Representing Calendar Expressions with Finite-State Transducers that Bracket Periods of Time on a Hierarchical Timeline

Jyrki Niemi and **Kimmo Koskenniemi**
University of Helsinki,
Department of General Linguistics,
PO Box 9, FI–00014 University of Helsinki, Finland
`{jyrki.niemi, kimmo.koskenniemi}@helsinki.fi`

Abstract

This paper proposes representing the semantics of natural-language calendar expressions as a sequence of compositions of finite-state transducers (FSTs) that bracket the denoted periods of time on a finite timeline of nested calendar periods. In addition to simple dates and times of the day, the approach covers more complex calendar expressions. The paper illustrates the model by walking through the representation of the calendar expression *January to March and May 2007*. The representation of the expressions considered is compositional with reference to their subexpressions. The paper also outlines possible applications of the model, based on finding the common periods of time denoted by two calendar expressions.

1 Introduction

Temporal information is essential in various applications, for example, in event calendars and appointment scheduling. A common class of temporal information are calendar expressions, which range from simple dates and times of the day to more complex ones, such as *the second Tuesday following Easter*. In applications, calendar expressions and other temporal information should often be both processed by software and presented for a human to read.

Numerous approaches and models have been developed to represent and process temporal information in various fields, from temporal databases to natural-language processing. Temporal information often contains cycles and repetition, such as the cycle of hours within a day. A natural means to process cyclical structures could be provided by finite-state methods. These methods have a sound theoretical basis, and they are easier to control than ad hoc methods. Finite-state transition networks may also provide a usable representation for sparse sets of sets. Despite these advantages, explicitly finite-state methods and representations seem to have been relatively little used in temporal representation and reasoning. (However, see Sect. 4 for previous work.) Although these methods indeed have their limitations, in particular their restricted numerical calculation ability, we believe that they would suit well to representing and processing various kinds of temporal information.

In this paper, we use finite-state transducers (FSTs) in representing the semantics of calendar expressions and in finding the common periods of time denoted by two or more calendar expressions. We start from the (intensional) meaning of a natural-language calendar expression; we do not treat the extraction of the meaning from the original expression. We use FSTs to mark the denotations of calendar expressions with brackets on a timeline string. Thus we call the representation presented here the bracketing FST model. We also use the composition operation of FSTs both to construct more complex expressions from simple ones in a compositional way, and to find the common periods of time denoted by several expressions.

Some aspects of the present bracketing FST model are based our earlier model presented in

Niemi et al. (2006). That temporal model and representation of calendar expressions was based on a string of hierarchical calendar periods, expanded to finer ones by FSTs as needed. Although the approach seemed to promise to make reasoning tractable, it was non-compositional and rather procedural. In that respect, we regard the present model as a marked improvement.

The rest of the paper is organized as follows. Section 2 presents the basic principles of the bracketing FST model and illustrates them with examples. Section 3 describes some possibilities of simple temporal reasoning using the model in such applications as event calendars and appointment scheduling. Section 4 presents some related work in temporal representation and reasoning research. Section 5 concludes the paper with discussion and some directions for further research.

2 Calendar Expressions in Bracketing FST Model

In this section, we present the basic principles of the bracketing FST model. We illustrate the principles primarily with the representation of a fairly simple calendar expression, *January to March and May 2007*. The expression contains four basic calendar expressions corresponding to specific calendar periods of the Gregorian calendar, combined with an interval, a list and a refinement. Before the example, we briefly discuss the levels of representation of calendar expressions.

2.1 Calendar Expressions and their Levels of Representation

A calendar expression generally denotes a period of time that does not depend on the time of use of the expression, such as *25 May 2007*. However, the denotation may be vague or underspecified without context, as in *September* or *in the morning*. The denotation may also be ambiguous; for example, *a week* may denote either a calendar period or a duration. In this paper, we model the disambiguated meanings of natural-language calendar expressions, while trying to retain underspecification wherever possible.

A calendar expression can denote disconnected periods (non-convex intervals) of time, as well as connected ones. For example, *two Sundays* denotes a period consisting of two Sundays without the intervening days.

