopis zdaleka tak standardizovaný a je třeba pracovat s určitou dávkou nejistoty. Výstupem převaděče pro střední angličtinu tedy není jedna navrhovaná forma výslovnosti, ale řetězec obsahující popis více možných forem, které jsou navíc ohodnocené podle odhadované pravděpodobnosti. Tohoto je dosaženo vytvořením složitějších prepisovacích pravidel, která jsou nicméně přesto vyhodnocována technicky stejně jako v případě staroangličtiny.

Z možných tvarů slov a seznamu fonetických změn se vytvoří graf, ve kterém je následně vyhledána cesta ze staroanglické do jedné z navrhovaných středoanglických forem. Tato cesta, pokud je nalezena, potom umožňuje stanovit posloupnost fonetických změn, která by měla popisovat vývoj slova. Algoritmus bere v potaz pravděpodobnost výskytu změn, jejich přibližnou dataci, ohodnocení pravděpodobnosti legitimacy tvarů slov apod.

Program (či jeho části) má mnoho využití, od didaktických záměrů až např. po integraci se staroanglickými slovníky či korpusy, u kterých ho lze použít k automatickému převedení textů do fonetického přepisu, čímž se umožní vyhledávání na základě fonetických kritérií. Upravením dat lze program adaptovat i např. k použití pro stejné účely v jiných časových obdobích, či dokonce pro jiné jazyky.

Klíčová slova: staroangličtina, střední angličtina, fonetika, fonetický vývoj, formalizace

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a že jsem uvedl všechny použité prameny a literaturu.

I declare that the following BA thesis is my own work for which I used only the sources and literature mentioned.

V Praze dne 25. srpna 2010 / Prague, 25th August 2010

Daniel Marek

Souhlasím se zapůjčením bakalářské práce ke studijním účelům.

I have no objections to the BA thesis being borrowed and used for study purposes.

Contents

1	Introduction: The Scope and Limitations of the Thesis	10
1.1	The Kind of Language Change to be Described	10
1.2	The Articulatory Point of View	12
1.3	The Generativist Description: Optimality Theory	13
1.4	Algorithmic Analysis of a Development of Changes in a Word	14
2	Methodology	15
2.1	Features of the computer program	15
2.2	Encoding and representation	16
2.3	The core of the program	18
2.4	Expected complications	20
3	Phonological Changes between Old and Middle English	22
3.1	The Syntax of the Rules	22
3.2	Pronunciation of Old English	26
3.3	Pronunciation of Middle English	30
3.4	Phonological Changes between OE and ME	35
4	Technical Information	40
4.1	System Requirements to Run the Program	40
4.2	Program interface	41
4.3	A Website Presenting the Results of the Experiment	43
5	Possible Ways of Utilization	44
5.1	Reusability & Compatibility	45
5.2	Adaptability: A Different Language	45

5.3	Didactic Purposes	46
5.4	Automated Analyses	47
6	Resumé	48
6.1	Využití softwaru či jeho částí	51
7	Glossary of technical terms	53
8	Miscellaneous Data Samples	55
8.1	The Phonemic Inventory	55
8.2	The Old English Pronunciator Sample Output	58
8.3	Sample Output of the Development Analyzer	60
9	Enclosure list	61

1 Introduction: The Scope and Limitations of the Thesis

1.1 The Kind of Language Change to be Described

There are many aspects of the change that English had undergone between the Old English and Middle English eras, both in terms of the kinds of sources of influence (ranging from phonetic pressures to political circumstances) and in terms of what level or aspect of the language was being altered (e.g. syntax, lexis, etc.).

Millward provides a basic classification of the factors that trigger a change in a language: she makes the distinction between the internal and external motivations [Millward(1988), page 10].

The internal motivations are grounded in the structure of the language itself, and are not unrelated to language changes in other languages, since various concepts, such as a language's tendency to maintain a certain level of efficiency (e.g. the principle of least effort [Millward(1988), page 8]) or redundancy in expressing different elements, is something that can be traced across multiple languages, and the scope and nature of which should be similar in languages that are typologically similar. The motivations for the internal changes can be thus explained by phenomena such as phonetic factors, because adjacent sounds have the tendency to influence each other. However, as to the results of these internal changes, they are not limited to phonetic or phonological changes, but also sometimes appear for example at the lexical level of the language (such as

Millward's example of the latter clash of OE *lætan* with *lettan* and its subsequent semantic/lexical implications [Millward(1988), page 11]).

The external pressures for the language change tend to include the influence of other languages, in one form or another: they are often stimulated by political circumstances, such as when a nation is subdued by another one, or when a dialect reaches a more prominent social status than other dialects have. A result of the external change can be for example a change in the lexicon through loanwords, but structural changes, as for example an alteration of syntax or morphology by the influence of another language is also possible, e.g. when the speakers of the two languages live in competing conditions, the political struggle can have a linguistic impact, as in the case of English and French after the Norman conquest [Millward(1988), page 120-124].

The thesis concentrates on changes caused by internal pressure, namely changes resulting from phonetic and phonological properties of groups of sounds within words. Both systemic and sporadic changes will be taken into account, and the main objective of the thesis is to formalize them in the following manner:

1. A system of representation of rules of phonetic changes will be devised, so that rules represented in this manner will be easily adaptable for using with standard computer programming techniques.
2. The rules describing phonetic changes between Old and Middle English will be extracted mainly from secondary sources, adapted and transcribed into the proposed system of representation.
3. An algorithm will be designed and implemented, that will get two words as the input: an Old English and Middle English version of

a word. It will then attempt to suggest a series of phonetic changes that had lead to the gradual transformation of the Old English word to its later counterpart. The computation will employ the knowledge of already known sound changes.

4. A webpage will be implemented that will make the work with all the elements of this process (alphabets, phonetic properties, phonetic rules) accessible to anybody without the necessity of having any knowledge of computer programming.
5. Specific ways of utilization of the system or its parts will be suggested and described.

1.2 The Articulatory Point of View

The phonetic changes that appeared between the Old and Middle English periods will be described and summarized, working chiefly with articulatory properties of speech sounds such as places and manners of their production and the impact of their combinations on the changes. Millward claims that “all the changes that occur in speech sounds can be described in articulatory terms.” [Millward(1988), page 18], which the syntax of the proposed system for describing rules of phonetic changes will take advantage of, as it will be made possible to condition the application of the rules by articulatory properties of the sounds involved.

The relationship between the written and spoken forms of both Old English and Middle English will be also described, as this knowledge will be necessary for the proposed software for language change analysis, as it will be possible to submit the input words in their written form, while the computation dealing with the sound changes will operate on their

spoken form.

1.3 The Generativist Description: Optimality Theory

One of the approaches that inspired the creation of this thesis was the optimality theory with its generativist principles. The individual changes, as described in one of the following sections, could be also modeled using the principles proposed by this theory. The optimality theory uses a system for evaluating and comparing the feasibility of different word forms, and draws from phonetic properties when constructing rules in order to achieving that [Walther(1996), pages 1-2]. Although the purpose and use of the optimality theory is different, certain of its aspects and principles were reused in this thesis.

The system of optimality theory allows for evaluating the “optimality” of how a word fits a language, based on a system of rules that work with the phonetics of the words: the rules basically take the form of a list of phonetic properties that are sorted by the importance (or appropriateness) of their occurrence (or the lack thereof) in the evaluated word [Walther(1996), pages 1-2]. The fact that the rules are sorted by their importance allows for a relative comparison of two or more words and for choosing the most probable one. Therefore, if we have several variants of a word (different in the voicing of some consonant, length or quality of a vowel, etc.), optimality theory allows us to choose the one that is most probable to exist in the language (if the set of rules is provided in an ideal manner). Adding rules that refer to similarity to a chronologically

preceding form of the word in question allows for modeling a most-likely language change instead of only measuring the extent to which a word fits into the system, in other words, it allows us to enter the diachronic dimension.

However, there are limitations of the applicability of optimality theory to the modeling of the change as a whole, since its capability of describing several subsequent changes by only one set of constraints is quite limited (not even speaking about the way this complicates the understanding of their order of appearance).

A computer model, which is capable of evaluating the optimality of words for given sets of constraints, has been devised and implemented by Markus Walther in 1996: it is built upon the standard set of UNIX shell tools [Walther(1996), pages 20-25]. The idea to model phonetic changes using the power of regular expressions was inspired by his utilization of these for modeling optimality theory rules.

1.4 Algorithmic Analysis of a Development of Changes in a Word

An attempt shall be made to create a piece of software that would be capable of analyzing the history of changes of a word. The program shall receive two words as the input: an older (Old English) and a more recent (Middle English) form of the same word. The program will present the output in the form of a set of elementary changes (ideally as they appeared chronologically) that should be the best possible explanation of the change in the word the program was capable of finding. The changes will be tracked in the spoken forms of the words, so the analyzer will also

have to be automatically capable of computing a pronunciation transcription of both Old and Middle English words based on their written form. It will be however capable of taking both phonetic transcription and orthography from the input. The implementation would assume the form of a modification of the Dijkstra's graph search algorithm.

Technical details concerning the design and implementation shall be also discussed.

2 Methodology

2.1 Features of the computer program

The program will have the form of a PHP script, structurally designed in such a way that its most CPU-consuming parts could be also easily rewritten into C++ (or solved externally in any other way). The PHP scripting language was chosen because developing software in it is faster than in a regular compiled language as C++, the tradeoff being that actual running of such a program is slower. This choice was made in order to achieve quicker development that will allow me to concentrate on the linguistic problems. C++ could be then possibly used to achieve a better processing effectivity (if necessary).

The scripts and the code will be provided for public use under the GPLv2 license, thus open-source. The interface will have the form of a web page, so that computer programming skills will not be necessary to access and use the program.

The program will be capable of the following:

1. Translating between different forms of representation of words (sev-

eral encodings of both written and spoken forms will be used: the particular choice of encodings will be described in more detail later)

2. Working with a set of rules that will describe sound changes in a form similar to regular expressions describing both the sound in question but also the neighbouring ones, using phones and phone wildcards operating on the basis of phonetic features.
3. Finding a sequence of rules that would describe and explain a feasible change from one form of a word to another one (if such a sequence exists; also, it may find more than one solution). This process will be used to determine a possible sequence of sound changes describing the development between an Old English form of a word and a late Middle English form of one.

