# An Algorithm for Resolution of Time Expressions in Estonian Natural Language Dialogue Systems


## Abstract

Extraction of temporal expressions from an input text is an important step in natural language processing tasks. Automated extraction of temporal expressions can be used in dialogue systems where temporal constraints need to be enforced. The paper proposes an algorithm for processing temporal information in natural language. The algorithm was implemented as a standalone rule-based temporal expression recognizer and was made available as a web-service. Finally the implemented module was partially integrated into a spoken language dialogue system that is an interface to a theater information database.


## 1 Introduction

Temporal information can often be a significant part of meaning communicated in dialogues. There are various kinds of dialogues where people negotiate dates and times. Therefore, the automatic extraction of temporal expressions in natural language is required in building dialogue systems where temporal constraints need to be enforced.

Temporal expressions in text vary from explicit references, e.g. *June 1, 1995*, to implicit references, e.g. *last summer*, to durations, e.g. *four years*, to sets, e.g. *every month*, and to event-anchored expressions, e.g. *a year after the earthquake*. (Hacioglu, et al., 2005)

The paper proposes an algorithm for processing temporal information in natural language. The algorithm was implemented to work on Estonian texts and partially integrated to an Estonian spoken language dialogue system that is an interface to a theater information database (Treumuth, et al., 2006).

The extraction tool of time expressions was implemented as a standalone non-domain-specific module, and was made available as a web-service, that can be plugged into dialogue systems with some minor adjustments.

The time expression recognizer could be a useful software tool in the following list of currently available Estonian language technology software tools:

- Text-to-speech synthesizer (Meister, et al, 2003) (Mihkla, et al, 1999)
- Speech recognizer (speech-to-text): experimental version (Alumäe, 2004)
- Morphological analyzer + generator => spelling checker, hyphenator (Kaalep, 1997)
- Shallow syntactic analyzer => experimental versions: noun phrase extractor, text summarizer, grammar checker (Müürisep, et al, 2006)
- Word sense disambiguator: experimental version (Kaljurand, 2004)
- Dialogue act recognizer: experimental version (Fišel, 2005)

## 2 Algorithm

The rule-based algorithm involves a grammar and a parser. A top down approach is used in matching the regular expressions to input text. The more specific patterns precede the less specific ones in the matching cycle.

The input to the algorithm is text in Estonian, e.g. an utterance from the user of a dialogue system.

The output of the system is:
1. recognized time expression as text
2. recognized time expression as a logical expression (query constraint in meta SQL)

For example:

**INPUT**: *veebruari teises pooles* (second half on February)

**OUTPUT:**
RECOGNIZED: *veebruari lõpus* (in the end of February)

CONSTRAINT = DATE between 15.02.$YEAR and last_day(01.02.$YEAR)

Notice that the input term and the output term differ slightly, yet their meaning is the same. The output term is used in generating the answer and is a predefined term for each rule. This will be explained in more detail in the following sections.

The output can contain many sets of recognized expressions and constraints. The more specific ones are listed prior to less specific ones.

The Estonian Morphological Analyzer (Kaalep, 1997) was not used in generating the grammar and was also not used in parsing the grammar. The inflections and agglutinations of Estonian date expressions are easily predictable and can be handled manually. Yet, the morphological analyzer could be used in the dialogue system, that employs the temporal recognition module. Therefore, the input to the temporal recognition module coming from a dialogue system can well be morphologically analyzed, e.g. providing lemmas or base forms, if no other forms yielded a recognition result.

When integrating the parser with a dialogue system, it would be useful to get some additional input from the dialogue system in addition to the current utterance. For instance, knowing the dates that were recognized earlier in the current conversation would provide a way to accept corrections from a user, in case the user would like to clarify prior temporal expressions.

### 2.1 Grammar

The analysis for the grammar generation process involved studying some real-life dialogues that were held with a dialogue system. It turned out that the users of the dialogue system often like to query the database for intervals of time, rather than for a specific date. That is, instead of requesting information for a specific date as "*January 11th*", the users often tend to say "*something in January*".

In addition, students of computational linguistics were used in checking corpuses for various representations of Estonian time expressions, finding out all the different ways to refer to a same single date expression.