Calendar expressions can be represented at several different levels. We distinguish between the following levels of representation: (1) a natural-language calendar expression: *January to March 2007*; (2) a semi-formalized term representation of its semantics: intersect(interval(mon(jan), mon(mar)), year(y2007)); (3) the representation of this as a regular (relation) expression or a sequence of compositions of them; (4) the FST constructed from the regular expression; and (5) the string or set of strings specified by the FST composed with a string representing a timeline. In this paper, we present natural-language expressions, regular expressions at the level of macros and composed timeline strings. The parametrized regular relation macros can be regarded as the basic building blocks of calendar expression FSTs.

Our work does not cover the conversion of a natural-language calendar expression to the term representation. Instead, we assume the semantic term representation as a starting point. The semantic representation is then converted into a regular expression, which is further compiled into an FST.

The term representation is similar to the calendar XREs (extended regular expressions) of Niemi and Carlson (2006) in that they are structurally fairly close to natural language calendar expressions. Thus it should be relatively simple to generate a natural-language calendar expression from the term representation. It should also be possible to parse a natural-language calendar expression to a term representation.

2.2 FST Expressions and Macros

We represent the denotation of the example expression *January to March and May 2007* as the complex term intersect(union(interval(mon(jan), mon(mar)), mon(may)), year(y2007)). This in turn translates to the following sequence of compositions of FSTs. (The parameters i*n* refer to marker bracket indices, explained below.)

$$\begin{aligned}&\mathit{mon}(\mathrm{Jan}, \mathrm{i1}) \circ \mathit{mon}(\mathrm{Mar}, \mathrm{i2})\\ &\circ \mathit{interval}(\mathrm{i1}, \mathrm{i2}, \mathrm{i3})\\ &\circ \mathit{mon}(\mathrm{May}, \mathrm{i4}) \circ \mathit{union}(\mathrm{i3}, \mathrm{i4}, \mathrm{i5})\\ &\circ \mathit{year}(\mathrm{y2007}, \mathrm{i6}) \circ \mathit{intersect}(\mathrm{i5}, \mathrm{i6}, \mathrm{i7})\end{aligned}$$

Remarkably, this composition sequence corresponds directly to a postfix representation of the term expression.

The denotation of a calendar expression is represented by indexed marker brackets that delimit the denoted periods on a timeline. Marker brackets are added by FSTs corresponding to either basic calendar expressions or operations combining simpler expressions to more complex ones. Basic expression FSTs add new marker brackets to each calendar period denoted by the expression, whereas operation FSTs add brackets based on their operands that are denoted by previously added brackets. A composite expression is represented as a sequence of compositions of such FSTs.

We present FSTs at the level of simple parametrized macros, as above. For example, the FST constructed from *mon*(Jan, i1) marks each January with the marker brackets i1, whereas *intersect*(i5, i6, i7) marks with the marker brackets i7 the periods of time that are inside both marker brackets i5 and i6. More generally, each operation FST macro takes as its arguments marker bracket indices corresponding to the subexpressions of a calendar expression.[1] The last argument of each macro indicates the marker bracket index with which the FST marks the result of the operation.

The representation of a calendar expression is compositional as each operation operates on the indicated periods marked on the timeline and marks its own denotation on the timeline.

2.3 The Representation of a Timeline

To illustrate the representation of a calendar expression, we need a timeline on which to mark the denotation of the expression. We use a simplified timeline consisting solely of the year 2007 at the level of months. We begin with the following timeline without marker brackets. (We separate the symbols of a string with spaces.)

[y y2007 [m Jan m] [m Feb m] [m Mar m] [m Apr m] [m May m] [m Jun m] . . . [m Dec m] y]

Largely following Niemi et al. (2006), we represent a finite timeline as a string consisting of hierarchical (nested) markings for different calendar periods. Each calendar period is delimited by granularity-specific begin and end markers: for example, [y marks the beginning of a year and m] marks the end of a month. A begin marker is followed by a symbol indicating a specific period, such as y2007 for the year 2007 and Jan for a January. A day is marked for both the day of the month and the day of the week. The period indicator may be followed by a sequence of markers for a finer granularity.

A timeline string can be constructed by a sequence of compositions of FSTs that expand the timeline a granularity at a time to finer granularities, for example, a year to contain months.[2] Granularities need not be strictly nested, which allows the representation of weeks. The level of detail in a timeline can vary: for example, if hours are not referred to in the expression, they are not needed in the timeline.

A calendar expression is represented on a timeline by enclosing the denoted periods of time in marker brackets which have an index corresponding to the expression: {*in* . . . }*in*. The indices distinguish between the denotations of different subexpressions in a composite expression.