However, reusability will be kept in mind during the creation of the program in such a way that the design of the framework will be kept as general as possible and will thus be independent of this particular language and epoch described: by changing the set of rules (and altering the phone inventory) it will be possible to describe any other language with its different history of sound development.

Also, the translation between the written and the spoken form may be represented reusing the same system for rules (only with a different set of rules) that will be used for describing the diachronic sound changes.

2.2 Encoding and representation

Choosing a fit representation of the language material in question will be crucial for the result of the experiment. There are at least two levels

of expression that will require some kind of representation (i.e. written English in different historical periods and the corresponding spoken form) and even more different encodings to capture the language material at the several levels of the process of the analysis. This list of encodings the software will operate in also mirrors the structure of the experiment.

1. Unicode representation of the written form of English - both for the Old English period and for the Middle English period.
2. ASCII representation of the same - for compatibility purposes and also to enable users easily enter non-standard characters (such as æ) from their keyboard.
3. The Unicode representation of the International Phonetic Alphabet.
4. Kirshenbaum's ASCII representation of the International Phonetic Alphabet, which this member of Hewlett-Packard laboratories proposed in 1992 in order to unite the representation of IPA symbols in the discussions on Usenet [Kirshenbaum(1992), page 1]. His encoding will be used to capture the spoken form of the words.
5. An internal alphabet that will capture the phonemes in a one-to-one relationship (one phoneme will be represented by exactly one ASCII character). This is chosen for the PHP implementation to utilize the power of regular expressions in a more efficient manner: both in terms of description and in terms of processing and memory efficiency. This will not necessarily be easily read by humans, since it is intended for internal use only and its relation to Kirshenbaum's representation will be arbitrary (but obviously constant during different computations). This internal encoding could also be used for passing the computing task to a C++ implementation, which could

increase the computing speed.

6. Kirshenbaum’s “Explicit” representation that describes phonemes as sets of phonetic features [Kirshenbaum(1992)], for example: $\{low, fnt, unr, vw1\}$, which represents the phoneme [a]. This will become useful for creating wildcards based on phonetic criteria (i.e. the definition of a wildcard for “any voiceless sonorant” will be possible).

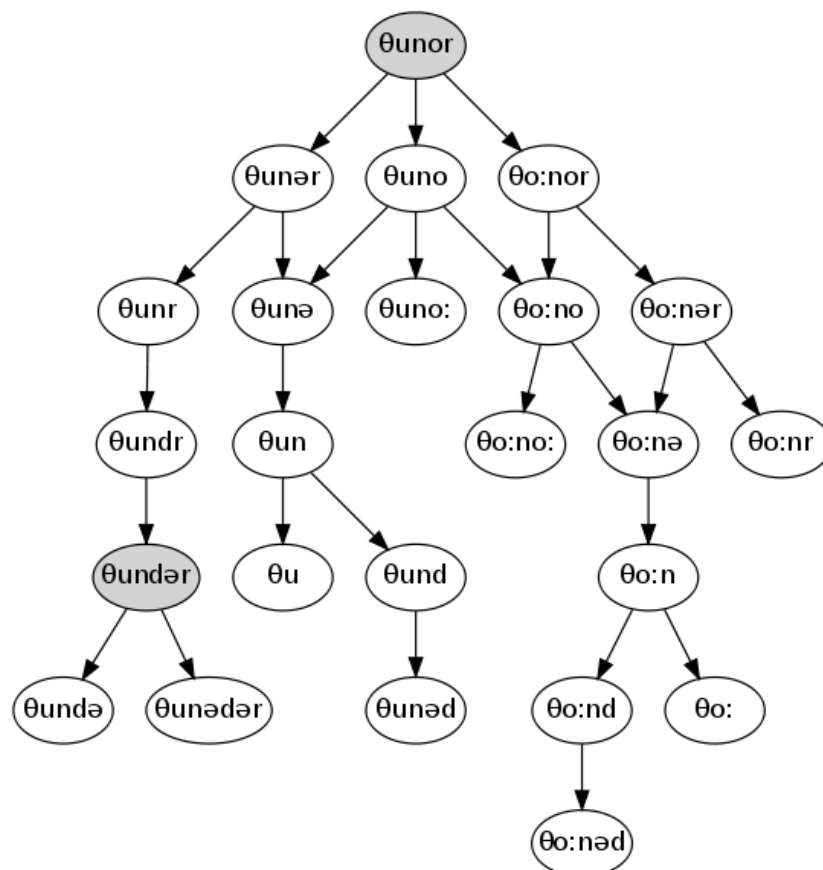
The translation between points 1. and 2. should be trivial. The creation of the third form (from the second one) will require the knowledge of pronunciation of English in the respective time periods, which will be mostly taken from secondary materials, such as Millward [Millward(1988), page 69]. The correct and efficient implementation of this is one of the major challenges. It will be a matter of mechanical translation to convert words between Unicode, Kirshenbaum and the internal encoding. The 6th form of representation should not be linguistically complicated either. The second of the two big challenges of this thesis will be to determine the set of rules that would describe the sound changes in English during the given epochs as exactly as possible.

2.3 The core of the program

Algorithmically speaking, the input of the core of the program will be the spoken form of the two words and the set of the sound changes that occurred in the given time period of the given language (English between its Old and Middle stage). The algorithm will then determine a possible sequence(s) of the changes that would connect the two forms and will be the most “optimal” at the same time (different sound changes in particular contexts will contribute to the overall sequence’s “optimality”

differently).

All possible forms of words (i.e. the infinite number of words created by all possible combinations of phonemes) can be understood as vertices forming an (implicit) graph where the set of the rewriting rules would become the (orientated and weighted) edges. In this graph, the task of the algorithm would be to find a minimal path (or more paths) from the node representing the older word to the node that represents the later one. A simple graph for a possible input (the corresponding output is presented at the end of the thesis) is shown below, the initial and final nodes being highlighted:



Please note that the graph is simplified in two ways: first, certain characteristics of the edges are left out for the sake of visual clarity of the graph (weight, date span, description of sound change) and second, the

actual (implicit) graph is larger, because the search was stopped when it was certain that no better path could be found. Each edge (displayed as an arrow) represents one sound change from our inventory and thus the development from the original word form to any form that you can see in the graph could be “explained” as the particular series of the changes that are represented as the series of arrows that form the path from the initial node to the node in question. Not all of these developments are obviously feasible and it is one of the important tasks of the project to separate the wheat from the chaff.

The problem thus described is solvable for example by the Dijkstra’s graph search algorithm. However, the reality will probably bring irregularities and complications so that a related algorithm, such as A* (graph search algorithm where the order in which new edges of the graph are explored is based on heuristic analysis) might bring better results.

2.4 Expected complications

The system will be probably rather sensitive to precise spelling which may become a problem when encountering the “chaotic state of Middle English spelling” [Millward(1988), page 78]. The necessity will probably arise to employ some fuzzy logic or a more complex set of rules to cover this spelling diversity. It may even be possible that the spelling variations will turn out too complex to be covered satisfactorily. In that case, as the aim of the thesis is to describe the sound changes and this problem is out of the scope of the current thesis, it might be put aside.

Another possible problem might arise with finding the correct set of rules describing the successive sound changes. Let us suppose that we have

the correct set of rules, and the sequence we are searching for does exist: these are the basic prerequisites for the success, and yet it is possible that there is more than one feasible sequence that describes the development of the word, which is a complication. The feasibility of the different explanations will vary, and thus it will be necessary to establish some metacriteria in order to distinguish between more and less feasible explanations: first, possibly some rough dating of the individual changes in the set of the rules will have to be taken into account, so that differing feasibility of various orders in which the changes took place could be expressed. It might be better to express them on a continuous scale of probability rather than by boolean logic. Second, some degree of limitation on the number of elementary changes in the resulting sequence may turn out to be useful, since an overcomplicated long sequence of sound changes will be surely less satisfactory as an explanation; the number of occurrences of individual sound changes may be used as another indicator of feasibility.

The computational complexity would be exponential in relation to the expected number of changes in the sequence. However, there would be another limitation as to the number of iterations: the number of the possible (reachable) states might be small enough to efficiently limit the estimated number of iterations of the Dijkstra's graph search algorithm. Depending on the complexity of the graph, some heuristics may be used to decide what nodes should be traversed first, such as edit distance or even some analysis of "probability of a word being a real one" based on its pronouncibility: being thus estimated from the alteration and sizes of vowel/consonant clusters, phonetic proximity of neighbouring sounds

etc.

3 Phonological Changes between Old and Middle English

Sound changes occur primarily in the spoken language, although there are exceptions where orthography matters [Millward(1988), page 29]. Since the available primary material is (obviously) written, it is necessary to start with the explanation (and treatment) of the relationship between the written and spoken forms of both Old and Middle English.

The description of the relationship between orthography and pronunciation will be provided as a list of particular formalized rules. These rules are taken and adapted mostly from Millward's *Biography of the English Language* and will be already provided with their representation in the system that the program uses, and thus an explanation of the rule system will now follow.

3.1 The Syntax of the Rules

The rules can be used to express different phenomena, and the project contains three basic sets of rules covering the following:

1. Phonetic changes between Old and Middle English
2. The pronunciation of Old English (i.e. the rules will cover the relationship between orthography and pronunciation)
3. Middle English Non-Deterministic Pronunciation (explained later)

Both the orthography-to-pronunciation rules and the rules describing historical changes are using the same syntax, although they take advan-

tage of it in different ways and the subsequent treatment of the sets of rules also varies. The application of a single rule is always the same, but the difference is in how each set of rules is used as a whole.

In the case of the pronunciation generator, the rules are applied in a fixed order and each one only once (if applicable). The language development rules are taken in any order, can be applied more than once and their weight is taken into account. The weight of an edge is a numeric value; the relative probability of the whole explanation will be computed as the sum of weights of all the rules applied in the explanatory sequence.

Because of the ambiguous nature of Middle English grapheme to phoneme relationship, the Middle English pronunciation generator has to use more complicated rules to allow expressing all possible pronunciations of a word at once, and also expresses the relative feasibility of each of them (explained later).