The grammar consists of 1405 rules where regular expressions are mapped to corresponding SQL constraints as follows:

regular expression ==> SQL constraint

For example:

/ (oktoober|oktoobri)\S* laupäev\S* /U
==>
weekday(DATE) = 'laupäev' and DATE between 01.10.$YEAR and last_day(01.10.$YEAR)

This rule would recognize expressions like *"oktoobris laupäeviti"*, *"oktoobri laupäevadel"* (in October on Saturdays) and the corresponding SQL constraint can be enforced on a relational database.

The grammar can handle various constructions of time expressions where names of months and weekdays are used to represent an interval of time. Following are a few examples of some date expressions that are recognized by the temporal recognition agent:

*pühapäeviti ja esmaspäeviti jaanuaris* - on Sundays and Mondays in January
*jaanuaris ja veebruaris* - in January and in February
*esmaspäeviti ja laupäeviti* - on Mondays and on Saturdays
*aprilli lõpus* - at the end of April
*juuni keskel* - in the middle of June
*oktoobri alguses* - in the beginning of October
*mais neljapäeviti* - on Thursdays in May

If the regular expressions in grammar are matched to a natural language input, the corresponding SQL constraints are integrated into a SQL query's template *WHERE clause* as follows:

<SELECT clause>
<FROM clause>
<WHERE clause
**[temporal constraints]**
[all other constraints]>

For example:

SELECT title
FROM performances
WHERE
**weekday(DATE) = 'Saturday' and DATE between 01.10.2007 and last_day(01.10.2007)**
[all other constraints]

Upon execution of this query the dialogue system would return the performances that match the time constraint and other constraints. These SQL constraints can easily be altered to suit the needs of a specific database. Also the functions *weekday* and *last_day* are available in most database engines or can easily be implemented.

It was more efficient and extendable to create an explicit grammar, rather than trying to implement an implicit black-box program to cope with these expressions. The grammar is residing in a text file (outside of program code) and can easily be altered and extended. This approach is quite similar to the one described by Berglund (2005).

### 2.2 Deictic Expressions

Deictic expressions are expressions that refer to temporal aspect of an utterance and depends on the context in which they are used (Wiebe, et al., 1998). For example "*tomorrow*" depends on current date and is recognized as "*current date + 1 day*" (with respect to the conversation date).

The grammar currently contains a non-terminal *$YEAR*, that is used to enforce dependencies to current date by avoiding looking in past dates. No other deictic expressions are represented in grammar. The algorithm copes with deictic expressions in a separate parser. It can recognize patters like "*on weekends*", "*day after tomorrow*", "*today*", "*next Monday*" and so on.

## 3 Extensions to Grammar

### 3.1 Answer Phrases

While the grammar is used to recognize time expressions and execute queries based on returned constraints, there is also need to provide input for the answer generation, as the answer should also contain the recognized time expression in correct form.

For that reason, the grammar that was described above, was extended by adding the recognized term into the rule as follows:

```
oktoobris laupäeviti
==>
/ (oktoober|oktoobri)\S* laupäev\S* /U
==>
weekday(DATE) = 'laupäev' and DATE between
01.10.$YEAR and last_day(01.10.$YEAR)
```

The recognized term, in correct form, can be used in generating an answer to the user by plugging it in a sentence. Assume a conversation:

<User>: Are there any performances in **October on Saturdays**?
<System>: Here are the plays that I found in **October on Saturdays** …

The pattern can match multiple formats, yet the answer phrase can be fixed to one format, as the rules are built to support this approach.

### 3.2 Constraint Relaxation

Partial constraint relaxation is implemented in a dialogue system that uses the temporal constraint grammar, yet the rules for constraint relaxation are not defined in the grammar.

For example, the user might mention a date, that would result in *"not found"* response. Then it would be appropriate to relax this constraint, as in the following dialogue.

<User>: Are there any performances on **Saturdays**?
<System>: No, yet I found one on this **Sunday** …

Here we saw an example of a constraint relaxation where the original date constraint was relaxed by adding one day. This way the users of the system can receive some alternative choices, instead of plain *"not found"* responses.

The constraint relaxation properties can be held in the grammar as long as they stay separate from the dialogue domain.