2.4 Basic Calendar Expressions

Basic calendar expressions correspond to the basic periods of the Gregorian calendar. Basic calendar periods include both generic periods, such as hour, day, month and year, and specific ones, such as each hour, day of the week, day of the month, month and year. We also assume expressions for seasons and holidays, such as Easter and Christmas Day.

A basic calendar expression such as *January* is represented as an FST that adds marker brackets on the timeline around each period denoted by the expression.[3] In the present example (see Sect. 2.2), the first FST *mon*(Jan, i1) adds marker brackets i1 around each month identified by the symbol Jan,

[1] Macros may also have integer arguments.

[2] To be able to expand the months of a year and the days of a month independently of the neighbouring periods, each year contains symbols indicating its leap-year status and the day of the week of its first day, and each month, the number of its days and the day of the week of its first day. However, for clarity, we omit this information in the examples of this paper.

[3] Although unqualified natural-language calendar expressions, such *January*, typically refer to the nearest past or future period relevant in the context, in this work we interpret them as underspecified, for example, referring to any January.

that is, each January: {i1 [m Jan ... m] }i1. Similarly, *mon*(Mar, i2) adds i2 around each March. The timeline is now as follows. (Boldface indicates brackets added at this stage.)

> [y y2007 **{i1** [m Jan m] **}i1** [m Feb m] **{i2** [m Mar m] **}i2** [m Apr m] [m May m] ... [m Dec m] y]

Basic calendar expressions form the basis for more complex expressions. The complex example expression *January to March and May 2007* contains three basic constructs used to combine calendar expressions: an interval, a list and a refinement. We treat each of them in the following subsections.

2.5 Interval

In the example, the subexpression *January to March* denotes an interval which begins from the beginning of a January and ends at the end of the closest following March. The January and March marked above are combined to the interval by the FST *interval*(i1, i2, i3), which marks with i3 the intervals beginning from i1 and ending to i2:

> [y y2007 **{i3** {i1 [m Jan m] }i1 [m Feb m] {i2 [m Mar m] }i2 **}i3** [m Apr m] [m May m] ... [m Dec m] y]

In general, an interval FST adds a begin marker bracket for the result at the beginning of each begin period of the interval and an end marker bracket at the end of the closest following end period.

2.6 List

The next step is to combine the interval *January to March* with *May* to a list representing *January to March and May*. We treat all list expressions as disjunctive; for example, *January and May* is interpreted as “any January or May”.

In the example, the FST *mon*(May, i4) marks May with i4, and *union*(i3, i4, i5) adds i5 around the periods marked with i3 or i4 or both:

> [y y2007 **{i5** {i3 {i1 [m Jan m] }i1 [m Feb m] {i2 [m Mar m] }i2 }i3 **}i5** [m Apr m] **{i5 {i4** [m May m] **}i4 }i5** ... [m Dec m] y]

More generally, we represent a list as the union (disjunction) of its elements. A (binary) union FST adds result marker brackets around the periods covered by at least one of the operands.

2.7 Refinement

Finally, we combine the previous subexpression with the year 2007 to the final expression *January to March and May 2007*. We call this construct a refinement, following Niemi et al. (2006), as each of the subexpressions refines or restricts the denoted period of time. We implement refinement as an intersection (conjunction). In the example, the FST *year*(y2007, i6) marks the year 2007 with i6, and *intersect*(i5, i6, i7) marks the denotation of the final expression with i7:

> **{i6** [y y2007 **{i7** {i5 {i3 {i1 [m Jan m] }i1 [m Feb m] {i2 [m Mar m] }i2 }i3 }i5 **}i7** [m Apr m] **{i7** {i5 {i4 [m May m] }i4 }i5 **}i7** ... [m Dec m] y] **}i6**

An intersection FST adds result marker brackets around periods covered by both the operands.

From this final timeline, we can then remove all other marker brackets except the i7s that mark the denotation of the whole expression:

> [y y2007 {i7 [m Jan m] [m Feb m] [m Mar m] }i7 [m Apr m] {i7 [m May m] }i7 ... [m Dec m] y]

We now take a brief look inside an FST expression, followed by an outline of the representation of a few other calendar expression constructs and discussion about the interpretation of calendar expressions.