Each rule is built of several basic components: **the prefix**, **the replaced part**, **the suffix**, **the replacement** and some metadata (weight, order, description). If **the replaced part** is found in the treated word and is surrounded by **the prefix** and **the suffix** (please note that I am not using these two terms here in their linguistic sense: they just denote phonemes immediately surrounding **the replaced part**), then the rule *matches* and **the replaced part** is replaced by **the replacement**. All of the basic components are facultative and any of them can be left blank (not all sound changes depend on the surroundings, **the replaced part** is blank in the case of intrusive sounds and **the replacement** is blank in the case of sound deletion).

The following table gives an example of a simple sound change adapted

for the program’s use. For comparison, the original description by Millward is formulated as follows: “By the end of ME a final /b/ after /m/ was being lost in pronunciation” [Millward(1988), page 78]. The sound **b** is being replaced by an empty string (thus deleted), but only if it is directly preceded by **m** and at the same time is at the end of the word (which technically means that a special character **\$** that denotes the end of the word directly succeeds our **b** in question).

Table 1: An Example of a Rule

prefix	rule	suffix	repl	Description
m	b	\$		/b/ was lost after /m/ in the word-final position by the end of ME [Millward(1988), 128]

The rules are written in Kirshenbaum’s phonetic notation [Kirshenbaum(1992), page 4-7] with the addition of several kinds of wildcards and other special characters the form of which mostly stems from the syntax of regular expressions, since these are then also used for internal processing of the rules. The accepted wildcards (and metacharacters) are expressed in the following list (please note that this necessarily mostly consists of the description of regular expressions [Group(1997)]):

1. **.** The **dot** matches any character (but exactly one).
2. **^**, **\$** The **caret** sign matches the beginning of a word and the **dollar** sign matches its end.
3. **[abcdee:]** The **square brackets** enclose a set of characters and they match exactly one character from the set (**hi[mt]** thus expresses **him** and **hit**, but not **hi** or **himt**). The rules in my program allow for entering two-character sounds in the case of long vowels,

such as `e:`, because these are then internally translated to another notation where even the long vowels are represented by a single character (if we used for example `[e:i]` in a regular expression directly without the translation to the internal notation, the brackets would match either `e`, `:` or `i`).

4. `{vcd & frc}` The **curly brackets** also match exactly one character from a set, but the set is not given by enumeration, but by the phonetic properties of the sound. For example, `{vcd & frc}` (which represents a sound that is voiced and a fricative at the same time) thus matches exactly one sound from the following set: `v`, `z`, `ð`, `ʒ`, `ʧ`. This kind of a wildcard is not a standard functionality present in regular expressions, but it is my custom extension. The abbreviations of the phonetic properties and the description of the sound inventory by these abbreviations are taken from Kirshenbaum [Kirshenbaum(1992), page 3]. Inside this wildcard, the phonetic properties can be combined by a logical AND (the `&` symbol), a logical OR (`|`), and negation (the `!` symbol; `{!vw1}` thus matches all consonants: all sounds not being vowels). The rules can be grouped by being enclosed in brackets, in order to apply the logical operators to whole combinations of conditions (`{vw1 & ! (lng | rnd)}` matches a vowel that is neither long nor rounded).
5. `(ab)\1` Text enclosed in **round brackets** can be backreferenced by a **backslash** followed by the sequential number of the brackets. Bracket pairs are numbered from 1 and brackets can be also placed inside other brackets (the order of the opening/left brackets is then decisive). The backreference matches *exactly* what was matched in-

side the corresponding pair of brackets, thus `([ab])c\1` matches `aca` and `bcb`, but not `acb` or `bca`.

The backreference can be used in any of the four basic components of a rule (when used in the `replacement` component, the substring that matched with the brackets in the `prefix`, `rule` or `suffix`, is then pasted into the replacement). This is a standard regex behaviour, though with one exception: the `prefix` and the `suffix` of the rule are each enclosed in their own pairs of round brackets, and these brackets are also taken into account in the numbering of the brackets. The `prefix` is therefore backreferenced by `\1`, the first user-supplied pair of brackets (if within the `prefix` or `rule`) will be backreferenced by `\2`, etc.

6. `(mag|cunn|scul)an` The **pipe symbol** (`|`) works as a logical OR, and separates the expression into two (or more) parts: if any of them matches the input string, the whole regular expression matches the input string. The scope of this operator can be limited by round brackets, so that for example `wes(an|e|st)` matches `wesan`, `wese`, `wes` (that's for the empty string between the two successive pipe symbols) and `wesst`.
7. `?` The **question mark** makes the preceding element (either a character or the whole content of round brackets) facultative, thus `naman?` expresses both `nama` and `naman`.

3.2 Pronunciation of Old English

As Old English has largely phonetic spelling, which is moreover to a great extent regular and standardized, the task was much easier to create the

generator of Old English pronunciation than it was in the case of Middle English. The rules mostly consist of a straightforward translation of the rules traditionally described by Millward [Millward(1988), page 69]. However, there are several possible complications that have to be taken care of.

The rules are applied to the word in question one after another in a fixed order. Most of the rules are mutually independent, so the proposed order is just one amongst many other (also equally functional) orders. However, there are several rules that might interfere with one another and therefore the order of these “problematic” rules must be established with a little bit of caution (for example, if the rule that translates <sc> to /ʃ/ came *after* the one that translates <ce> to /tʃe/, then the word <scēap> would be wrongly pronounced as /stʃeap/ instead of /ʃeap/).

Another problem is the treatment of prefixes. For example, in the word <gesēon>, <ge> is the prefix and therefore the letter <s> should remain voiceless (it has the voiced and voiceless allophonic variants /s/ and /z/ that are distributed according to whether the letter <s> is surrounded by voiced sounds [Millward(1988), page 71]). The rule that decides whether the letter <s> should be voiced or not, only checks for voiced characters before and after the <s> in question and would decide to make it voiced. This has to be fixed, therefore another rule has been devised that puts an “intrusive dash character” between the prefix and the root: this dash prevents the rule that checks for voiced characters from being applied (and the very last rule strips these helper dashes from the words). It is also possible to enter a word already containing a dash, which often connects words (creating kennings, for example) in primary texts, or which

may be present in a dictionary.

This dash-inserting rule only covers (in the sample rules that are a part of the thesis) several most frequent prefixes to demonstrate the principle, even though there are much more prefixes in Old English. Covering all the prefixes and deciding whether the word-initial <ge> is a prefix or a part of the root (in words such as <geseōn> vs. <gelīce>) would be possible to achieve by using a dictionary. For example, Bosworth and Toller's *An Anglo-Saxon Dictionary* that is available in an electronic form at our faculty's webpages at <http://bosworth.ff.cuni.cz/> lists entries already with a dash indicating the division between the prefix and the root. Every word, before the OE pronunciation rules are applied on it, could be thus checked against this dictionary for the prefix boundary, which would lead to more precise results. My program could (and most probably will) reciprocally contribute to this dictionary by automatically generating pronunciation in IPA for each of the thousands of the entries. There have been slight simplifications, concerning an artificial merger of similar phonemes. I decided not to distinguish between very similar phones, such as /u/ and /ʊ/ (and merge them both as /u/), because the chief purpose of this grapheme-to-phoneme translator was to prepare grounds for the sound change analyzer, and towards Middle English there is much more fuzziness and sources of inaccuracy that merging /u/ and /ʊ/ or /i/ and /ɪ/ does not inflict much harm. Also, rules (for the diachronic development) would have to be defined that would describe the transition between /u/ and /ʊ/ to make the sound change analyzer work, because for a computer, any inequality is inequality and the explanatory sequence of sound changes would not be thus found (or the

method would have to be more complicated in another way to provide some kind of fuzziness). However, if it were necessary for some other purpose, it would not be a problem to alter the phones and rules in the program, so that the distinction could be made.

Some complications may occasionally arise from the fact that the system operates only on phones and does not take into account suprasegmental aspects of speech, such as stress, intonation or syllable boundaries; but even though it uses only phones for the calculations, the results are good. Also, the program is not capable of automatically assigning the vowel length marks to the vowels, and the input text must thus already contain them.

The following table shows the rules I adapted from Millward [Millward(1988), page 69] and use for grapheme-to-phoneme translation for Old English.

The dialect covered by this description is late West Saxon [Millward(1988), page 69].

Table 2: Old English Pronunciation in Rules

prefix	rule	suffix	repl	Description
(ge wiT for)		...	-	<ge->, <wið-> and <for-> prefix treatment [Bright(1972), 17]
{!vls}	f	{!vls}	v	<f> read as /v/ between voiced sounds [Bright(1972), 17]
{!vls}	s	{!vls}	z	<s> as /z/ between voiced sounds [Bright(1972), 17]
{!vls}	T	{!vls}	D	<thorn/eth> voiced between voiced sounds [Bright(1972), 17]
	ng		Nŋ	<ng> as /ŋ/ [Bright(1972), 18]
	nk		Nk	<nk> as /ŋk/ [Bright(1972), 18]
	cg		dʒ	<cg> as /dʒ/ [Bright(1972), 18]
ˆ	g	{fnt & vwl & !low}	j	<g> as /j/ before long or short /i/, /e/ [Bright(1972), 18]
{fnt & vwl}	g	{fnt & vwl}	j	<g> as /j/ between front vowels [Bright(1972), 18]
{fnt & vwl}	g	\$	j	<g> as /j/ word-finally after a front vowel [Bright(1972), 18]
	g	{!vwl}	j	<g> as /j/ syllable-finally (?) [Bright(1972), 18]

	g		Q	default <g>pronunciation as /ɣ/ [Bright(1972), 18]
	sc		S	<sc>as /ʃ/ [Bright(1972), 18]
	cc		k	/cc/ should be read /k/ [Bright(1972), 18]
	c	{fnt & vwl& !low & !rnd}	tS	<c>as /tʃ/ before <i>,<e>[Bright(1972), 18]
i:	c	{vwl & fnt}	tS	<c>as /tʃ/ between /i:/ and a front vowel [Bright(1972), 18]
{fnt & vwl& !low & !rnd}	c	\$	tS	<c>as /tʃ/ word-finally (after a front vowel) [Bright(1972), 18]
.	c	{!vwl & !(apr&lbv)}	tS	<c>as /tʃ/ syllable-finally [Bright(1972), 18]
	c		k	default <c>pronunciation as /k/ [Bright(1972), 18]
.	h		x	<h>as /x/ non-initially [Bright(1972), 18]
{(hgh smh) & unr}	e		@	<e>read as /ə/ after /i/ and /i:/ in diphthongs [Bright(1972), 19]
	ea		&@	<ea>is read /æə/ [Bright(1972), 19]
	e:a		&:@	<e:a>is read /æ:ə/ [Bright(1972), 19]
{!vwl}	\1			double consonants as one phonetically
	-			stripping the helper dashes

3.3 Pronunciation of Middle English

With Middle English, the situation is much more complex. Because of its ambiguous relationship between orthography and pronunciation [Lohófer(2007)] (and also the aforementioned generally poor level of the standardization of Middle English spelling), it is not possible to make a reliable standalone grapheme to phoneme convertor that would operate with the proposed rule system and that would provide one correct form of pronunciation as the output.