### 3.3 Correction Questions

There are a some problems with deictic expressions that can be solved by correction questions.

For example, if user mentions the word "*weekend*" on Sunday evening, does the user mean next weekend or the current weekend.

The correction questions are not implemented in the grammar, as they tend to be domain specific. The grammar could be extended by adding correction questions and choices for corresponding answers, also as long as they stay separate from the dialogue domain.

## 4 Conclusion

The paper has described an algorithm that was implemented as a standalone automated extraction tool for processing temporal information in Estonian natural language. This rule-based approach can be used for other languages (English). The main idea and benefit of current approach is the output of logical expressions that can be used in SQL queries.

The rule-based approach was chosen, as it turned out to save a lot of time to implement an explicit grammar by automatically generating hundreds of rules and a parser (regular expression pattern matching), rather than trying to implement a clever, yet implicit black-box algorithm to cope with all these rules. Also a small grammar generator was built, which is able to re-generate the grammar of 1405 rules, making it easy to manage and extend the grammar.

The grammar can be improved in using constraint relaxation options and predefined question-answer sets for correction sub-dialogues.

It would also be useful to get some additional input from the dialogue system in addition to the current utterance. For instance, knowing the dates that were recognized earlier in the current conversation would provide a way to accept corrections from a user, in case the user would like to clarify prior temporal expressions.

The time expression recognizer (e.g. as tagger) could be a useful software tool among the other currently available Estonian language technology software tools.

## References


Anders Berglund. Extracting Temporal Information and Ordering Events for Swedish. *Master's thesis report.* 2004.

Einar Meister, J. Lasn, L. Meister. SpeechDat-like Estonian database. - In: *Text, Speech and Dialogue* : 6th International Conference, TSD 2003, Czech Republic, September 8-12, 2003 / Eds. Matoušek [et al.]. Berlin [etc.] : Springer, Lecture Notes in Artificial Intelligence, Vol. 2807. 412-417. 2003.

Heiki-Jaan Kaalep. An Estonian Morphological Analyser and the Impact of a Corpus on Its Development. *Computers and the Humanities* 31: 115-133. 1997.

J. M. Wiebe, T. P. O'Hara, T. Ohrstrom-Sandgren, and K. J. McKeever. (1998). An Empirical Approach to Temporal Reference Resolution. *Journal of Artificial Intelligence Research*, 9, 247-293. 1998.

Kaarel Kaljurand. Word Sense Disambiguation of Estonian with syntactic dependency relations and WordNet. *In Proc. ESSLLI-2004*, Nancy, France, 128-137. 2004.

Kadri Hacioglu, Ying Chen, and Ben Douglas. Automatic Time Expression Labeling for English and Chinese Text. *In Proceedings of CICLing-2005*, pages 348-359; Springer-Verlag, Lecture Notes in Computer Science, Vol. 3406. 2005.

Kaili Müürisep, Heli Uibo. Shallow Parsing of Spoken Estonian Using Constraint Grammar. In: *Treebanking for Discourse and Speech. Proceedings of NODALIDA-2005 special session on treebanking*: NODALIDA-2005 special session on treebanking, Joensuu, 2005. (Ed.) Peter Juel Henrichsen, Peter Rossen Skadhauge. Frederiksberg, Denmark: Samfundslitteratur, 105 - 118. 2006.

Margus Treumuth, Tanel Alumäe, Einar Meister. A Natural Language Interface to a Theater Information Database. *Proceedings of the 5th Slovenian and 1st International Language Technologies Conference 2006 (IS-LTC 2006)*, 27-30. 2006.

Mark Fišel. Dialogue Act Recognition in Estonian Dialoguesusing Artificial Neural Networks, *Proceedings of the International Conferenc The Second Baltic Conference on Human Language Technologies*, 231-235; 2005.

Meelis Mihkla, A. Eek, E. Meister. Text-to-Speech Synthesis of Estonian. – *Proceedings of the 6th European Conference on Speech Communication and Technology*, Budapest, Vol. 5 2095-2098. 1999.

Tanel Alumäe. Large Vocabulary Continuous Speech Recognition for Estonian Using Morphemes and Classes. *TSD 2004*: 245-252. 2004.