2.8 The Implementation of an FST Expression

The FST expressions used above in representing calendar expressions are rather abstract in that they use macros which correspond to parametrized regular relation expressions. For example, the following macro implements the union operation:

$$\begin{array}{l} \mathit{union}(i, j, r) = \\ \quad \mathit{noijrs} \\ \quad\; .\, (\varepsilon{:}\{r \,.\, (\{i \cup \{j) \\ \qquad .\, (\mathit{noijrs} \cup ((\{i \cup \{j) \,.\, \mathit{noijrs} \,.\, (\}i \cup \}j)))^{*} \\ \qquad .\, (\}i \cup \}j) \,.\, \varepsilon{:}\}r \,.\, \mathit{noijrs})^{*} \end{array}$$

Here *noijrs* is an abbreviation of the expression $(\sim(\{i \cup \}i \cup \{j \cup \}j \cup \{r \cup \}r))^{*}$, ε denotes the empty string, $A{:}B$ the transduction from symbol A to B, $A\,.\,B$ the concatenation of A and B, and $\sim A$ the symbols of the alphabet excluding those in A.

In effect, the expression inserts the marker brackets $\{r \ldots \}r$ around each brackets $\{i \ldots \}i$ or $\{j \ldots \}j$, which may contain or overlap with one or more instances of the other pair of marker brackets. The definition assumes that marker brackets with a certain index appear neither nested nor overlapping with themselves. Moreover, this slightly simplified version does not consider the special cases that may occur at the beginning or the end of the timeline, such as an end marker bracket not preceded by a corresponding begin marker bracket.

2.9 Other Calendar Expression Constructs

In addition to intervals, lists and refinement expressions, it is relatively straightforward to represent in the bracketing FST model also various more complex types of calendar expressions. They include constructs presented in Niemi et al. (2006): exception expressions (*8 am, except Mondays 9 am*), various kinds of anchored expressions (*the second Tuesday following Easter*) and consecutiveness expressions (*two consecutive Sundays*). Furthermore, the model can represent a number of deictic and anaphoric temporal expressions. In this subsection, we outline the representation of these types of expressions. In several examples below, we present the corresponding term representation instead of or in addition to the FST macro representation.

The expression *8 am, except Mondays 9 am* (except(h08, mon, h09)) is an exception expression. Such an expression consists of two or three parts: a default time, an exception scope and an optional exception time. In the above expression, *8 am* is the default time (*dt* below), *Mondays* the exception scope (*es*) and *9 am* the exception time (*et*). Following Carlson (2003), we express the denotation of an exception expression with union, difference and intersection as $(dt \setminus es) \cup (es \cap et)$. The union and intersection operations are described above, and the difference $dt \setminus es$ corresponds to $dt \cap \neg es$. The complement $\neg es$ is in turn implemented by the macro $\mathit{complement}(i, r)$, which precedes each $\{i$ with a $\}r$ and follows each $\}i$ with a $\{r$. For example, the timeline with marker brackets $\ldots\{i\ \ldots\}i\ \ldots\{i\ldots\}i\ldots$ becomes $\{r\ldots\}r\ \{i\ldots\}i\ \{r\ldots\}r\ \{i\ldots\}i\ \{r\ldots\}r$.

An anchored expression denotes a time relative to an anchor time. For example, *the second Tuesday following Easter* (nth_following(2, tue, easter)) refers to a time relative to Easter. To obtain the denotation of the expression, FSTs first mark each Tuesday and Easter with the respective marker brackets ($\mathit{wday}(\mathrm{Tue}, \mathrm{i1}) \circ \mathit{easter}(\mathrm{i2})$). The FST $\mathit{nth\_following}(2, \mathrm{i1}, \mathrm{i2}, \mathrm{i3})$ then inserts result marker brackets i3 around each Tuesday that is preceded by Easter, with exactly one Tuesday in between. This condition is fairly simple to express in regular relation expressions.[4]

The consecutiveness expression *two consecutive Sundays* (consecutive_n(2, sun)) differs from the other constructs above in that it represents a set of disconnected periods of time, and thus it should be interpreted disjunctively (see Sect. 2.10 below). However, it is simple to implement as an FST that marks any two Sundays with no Sundays in between them and leaves the rest of the timeline intact.

Among other calendar expression constructs that we have represented in the model are parity expressions (*even Tuesdays of the month*), ordinal expressions (*every second Monday of the year*) and containment expressions (*the months with a Friday 13th*). These expressions can be represented in a manner fairly similar to anchored expressions.