However, certain advantages arise from connecting this with the Old English orthography to pronunciation translator and the algorithm for suggesting a feasible sequence of sound changes to explain the develop-

ment of a word. The search algorithm is attempting to find a sequence of word alterations that lead from the Old English word (the pronunciation of which we can generate quite reliably) to its Middle English counterpart, thus searching for a minimal path in a graph that leads from an initial node to a final one.

It is however easy to adapt the algorithm to make it consider more than one node a final one. It would therefore start in the node of the initial Old English form and if a path was found that would lead to any of these destination nodes (representing more possible Middle English pronunciations), the algorithm would consider it as a possible answer (moreover, it would also suggest the ME spoken form). These possible answers are however not equal in their feasibility as suggested pronunciations of the given written form: each of them will have a particular numeric **weight**, the higher which is, the less feasible the particular pronunciation is. The syntax of the rules for grapheme to phoneme conversion for Middle English is the same as for Old English, but more complicated operations are used, so that not one proposed ME pronunciation is suggested, but the result of the convertor is a string with a special format that I have devised that has the following features:

1. Describes all possible pronunciations for the given written form of a word.
2. The **weight** of each of the possible pronunciations can be simply extracted from the string.
3. The string at the same time has the form of a regular expression that matches all the possible pronunciations (and none other).
4. The computational complexity of creating this string for the given

spoken form is linearly proportional to both the number of the rules and to the length of the input string.

The format of the string will be probably best explained using an example: let us consider the Middle English word <thombe>. Its non-deterministic reading will have the following form (the phones are encoded in Kirshenbaum's IPA notation):

$((D(100)?|T(1)?)(o(1)?|o:(1)?|u(1)?)mb(@(1)?|(1)?))$

This says that there are two possibilities as to what the initial fricative would be: either voiced or voiceless. The probability is strongly in the favour of the voiceless one (the weight being 100 vs. 1), but there is still room left for the voiced one (since there are some words beginning with <th> where the initial fricative is voiced and these words must be taken into account as well). There are three possible sounds to which the letter <o> normally translates, varying both in quantity and quality. Since it is not easily decidable which of them will be used in this position, the choice is left up to the Dijkstra's algorithm and therefore to a hypothetical diachronic analysis. The consonant cluster <mb> will be read literally. The final <-e> can be transcribed either as schwa or left out completely, both variants are deemed equally correct (by the algorithm). Once a rule puts a "reverse wildcard" (for example $(D(100)?|T(1)?)$) into the string, it could possibly happen that the following rules would rewrite the characters within these "reverse wildcards" (D or T in this case), which must be avoided. To avoid this scenario, all of the characters in the input word are first (i.e. by the very first rule) made to be followed by an underscore (_). This underscore marks characters that are still allowed to be replaced by the rules: once they are replaced, the

underscore pertaining to the affected characters is deleted and they are thus “locked” and prevented from being changed again (phones within the “reverse wildcards” are always locked by default). This was not necessary in the case of Old English, because the situation there was much less complicated, but here it is indispensable.

It might be possible to adjust the values in a more satisfactory way if more detailed secondary sources are used and more primary material is available for tuning the algorithm (i.e. a large parallel corpus of written text with its pronunciation and sample explanations of phonetic development of enough words).

The proposed rules for calculating the non-deterministic description of Middle English pronunciation were mostly adapted from Astrid Lohófer’s explanations [Lohófer(2007)] and have the following form:

Table 3: Middle English Pronunciation in Rules

prefix	rule	suffix	repl	Description
.			_	putting an underscore after each letter
	d_g_		dʒ	late ME <dg>as /dʒ/ [Millward(1988), 138]
	c_h_		tʃ	<ch>read as /tʃ/ [Millward(1988), 138]
	s_h_		ʃ	<sh>is read as /ʃ/ [Millward(1988), 138]
^	s_		s	<s>is never voiced initially [Lohófer(2007), 3]
{vwl}_?	s_	{vwl}	(z(1)? s(10000)?)	<s>voiced mostly between vowels [Lohófer(2007), 3]
	s_		(s(1)? z(1000)?)	<s>otherwise probably not voiced [Lohófer(2007), 3]
	c_k_		k	<ck>should be read as /k/ [Lohófer(2007), 3] [Strang(1990), 228]
	c_	{vwl & fnt}	s	<c>as /s/ before front vowels [Lohófer(2007), 3]
	c_	{vwl & bck}	k	<c>as /k/ before back vowels [Lohófer(2007), 3]
	c_		(sj(800)? k(1)? s(1)?)	<c>as either /sj/, /k/ or /s/ [Lohófer(2007), 3]
	n_	[kg]_	ŋ	<n>is velar before velar consonants [Lohófer(2007), 3]

	q_u_		kw	<qu>read as /kw/ [Millward(1988), 135]
	x_		ks	<x>is read as /ks/ [Lohöfer(2007), 3]
{vwl vcd}	t_h_	{vwl !vls}	(D(1)? T(10000)?)	<th>between voiced sounds [Lohöfer(2007), 3]
._	t_h_	._	(D(1)? T(1)?)	<th>otherwise inside a word [Lohöfer(2007), 3]
	t_h_		(D(100)? T(1)?)	otherwise <th>is probably not voiced (initially/finally) [Lohöfer(2007), 3]
	g_h_		x	<gh>as /x/ (we ignore the /x/ vs. /ç/ allophones) [Millward(1988), 138] [Lohöfer(2007), 3]
	w_h_		hw	<wh>still read as /hw/ [Lohöfer(2007), 3]
^	h_	{vwl}	h	<h>as /h/ word-initially before vowels [Lohöfer(2007), 3]
	h_		(h(10)? x(1)?)	<h>is otherwise /h/ or /x/ [Lohöfer(2007), 3]
	g_u_		g	<gu>for /g/ (mostly in loanwords though) [Millward(1988), 139] [Strang(1990), 229]
	g_(g_)?		(g(1)? dʒ(1)?)	<g>and <gg>can be either /g/ or /dʒ/ [Lohöfer(2007), 3] [Strang(1990), 232]
	i_e_		(e:(1)? i:(1)?)	<ie>as either /e:/ or /i:/ [Lohöfer(2007), 3]
	e_[ae]_		e:	<ee>and <ea>as /e:/ [Lohöfer(2007), 3]
	e_[iy]_		(ei(1)? ej(10)?)	<ei>and <ey>as /ei/ [Lohöfer(2007), 3]
	a_[iy]_		ai	<ai>and <ay>as /ai/ [Lohöfer(2007), 3]
	a_[uw]_		au	<au>and <aw>as /au/ [Lohöfer(2007), 3]
	o_o_		(u(1)? o:(1)?)	<oo>as either /u/ or /o:/ [Lohöfer(2007), 3]
	o_a_		o:	<oa>as /o:/ [Lohöfer(2007), 3]
	o_[uw]_		(u:(1)? ou(1)?)	<ou>and <ow>as /u:/ or /ou/ [Millward(1988), 138] [Lohöfer(2007), 3]
	[iy]_		(i(1)? i:(1)? j(4)?)	<i>and <y>as /i/, /i:/ or /j/ [Lohöfer(2007), 3] [Strang(1990), 230]
	e_	\$	(@ (1)? (1)?)	reduced final /e/ [Lohöfer(2007), 3]
	e_		(e(1)? e:(3)? @ (5)?)	<e>as /e/ or /e:/ [Lohöfer(2007), 3]
	a_		(a(1)? a:(1)?)	<a>as /a/ or /a:/ [Lohöfer(2007), 3]
	o_		(o(1)? o:(1)? u(1)?)	<o>as /o/, /o:/ or /u/ [Lohöfer(2007), 3]
	u_		u	<u>as /u/ [Lohöfer(2007), 3]
[bpdtkflmrn]_	\1			making double consonants in orthography single
	-			cleaning all remaining underscores

3.4 Phonological Changes between OE and ME

As has been said before, to describe the phonological changes in the development of the language, we must trace it within the framework of its spoken form. The proceedings of the program, the objective of which is to find a feasible explanation of a change of an Old English word and present the answer in the form of a series of individual sound changes, can be summarized in the following way:

1. The spoken form of the Old English word is determined from its written form (that was given to the program as the input).
2. All possible Middle English spoken forms (with their various “weights”) are computed from the ME written form (from the input).
3. An implicit graph is given as the input to (a modification of) the Dijkstra’s graph searching algorithm. The set of its nodes is formed by (the infinite number of) all possible pronunciations of any word (all combinations of phones). An (orientated) edge between nodes A and B exists exactly when there is a sound change (within our set of rules) that transforms word A to word B. The initial node given to the Dijkstra’s algorithm is the suggested Old English pronunciation. The final nodes are formed by all possible pronunciations of the Middle English word (in fact, technically there is one virtual final node with no pronunciation assigned and there exist edges from each of the possible ME pronunciations to this virtual final node, their weight being the weight assigned to the pronunciation in question).
4. The graph is searched for such a path that has minimum total weight and leads from the initial node to the “virtual” final one, necessary information is collected throughout the search, and the path (if

found) is presented as a series of word forms and particular sound changes connecting them.

Details about the Dijkstra's algorithm can be found in his 1959 paper, "A Note on Two Problems in Connexion with Graphs" [Dijkstra(1959)]. The weight of the edges is computed from two basic components: the constant weight assigned to particular sound changes (for example, I evaluate deletions and reductions as more likely to happen than insertions) and the estimated probability of the resulting form being a real one in this situation (for example, the length of the form that significantly deviates from the lengths of the input and output forms is penalized; three identical consecutive phones are not allowed at all etc.).

The algorithm takes into account the rough chronology of the individual changes: the approximate date ranges are taken from secondary sources and the search algorithm respects the necessity of their being applied in the chronological order. Also, relative chronology (i.e. that two changes must take place in a given order and not vice-versa, because otherwise it would result in a different word) does not have to be taken care of explicitly, since this problem is solved implicitly. The more obvious solution is that the defined time spans would allow only one order of the two changes to be examined. The second possibility (which takes place when the time spans overlap or are not defined), is that the application of the rules will be tried in both orders: the non-feasible will simply lead to a dead-end and the feasible will remain.

The rules describing the phonetic changes between Old and Middle English were taken from Millward's *Biography of the English Language* [Millward(1988), page 127-133] and formalized for the use by the pro-

gram. They are presented in the following table:

Table 4: Old English to Middle English sound changes

prefix	rule	suffix	repl	Description, Approx. Dating
	[td]	{!vwl}{!vwl}		1150-1400: /t/ and /d/ were often lost in heavy clusters [Strang(1990), 250]
	w	l		1350-1600: /wl/ becomes simple /l/ [Strang(1990), 166]
	w	r		1350-1600: /wr/ is reduced to /r/ [Strang(1990), 166]
ˆ	f	n	s	1400-1500: initial /fn/ was absorbed into commoner /sn/ [Strang(1990), 166]
e:	o		@	950-1020: second element reduction in /e:o/ (by 970) [Strang(1990), 287]
[e&]	a		@	950-1020: second element reduction in /e:a/ or /æ:a/ (by 970) [Strang(1990), 287]
	@	\$		1050-1500: unstressed final <e>, read /ə/, was dropped during ME [Millward(1988), 133]
	l	tS		/l/ was lost in the vicinity of tS (before) [Millward(1988), 128]
tS	l			/l/ was lost in the vicinity of /tʃ/ (after) [Millward(1988), 128]
	i		e:	1200-1300: /i/ sporadically became /e:/ later in the 13th century [Millward(1988), 132]
	u		o:	1200-1300: /u/ sporadically became /o:/ later in the 13th century [Millward(1988), 132]
	a	(?!vwl}{+{vwl}.*)?&	a:	1200-1300: /a/ lengthened in open syllables in the 13th century (FIXME: open syllables) [Millward(1988), 132]
	e	(?!vwl}{+{vwl}.*)?&	e:	1200-1300: /e/ lengthened in open syllables in the 13th century (FIXME: open syllables) [Millward(1988), 132]
	o	(?!vwl}{+{vwl}.*)?&	o:	1200-1300: /o/ lengthened in open syllables in the 13th century (FIXME: open syllables) [Millward(1988), 132]
	a:	{!vwl}{?!vwl}.*)?&	a	900-1000: /a:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	&:	{!vwl}{?!vwl}.*)?&	&	900-1000: /æ:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	e:	{!vwl}{?!vwl}.*)?&	e	900-1000: /e:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	i:	{!vwl}{?!vwl}.*)?&	i	900-1000: /i:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	o:	{!vwl}{?!vwl}.*)?&	o	900-1000: /o:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]

	u:	{!vwl}{(!vwl}.*)?&	u	900-1000: /u:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	y:	{!vwl}{(!vwl}.*)?&	y	900-1000: /y:/ shortened in closed stressed syllables during the 10th century (FIXME: closed stressed) [Millward(1988), 132-133]
	i	mb	i:	950-1100: /i/ lengthened before /mb/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	o	mb	o:	950-1100: /o/ lengthened before /mb/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	i	[nl]d	i:	950-1100: /i/ lengthened before /nd/ and /ld/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	u	[nl]d	u:	950-1100: /u/ lengthened before /nd/ and /ld/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	a	ld	a:	950-1100: /a/ lengthened before /ld/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	e	ld	e:	950-1100: /e/ lengthened before /ld/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	o	ld	o:	950-1100: /o/ lengthened before /ld/ (as early as OE and survived the 14th c. shortening) [Millward(1988), 132]
	i:w		iu	/iu/ developed from /i:w/ [Millward(1988), 130]
	e:@w		iu	/iu/ developed from /e:@w/ [Millward(1988), 130]
	&:@?w		eu	/eu/ developed from /@:w/ and /@:@w/ [Millward(1988), 130]
a	[wQx]		u	/au/ developed from /aw/, /ay/ and /ax/ [Millward(1988), 130]
	[a:o:o] [wQx]		ou	/ou/ developed from /a:w/, /a:y/, /a:x/ and also from /o:w/, /o:y/ and /o:x/ [Millward(1988), 130]
	[&e]j		&i	/@:i/ developed from /@:j/ and /e:j/ [Millward(1988), 130]
&	@x		i	/@:i/ developed from /@:x/ [Millward(1988), 130]
	y		i	950-1500: /y/ unrounded to /i/ (dating dependent on dialect) [Millward(1988), 129]
	y:		i:	950-1500: /y:/ unrounded to /i:/ (dating dependent on dialect) [Millward(1988), 129]
	&		a	950-1500: /@:/ lowered to /a/ in all dialects by the end of ME (Millward) 970-1170 (Strang) [Millward(1988), 129] [Strang(1990), 292]
	a:		o:	950-1200: /a:/ was rounded to /o:/ in the beginning of ME [Millward(1988), 129] [Strang(1990), 292]
..	[aeou]		@	900-1500: short vowels were reduced to schwa in unstressed syllables (beginning in OE and continuing in ME) (FIXME: unstr.syl.) [Millward(1988), 130]

{!vwl}	@			after a consonant, the difference between a schwa and nothing is not very significant
	e@		e	1050-1500: the /eə/ diphthong smoothed to /e/ in Middle English [Millward(1988), 130]
	&@		a	1050-1500: the /æə/ diphthong smoothed to /a/ in Middle English [Millward(1988), 130]
	[e:&:]@		e:	1050-1500: the diphthongs /æ:ə/ and /e:ə/ smoothed to /e:/ in Middle English [Millward(1988), 130]
{!vwl}		{!vwl}	@	a parasitic schwa between two consonants [Millward(1988), 130]
^	j[eiI]		i	the <ge->prefix was reduced to /i/ [Millward(1988), 127]
[lr]	Q		w	1150-1400: /y/ became the semivowel /w/ after /r/ or /l/ [Millward(1988), 127]
{vwl}	{!vwl}	\$		1050-1500: unstressed final consonant was lost in Middle English (FIXME:unstressed) [Millward(1988), 127]
[st]	w			/w/ dropped after /s/ or /t/ [Millward(1988), 127]
	v	{vwl}?{!vwl}		1150-1400: /v/ tended to drop out before a consonant or vowel + consonant [Millward(1988), 128] [Strang(1990), 250]
m	b	\$		1300-1500: /b/ was lost after /m/ in the word-final position by the end of ME [Millward(1988), 128]
m		{!vwl & !(blb & stp)}	b	1350-1600: intrusive /b/ after /m/ [Millward(1988), 128] [Strang(1990), 166]
m		\$	b	1350-1600: final intrusive /b/ after /m/ [Millward(1988), 128] [Strang(1990), 166]
n		{!vwl & !(alv & stp)}	d	1350-1600: intrusive /d/ between /n/ and a consonant [Millward(1988), 128] [Strang(1990), 166]
n		\$	d	1350-1600: final intrusive /d/ after /n/ [Millward(1988), 128] [Strang(1990), 166]
s		{!vwl & !(stp & alv)}	t	1350-1600: intrusive /t/ after /s/ [Millward(1988), 128] [Strang(1990), 166]
s		\$	t	1350-1600: final intrusive /t/ after /s/ [Millward(1988), 128] [Strang(1990), 166]
{vwl}j[aeoi]				
	h			1150-1400: /h/ was often lost in unstressed positions (FIXME: unstressed) [Millward(1988), 128] [Strang(1990), 250]
^	h	[lnr]		1150-1400: /h/ was lost in initial position before /l/, /n/ and /r/ [Millward(1988), 127]
	Q		w	1100-1250: /y/ became /w/ by 1200 in most places [Millward(1988), 127]
	&:		e:	800-1100: /æ:/ became /e:/ (two possible sources) [Millward(1988), 129]
	&:		e	1050-1500: /æ:/ could also become /e/ during ME [Strang(1990), 292]

	&:		ea	1050-1500: /æ:/ could also became /ea/ during ME [Strang(1990), 292]
	ea		&	900-1000: /ea/ smooths to /æ/ (before /æ/ changes to /a/ ->10th cent?) [Strang(1990), 292]

4 Technical Information

4.1 System Requirements to Run the Program

In its current state, the program requires the following additional software:

1. PHP (developed on PHP 5.1) with the support for multibyte string functions (which should be available implicitly in PHP 5.3 and later).
2. Running MySQL database server.
3. The jQuery JavaScript library (only for the website interface), `http://jquery.com/`.
4. Richard Ramsden's PriorityQueue PHP class, available at `http://www.ramsden.ca/` (but also included with the source code of the program).

Assuming a PHP interpreter and database server are already working, the following extra steps must be taken in order to install the program:

1. The database dump (`oelab.sql`) must be loaded into a database on the MySQL server (the UTF-8 encoding is expected).
2. In the file `oelab.php`, in the class `OldEnglishLab`, in the function `mysql_connect_and_db()`, the content of the variables containing MySQL server address, username, password and database name

must be altered in order to fit the configuration with which the database dump was used.

3. The processing of multibyte strings should be configured to use the UTF-8 encoding, which can be achieved by the following commands being inserted in the PHP script before the class `OldEnglishLab` is used:

```
mb_internal_encoding('UTF-8');  
  
mb_regex_encoding('UTF-8');
```

4.2 Program interface

The core of the program is written as a PHP class, called `OldEnglishLab` in the file `oelab.php`. The following functions may be of interest:

```
function expand_phonetic_wildcard($rule, $fmt) Expands a phonetic wildcard given in the variable $rule, and a list of phones matching the rule is returned. The variable $fmt allows the programmer to select between two output formats: the result is in IPA (Unicode), separated by comma and a space ($fmt==1) or the result may be requested in the internal encoding and no separator is then used (when $fmt==2).
```

```
function update_internal_phs() This function reassigns internal encoding symbols to all the phones and should be always called after any alteration to the phone inventory is made (i.e. new phones added etc.).
```

```
function update_internal_rule($id) Updates the internal representation of a rule with the internal ID $id. Should be called after any
```

change to the rule in question is made (or when it is added).

`function update_internal_rules()` Updates the internal representation of all the rules at once. Should be called when there is any change to the phone inventory or after the function `update_internal_phs()` is called.

`function str_kph2int($s)` Converts a string from the Kirshenbaum's encoding to the internal one.

`function str_kph2ipa($s)` Converts a string from Kirshenbaum to IPA (in Unicode).

`function str_int2ipa($s)` Converts a string from the internal phonetic encoding to IPA (Unicode).

`function str_int2kph($s)` Converts a string from the internal phonetic encoding to Kirshenbaum.

`function str_uni2ascii($s)` Converts a string from Unicode (Old English text, for instance) to ASCII.

`function word_read($s, $realm)` Receives a word (passed by the variable `$s`) in ASCII and returns its pronunciation in Kirshenbaum. The `$realm` parameter chooses the set of rules that are applied: for transcribing Old English, enter 2, for non-deterministic Middle English, enter 4.

`function word_read_uni($s, $realm)` This is just a wrapper around `word_read($s, $realm)`, it only converts the input string from Unicode before calling it.

`function read_text_uni($s, $realm)` Returns a transcription of a longer text: it divides it into words, each of which is then processed by the `word_read($s, $realm)` function.

`function find_path($in_str, $max_iters, $goals, &$log)` Returns an array containing the sequence of forms describing the development of the input word (`$in_str`) into any of the forms contained in the array `$goals`, using at maximum `$max_iters` iterations of the Dijkstra's graph search algorithm. Information documenting the process of the search is logged in the variable `&$log`.

4.3 A Website Presenting the Results of the Experiment

First of all, the website dedicated to presenting the results of the experiment has been created mainly in order to make the results easily accessible and to facilitate their presentation. It is accessible online at <http://oe.danmarek.cz/> and before the school presentation of the thesis (13th September 2010), the access is secured by a password (which is only given to those responsible for evaluating the thesis). After the school presentation takes place, the website will be made available to the public (i.e. the password will be removed) and at the same time, functionality that allows altering the contents of the internal database will be disabled, i.e. random visitors will not be allowed to change the configuration of phonemes and the rules etc. (for working with these, it will be necessary to download and install one's own copy of the system; the installation files will be publicly available), but it will be possible to see the rules and phonemes. Also, the grapheme-to-phoneme convertors, historical change analyzer and phonetic wildcard evaluator will be accessible and functional.

Please note that the website contains special IPA characters, which must

be supported by the client's OS/browser (it should be supported by most modern systems; in the case of special characters not showing, the installation of proper fonts should remedy it). The website is composed of the following subpages:

1. **Characters** Shows the list of characters (concerning the orthography) in both Unicode and ASCII.
2. **Phones** Lists the phones with their Unicode IPA value, Kirshenbaum's ASCII encoding, internal encoding and phonetic properties.
3. **Playground** Invites the user to test the grapheme-to-phoneme converters on individual words, as well as the phonetic wildcard evaluator.
4. **Playground 2** Allows the user to automatically translate longer texts to their pronunciation.
5. **Test search** Will try to find an explanation of a historical development between two versions of a word. Sample words are listed in the right column.

5 Possible Ways of Utilization

I hope to be able to make the program available at my webserver at <http://oe.danmarek.cz/> for as long as possible: both as a running application and as an archive of source code and database dumps for whatever use may anyone see fit. The license of the source code is GPLv2 (GNU General Public License), and as an open-source project it can be thus freely reused for (almost) any purpose. Apart from using the program directly at the website, there are several possibilities of how it can be useful.

5.1 Reusability & Compatibility

As everything necessary for running the program is freely available: its source code, database data, platform (PHP, MySQL etc.); it is possible to easily run the program elsewhere. The program is written in a structured fashion (individual functions within a PHP class), and therefore the program (or its parts) can be simply connected to another piece of software that would utilize its functions. For example, our department runs an electronic version of *Bosworth and Toller's Dictionary* at <http://bosworth.ff.cuni.cz/> where the program will be utilized to automatically generate pronunciations for the entries.

The Old English orthography to pronunciation convertor can be used to process texts of virtually any length. Also, having such a grapheme-to-phoneme convertor is one of the components that are being used in speech synthesizers, therefore this work is partially preparing grounds for the creation of one (although the system should be extended, for example by a support for suprasegmental features).

Also, huge databases of corpora texts can be automatically assigned their pronunciation, allowing thus phonetics-based search to take place in historical corpora (possibly also taking advantage of the implemented system of phonetic wildcards).

5.2 Adaptability: A Different Language

The design of how the set of rules is constructed and how it processes data, is general enough to allow anybody to create a different set of rules in order, for example, to create an orthography-to-pronunciation convertor for another historical stage of English or even for a completely

different language. The experiment with Old and Middle English showed that the more standardized and regularly phonetic spelling a language has, the better the results. Therefore, it should be relatively easy to adapt the program for example to convert written Spanish to its pronunciation in IPA.

In the case of adapting the system to the processing of another language, it should be noted that it is perfectly possible to easily extend the system by the addition of other phonemes, their deletion or alteration of their phonetic properties.

5.3 Didactic Purposes

The possibility of using or adapting the pronunciation generator for didactic purposes is obvious (students can check their own transcription against the one suggested by the program, connecting orthography with one of several transcriptions, determining the orthography from a given transcription etc.). The system processing phonetic wildcards could be adapted in a similar manner (although its orientation would be more generally linguistic), and the piece of software that suggests a possible phonetic development of a word can also provide a student with an explanation of development (but should rather be used to trigger a discussion than as a reference, since its results are still experimental). Each change within this explanation is also accompanied by a reference(s) to relevant passages in secondary literature, so it can also be useful as a starting-point for a person who wants to quickly access relevant explanations of the phenomena by Millward or Strang.

Also, when trying to construct the set of rules to properly describe a lan-

guage change or pronunciation, it is necessary for the rules to be as exact as possible: the system does not have any implicit support for recognizing “similar” phonemes, it works only with the principle of (in)equality. Because of this narrow margin for errors, trying to construct a set of rules for either purpose thus may be used didactically to allow a student to reveal places where their understanding of the linguistic principles is either wrong or vague.

5.4 Automated Analyses

Providing an automated explanation of a diachronic development of a word also means that the system gives the user its rough estimate of the probability of the connection between the two forms: a rough estimate of whether one word can be an etymon of another. If an explanation of the development is not found, the program’s estimate is obviously that there is no connection. However, when it is found, the program’s verdict is not only that it deems the connection possible, but also the extent to which the connection is probable on a continuous scale (the extent being expressed by the overall weight of the path found - the system may give false positives and provide an explanation between forms that are in fact not connected, but this should be reflected to some degree in the overall weight). This function (of estimating the probability of a connection between two diachronic forms) might be possibly used in computer analyses for example to try to automatically extract pairs of words that represent the same word in two different time periods (when processing semi-parallel texts such as Old English and Middle English translations of the Bible).

6 Resumé

Tato bakalářská práce se zabývá pokusem o formalizaci fonetických změn mezi starou a střední angličtinou a navržením a implementací počítačového programu, který algoritmicky řeší následující funkce:

1. Automatické převedení staroanglického textu z psané formy do formy mluvené, reprezentované pomocí mezinárodní fonetické abecedy (IPA). Tohoto cíle se podařilo (pouze s drobnými nedostatky) dosáhnout, také díky tomu, že staroangličtina je (na rozdíl od střední angličtiny či dnešní angličtiny) dosti pravidelná a jednoznačná, co se týče vztahu pravopisu a výslovnosti.
2. Počítačové zpracování středoanglického textu na podobné bázi. Nicméně kvůli často mnohoznačnému vztahu mezi pravopisem a výslovností, výstupem tohoto postupu není jednoznačný přepis výslovnosti, ale vzorec, který v sobě nese popis všech potencionálních výslovností daného slova včetně odhadu relativní pravděpodobnosti toho, která z navrhovaných výslovností skutečně odpovídala realitě.
3. Algoritmus, který se (mimo jiné za pomoci dvou výše uvedených funkcí) na základě dvou vstupních slov (přesněji psané formy téhož slova ve dvou fázích vývoje: staroanglické a středoanglické) pokusí navrhnout posloupnost známých fonetických změn, které by vysvětlovaly historický vývoj daného slova. Algoritmus je založen na Dijkstrově algoritmu hledání nejkratší cesty v grafu a pracuje s daty jako je databáze známých fonetických změn ve zkoumaném období, jejich přibližné datace, relativní ohodnocení pravděpodobnosti toho, že nastanou konkrétní fonetické změny, ohodnocení přijatelnosti průběžných forem slova, existence více možných středoanglických variant slova

apod.

Data, se kterými vytvořený software pracuje (popisující fonetické změny mezi starou a střední angličtinou, výslovnost staroangličtiny a střední angličtiny) jsou brána ze sekundárních zdrojů a soustřeďují se na popis pozdní západosaštiny v případě staroangličtiny [Millward(1988), strana 69] a londýnský dialekt v případě středoangličtiny [Millward(1988), strana 125].

Všechny tři základní výše uvedené procesy staví na databázi změn (interpretace psané podoby do podoby mluvené je také zpracovávána jako změna). Tyto změny reprezentují (protože to úzce souvisí s jejich vnitřním zpracováním) ve formě vycházející ze syntaxe regulárních výrazů, s určitými modifikacemi a přidanými funkcemi. Každé pravidlo je definováno pomocí popisu fonetické podoby místa změny, jeho fonetického okolí (fonémy tomuto místu předcházející i jej následující) a případným řetězcem fonémů, které místo změny nahradí. Krom standardních metaznaků, které regulární výrazy nabízejí, systém umí pracovat s fonetickými "žolíky" (wildcards), které zastupují libovolný znak z množiny znaků, která je vymezená jejich fonetickými vlastnostmi (např. neznělá frikatíva).

U postupu převedení psané formy textu do odpovídajícího mluveného přepisu v IPA, které využívá sadu takovýchto přepisovacích pravidel, je také určeno, v jakém pořadí se pravidla na vstupní slovo či text aplikují. Každé pravidlo se aplikuje právě jednou v určeném pořadí. Pořadí je určeno s ohledem na možnou interferenci některých pravidel za určitých okolností (v případě určení špatného pořadí by mohlo nastat to, že některé pravidlo "zahradí fonetickou stopu", ze které vychází jiné pravidlo, které je aplikováno až následně).

Program nepracuje se suprasegmentálními vlastnostmi textu, jako je např. intonace, přízvuk, rozdělení na slabiky apod. Také nepracuje s morfologií a není tedy schopen sám od sebe odlišit, zda určité posloupnosti znaků na začátku slova patří k předponě či již tvoří kořen. Toto by šlo napravit rozšířením programu o slovník, např. Bosworth and Toller's *An Anglo-Saxon Dictionary*, který u jednotlivých slov hranici mezi předponou a kořenem uvádí. Sémantiku program také nevyhodnocuje a tudíž určité formy, které by při automatických analýzách vyhodnotil jako související, by mohly ve skutečnosti být homonyma.

Pro zpracování středoangličtiny bylo třeba definovat složitější pravidla, pomocí kterých lze vytvořit komplikovanější strukturu k popisu všech potenciálních výslovností a zároveň zabránit případným kolizím. Vnitřní zpracování těchto pravidel je úplně stejné jako v případě výslovnosti staroangličtiny, systém je tedy na tento úkol znovu využit bez vnitřních změn, složitější pravidla jsou pouze určitou nadstavbou postavenou na stejném funkčním základu.

Spojitosť mezi staroanglickou a středoanglickou formou slova, resp. kroky, které vedou od starší formy k novější, jsou určeny za pomoci Dijkstrova algoritmu na hledání nejkratší cesty v grafu. Grafem rozumíme množinu uzlů a hran. Jednotlivé uzly představují jednotlivé (povolené) sekvence fonetických symbolů (tedy libovolné výslovnosti slov, kterých je nekonečně mnoho). Hrany jsou orientované a hrana z uzlu A do uzlu B vede právě tehdy, když existuje v databázi fonetická změna, která transformuje tvar slova A do jiného tvaru B a zároveň forma B je vyhodnocena jako povolená sekvence (může tvořit slovo). Hrany jsou ohodnocené, určité změny jsou považovány za méně pravděpodobné či "krkolonnější". Výchozím

uzlem pro hledání je staroanglická forma slova (její výslovnost) a cílovými uzly jsou všechny navržené možné výslovnosti středoanglické formy slova. Cest v grafu může existovat více, algoritmus najde tu nejlépe ohodnocenou, tzn. tu nejpříjemnější (za předpokladu, že váhy hran a všechna ohodnocení, se kterými se počítá, jsou nastaveny ideálně).

6.1 Využití softwaru či jeho částí

Program je implementován v jazyce PHP, což sice není řešení, které by nabízelo optimální výpočetní výkonnost (tedy dobu trvání výpočtů), nicméně tento jazyk byl zvolen proto, že umožňuje rychlou implementaci (tedy vytvoření programu) i případné změny, což je užitečné pro experimentování. Struktura je navržena tak, aby bylo možné určitě, výpočetně náročné části implementovat externě, např. pomocí výpočetně optimálnějšího kompilovaného jazyka, jako je C++.

Software jako celek (tedy program, který navrhuje vysvětlení souvislosti mezi dvěma psanými formami) lze např. využít jako pomocnou funkci při strojovém zpracování korpusových dat, např. by mohl pomoci při zpracování staroanglického a středoanglického překladu Bible k určení dvojic slov, které si významem v paralelních textech odpovídají (a u takto vybraných dvojic slov např. ještě mapovat četnost výskytů jednotlivých fonetických změn apod.).

Také není problém využít program k automatickému doplnění historického korpusu o jeho fonetický přepis a umožnit v něm tak např. vyhledávání pracující s fonetickými parametry (případně i obohacené o implementovaný systém fonetických “žolíků”/wildcards).

Jednotlivé části programu lze využít k dalším účelům, např. převodník mezi staroanglickou psanou a mluvenou formou bude pravděpodobně použit u online verze Bosworth and Toller's *An Anglo-Saxon Dictionary*, kterou naše katedra provozuje na adrese <http://bosworth.ff.cuni.cz/> a kde bude automaticky poskytovat přeepsanou výslovnost jednotlivých slov.

Systém na správu a práci s pravidly pro fonetické změny lze například znovu využít pro generování výslovnosti i u jiných jazyků, které mají dostatečně pravidelný vztah mezi psanou a mluvenou formou, jako je například španělština.

Program má také didaktické využití: nutnost popsat pomocí pravidel výslovnost jazyka či jeho diachronní změny velmi přesně může také pomoci snáze odhalovat místa, která nepopisujeme dostatečně přesně či jednoznačně (a studenti si tak mohou např. vyzkoušet, zda mají přesnou představu o podobě změn či výslovnosti), případně místa, která se v jazyce chovají nepravidelně.

Tím, že je většina změn opatřena odkazy na sekundární zdroje (např. Millward nebo Strang), lze automatické vyhledání pravděpodobného vývoje použít jako startovní bod pro rychlé hledání detailnějších informací o pravděpodobném vývoji v těchto sekundárních zdrojích.

Systém práce se zástupnými znaky (wildcards) na bázi fonetických vlastností lze obdobně využít didakticky či napojit na jakékoli strojové zpracování většího množství dat.

7 Glossary of technical terms

A* A graph search algorithm which extends Dijkstra's algorithm by determining the precedence of the processing of nodes by a heuristic analysis (trying thus to guess which nodes will more probably be near to the final node).

ASCII One of the oldest encodings used, it is capable of expressing a very limited number of characters, but because of that it is also one of the most compatible ones.

Boolean Logic Logic that works with only two values: true and false (there are no intermediate values such as "rather true" etc.).

C++ a compiled computer programming language. Development in C++ is slower (compared to interpreted languages), but the execution of the resulted program is faster. As regards this thesis, it could be used to reimplement the core of the program (which is more computationally demanding).

Dijkstra An inventor of a graph search algorithm, the name is also used to denote the algorithm itself.

Edit distance The minimal number of editations necessary to change a given word into another (allowed editations being: changing a letter, adding one, deleting one).

Encoding In computer memory, characters are saved in a stream of bytes (eight succeeding bits, each of them being either 1 or 0). An encoding is a standardized plan/agreement in which particular sequences of bytes correspond to particular letters/symbols. There are dozens of encodings used nowadays, differing in storage effectivity and expressive capability.

Fuzzy logic In comparison with traditional logic, fuzzy logic is designed to work with a certain amount of vagueness or approximation.

Graph theory A mathematical model consisting of a set of nodes and a set of vertices interconnecting them. The connections may be one-way only or may be weighted (a certain numeric penalty for traversing an edge is assigned to each of them).

Heuristic analysis A function that compares different states or data configurations in terms of relative feasibility. Its use however does not guarantee complete reliability (it usually sacrifices estimation quality to computational speed).

PHP An interpreted scripting computer language. Development in PHP is fast, but the execution of the scripts is slow. It is here used for the development of the graphical user interface where the speed is not crucial and also for the core of the program itself.

Priority Queue A data container used in computer programming which may contain any number of records. It is effective in the ability to quickly absorb a new record and also can quickly produce the record with the maximal/minimal value.

Processing and memory efficiency The amount of processing time and memory necessary to complete a task, can be defined either absolutely or as a function of the parameters of the input data.

Regular expression Used to describe a set of strings (or their parts). For example, `/[bcm]at(ter)?/` describes such words as bat,cat,mat or matter.

UTF-8 A widely used encoding with a very extensive list of supported characters. Also called Unicode (UTF-8 is the most widespread flavour

of Unicode).

UNIX An historical operating system. Nowadays this term is also used to denote “UNIX-like systems” which are operating systems that share certain characteristics and functionalities.

Wildcard A meta-character (or expression) that stands for 1) any character from a given set and/or 2) variable number of characters

8 Miscellaneous Data Samples

8.1 The Phonemic Inventory

The data used for the manipulation with the phones and their characteristics are summarised in the following table. The phonetic properties in the rightmost column and Kirshenbaum’s ASCII pronunciation were taken from his article “Representing IPA phonetics in ASCII” [Kirshenbaum(1992), page 3-15] with the following exceptions: the (IPA) letter `r` is used for the alveolar approximant (we use the English-only IPA, while Kirshenbaum’s encoding mirrors the universal one), and the letter `q` is assigned only placeholder phonetic properties, because it is used only in orthography.

The particular letters used in the internal encoding are presented only as an illustration, since their assignment to particular phones is completely arbitrary and may change upon any call of the function `update_internal_phs()` function, the purpose of the internal encoding being only to reduce (for the purposes of the internal computations) the number of characters necessary to encode any phone to one (in the case of long vowels) and at the same time stay within the range of standard ASCII characters.

Table 5: Representation of Phones Used

IPA (Unicode)	Kirshenbaum	Internal Encoding	Phonetic Properties
a	a	b	low cnt unr vwl
a:	a:	H	low cnt unr vwl lng
æ	&	G	low fnt unr vwl
æ:	&:	I	low fnt unr vwl lng
b	b	a	vcd blb stp
c	c	c	vls pal stp
ç	C	V	vls pal frc
d	d	d	vcd alv stp
ð	D	x	vcd dnt frc
e	e	e	umd fnt unr vwl
e:	e:	J	umd fnt unr vwl lng
ə	@	F	mid cnt unr vwl
ə:	@:	K	mid cnt unr vwl lng
f	f	f	vls lbd frc
g	g	g	vcd vel stp
ɣ	Q	R	vcd vel frc
h	h	h	glt apr
i	i	i	hgh fnt unr vwl
i:	i:	L	hgh fnt unr vwl lng
ɪ	I	y	smh fnt unr vwl
ɪ:	I:	M	smh fnt unr vwl lng
j	j	j	pal apr
k	k	k	vls vel stp
l	l	l	vcd alv lat
m	m	m	blb nas
n	n	n	alv nas
ŋ	N	z	vel nas
o	o	o	umd bck rnd vwl
o:	o:	N	umd bck rnd vwl lng
ɔ	O	A	lmd bck rnd vwl
ɔ:	O:	O	lmd bck rnd vwl lng
p	p	p	vls blb stp

q	q	U	xxx xxx
r	r	T	alv apr
s	s	q	vls alv frc
ʃ	S	B	vls pla frc
t	t	r	vls alv stp
u	u	s	hgh bck rnd vwl
u:	u:	Q	hgh bck rnd vwl lng
ʊ	U	D	smh bck rnd vwl
ʊ:	U:	P	smh bck rnd vwl lng
v	v	t	vcd lbd frc
w	w	u	lbv apr
x	x	v	vls vel frc
y	y	S	hgh fnt rnd vwl
y:	y:	W	hgh fnt rnd vwl lng
z	z	w	vcd alv frc
ʒ	Z	E	vcd pla frc
θ	T	C	vls dnt frc

The following data are also taken from Kirshenbaum [Kirshenbaum(1992), page 3-15], and they describe the abbreviations used in the preceding table (and also in the phonetic wildcards).

Table 6: Explanation of the Phonetic Abbreviations

Abbreviation	Explanation	Phones Having the Property
vcd	voiced	b, d, g, l, v, z, ʒ, ʒ, ɣ
vls	voiceless	c, f, k, p, s, t, x, ʃ, θ, ç
blb	bilabial	b, m, p
lbd	labio-dental	f, v
dnt	dental	ð, θ
alv	alveolar	d, l, n, s, t, z, r
pla	palato-alveolar	ʃ, ʒ
pal	palatal	c, j, ç

vel	velar	g, k, x, ŋ, ɣ
lbv	labio-velar	w
glt	glottal	h
stp	stop	b, c, d, g, k, p, t
frc	fricative	f, s, v, x, z, ð, ʃ, θ, ʒ, ʝ, ʧ, ʥ
nas	nasal	m, n, ŋ
apr	approximant	h, j, w, r
vwl	vowel	ɑ, e, i, o, u, ɪ, ɔ, ʊ, ə, æ, æ:, a:, e:, ə:, i:, ɪ:, ɔ:, ɔ:, ʊ:, u:, y:, y:
lat	lateral	l
hgh	high	i, u, i:, u:, y, y:
smh	semi-high	ɪ, ʊ, ɪ:, ʊ:
umd	upper-mid	e, o, e:, o:
mid	mid	ə, ə:
lmd	lower-mid	ɔ, ɔ:
low	low	a, æ, æ:, a:
fnt	front	e, i, ɪ, æ, æ:, e:, i:, ɪ:, y, y:
cnt	center	a, ə, a:, ə:
bck	back	o, u, ɔ, ʊ, o:, ɔ:, ʊ:, u:
unr	unrounded	a, e, i, ɪ, ə, æ, æ:, a:, e:, ə:, i:, ɪ:
rnd	rounded	o, u, ɔ, ʊ, o:, ɔ:, ʊ:, u:, y, y:
lng	long	æ:, a:, e:, ə:, i:, ɪ:, o:, ɔ:, ʊ:, u:, y:

8.2 The Old English Pronunciator Sample Output

The written form of the following sample of 24 lines was taken from the Wikisource's version of the beginning of the poem "The Dream of the Rood" [Wikisource(2010)] and the pronunciation was generated completely automatically using the program.

Hwæt! Ic swefna cyst secgan wylle,
/hwæt itʃ swefna kyst sedʒan wyle/

hwæt mē gemætte tō midre nihte,
/hwæt me: jemæ:te to: midre nixte/

syðþþan reordberend reste wunedon.
/syθan reordberend reste wunedon/

þþūhte mē þþæt ic gesāwe syllicre trēow
/θu:xte me: θæt itʃ jesa:we sylitʃre tre:ow/

on lyft lādan, lēohte bewunden,
/on lyft læ:dan le:oxte bewunden/

bēama beorhtost. Eall þþæt bēacen wæs
/bæ:əma beorxtost æl θæt bæ:ətʃen wæs/

begoten mid golde; gimmas stōdon
/beyoten mid ʒolde jimas sto:don/

fægere æt foldan scēatum, swylce þþær fife wæron
/fæjere æt foldan ʃæ:ətum swyltʃe θæ:r fi:ve wæ:ron/

uppe on þþām eaxlgespanne. Behēoldon þþær engel dryhtnes ealle
/upe on θa:m æxlʒespane bexe:oldon θæ:r enʒel dryxtnes
æle/

fægere þþurh forðgesceaft; ne wæs ðær hūru fracodes gealga,
/fæjere θurx forθʒeʃæft ne wæs θæ:r hu:ru frakodes jealʒa/

ac hine þþær behēoldon hālige gāstas,
/ak hine θæ:r bexe:oldon ha:lije ʒa:stas/

men ofer moldan, ond eall þþēos mære gesceaft.
/men over moldan ond æl θe:os mæ:re jeʃæft/

Syllic wæs se sigebēam, ond ic synnum fāh,
/sylitʃ wæs se sijebæ:əm ond itʃ synum fa:x/

forwundod mid wommum. Geseah ic wuldres trēow,
/forwundod mid womum jesæx itʃ wuldres tre:ow/

wǣdum geweorðode, wynnum scīnan,
/wæ:dum jweorðode wynum fi:nan/

gegyred mid golde; gimmas hæfdon
/jeɣyred mid ɣolde jimas hævdon/

bewrigene weorðlice Wealdendes trēow.
/bewrijene weorðli:tʃe wældendes tre:ow/

Hwæðre ic þpurh þæt gold ongytan meahthe
/hwæðre itʃ θurx θæt ɣold onɣytan mæxte/

earmra ærgewin, þæt hit ærest ongan
/ærmra æ:ɣewin θæt hit æ:rest onɣan/

swǣtan on þpā swiðran healfe. Eall ic wæs mid sorgum gedrēfed,
/swæ:tan on θa: swi:ðran hæelve æl itʃ wæs mid sorɣum
jedre:ved/

forht ic wæs for þpære fæggran gesyhðe. Geseah ic þæt fūse bēacen
/forxt itʃ wæs for θæ:re fæggran jesyxðe jesæx itʃ θæt
fu:ze bæ:ætʃen/

wendan wǣdum ond blēom; hwīlum hit wæs mid wǣtan bestēmed,
/wendan wæ:dum ond ble:om hwi:lum hit wæs mid wæ:tan
beste:med/

beswyled mid swātes gange, hwīlum mid since gegyrwed.
/bezwyled mid swa:tes ɣaŋɣe hwi:lum mid sintʃe jeɣyrwed/

8.3 Sample Output of the Development Analyzer

The following output will be produced for the input words (Old English) <þunor> and (Middle English) <thunder>.

SUGGESTED DEVELOPMENT:

<þunor> (Given OE written form)

/θunor/ (Suggested OE spoken form)

short vowels were reduced to schwa in unstressed syllables (beginning in OE and continuing in ME) (FIXME: unstr.syl.) [Millward:130—date cca 900-1500]

/θunər/

after a consonant, the difference between a schwa and nothing is not very significant

/θunr/

intrusive /d/ between /n/ and a consonant [Millward:128; Strang:166—date cca 1350-1600]

/θundr/

a parasitic schwa between two consonants [Millward:130]

/θundər/ (Suggested ME spoken form)

<**thunder**> (Given ME written form)

9 Enclosure list

Only a CD is enclosed, which (besides of the PDF and DOC with the text of the thesis) contains the following files:

`oelab.php` PHP class containing the important algorithms and functionality.

`oelab.sql` MySQL dump of the database that contains the phoneme sets, rules etc.

`oeweb.tar.bz2` An archive which contains files with the web presentation that demonstrates the functions of the algorithms.

`heap.php` Priority queue implementation used by `oelab.php`. Created by Richard Ramsden, <http://www.ramsden.ca/>

References

- [Bright(1972)] Ringler Bright, Cassidy. *Bright's Old English Grammar and Reader*. Houghton Mifflin Harcourt, 1972.
- [Dijkstra(1959)] E. W. Dijkstra. A Note on Two Problems in Connection with Graphs. <http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf>, 1959.
- [Group(1997)] The Open Group. Regular expressions: The single UNIX (R) specification. <http://www.opengroup.org/onlinepubs/007908799/xbd/re.html>, 1997.
- [Kirshenbaum(1992)] Evan Kirshenbaum. Representing IPA phonetics in ASCII. <http://www.kirshenbaum.net/IPA/index.html>, February 1992.
- [Lohöfer(2007)] Astrid Lohöfer. Letter-Sound Correspondences in Middle English. http://www.staff.uni-marburg.de/~lohoefea/Sessions/S07_ME_Transcr.pdf, 2007.
- [Millward(1988)] C. M. Millward. *Biography of the English Language*. Delmar, 1988.
- [Strang(1990)] Barbara Strang. *History of English*. Routledge, 1990.
- [Strauss(1999)] Prof. Jürgen Strauss. Multimedia Editions of Medieval English Texts. <http://www.uni-trier.de/index.php?id=15573&L=2>, 1999.
- [Walther(1996)] Markus Walther. OT SIMPLE - A construction-kit approach to Optimality Theory implementation. Arbeiten des Sonderforschungsbereichs 282 'Theorie des Lexikons' Nr. 88, Seminar f. Allgemeine Sprachwissenschaft, University of Düsseldorf, Germany, October 1996. (ROA-152).

[Wikisource(2010)] Vercelli Book Wikisource. The Dream of the
Rood. [http://en.wikisource.org/wiki/The_Dream_of_the_](http://en.wikisource.org/wiki/The_Dream_of_the_Rood)
Rood, 2010.