The present model can also be extended fairly straightforwardly to simple deictic and anaphoric temporal expressions, such as *today* and *the following month*. Similarly to Carlson (2003), we define FSTs that mark speech time and the reference time of an anaphoric expression: $\mathit{now}(i)$ and $\mathit{then}(i)$, respectively. With their help, we can represent *today* as containing(day, now) (the day containing the speech time) and *the following month* as following_nth(1, month, then) (the first calendar month following the reference time).

2.10 Conjunctive and Disjunctive Interpretations

Many calendar expressions can be interpreted either conjunctively or disjunctively. For example, *Monday and Thursday* might denote either "a (certain) Monday and a (certain) Thursday" or "any Monday or Thursday", depending on the point of view. The conjunctive interpretation would be feasible for someone who provides a service on the days in question, and the disjunctive one for a client.

[4]The first argument of the macro *nth_following* is an integer n. In the macro definition it translates to the concatenation power R^{n-1}, where R is the part of the regular expression to be repeated $n-1$ times.

In the present work, we generally represent the denotation of a calendar expression conjunctively, on a single timeline string, whenever possible. Effectively, the denotation of *Monday and Thursday* is a single disconnected period of time that comprises all Mondays and Thursdays.

However, a purely conjunctive representation would provide an incorrect representation for some types of expressions. In particular, we need to represent disjunctively expressions denoting multiple separate disconnected periods of time and expressions with several possible overlapping denotations. Both of these properties are true of the expression *(any) two Sundays*, for example: if all the possible combinations of two Sundays were represented on a single timeline, it would in effect denote all Sundays. Instead, each possible combination of two Sundays is represented on its own timeline. This set of timelines is represented as an FST with many possible paths, each corresponding to a single timeline.

3 Temporal Reasoning Applications

The bracketing FST model could be used in temporal reasoning. We have mainly considered a form of reasoning that finds the common periods of time denoted by two calendar expressions. Such reasoning could be used, for instance, in querying an event database or in appointment scheduling.

A query to an event database could find out, for example, at what time certain museums are open on Fridays in May, or which museums are open on Fridays in May. For both queries, we should compute the common periods of time denoted by both the query and the target expression. The answer to the first query would be a representation of the common periods, and to the second one, the existence or non-existence of common periods. In our model, we would mark the denotations of both expressions on the same timeline and compute their intersection using the same operation as for a refinement within a single expression.

Although this kind of reasoning might not be efficient enough for large-scale on-line processing, it might be of use in an application in which some information is generated periodically from a database, such as a Web page showing events of the week. Such an application would also need a generation component that would generate from the term representation the corresponding natural-language calendar expressions, possibly in multiple languages, along the lines of Niemi and Carlson (2006).

In appointment scheduling we should find the common free periods of time in the calendars of two or more people or institutions. In addition, the appointment may have further constraints, such as *at least six hours without interruptions*. Finding common periods of time with intersection could be applied to appointment scheduling as well.

4 Related Work

Temporal expressions have been widely researched, including modelling and reasoning with calendar expressions. However, finite-state methods have seldom been used explicitly. In the following, we very briefly relate our work to a few other pieces of work in this area.

Our original inspiration was Carlson's (2003) event calculus, which includes modelling calendar expressions as extended regular expressions (XREs). In Niemi and Carlson (2006) we represented a number of calendar expression constructs as XREs; however, the approach generally appeared computationally intractable.

The Verbmobil project (Wahlster, 2000) had its own formalism to represent and reason with temporal expressions in appointment negotiation dialogues (Endriss, 1998). Its coverage of calendar expressions was similar to our present approach.

The calendar logic of Ohlbach and Gabbay (1998) and its time term specification language can represent calendar expressions of various kinds. However, compared to our term representation, the structure of calendar logic expressions differs significantly more from that of natural-language calendar expressions.

Han and Lavie (2004) and Han et al. (2006) use their own formalism in conjunction with reasoning using temporal constraint propagation. They cover more types of expressions than we do, including durations and underspecified and quantified expressions, such as *every week in May*.

TimeML (Boguraev et al., 2005) is a markup language for marking events and temporal expressions and their relationships in a text. Compared to our present approach, TimeML covers many more kinds of temporal expressions, such as temporal relations, durations and underspecification. However, as Han et al. (2006) point out, in many cases TimeML expresses only the denotation of the expression, such as a certain date, whereas our bracketing FST expressions, as well as and the TCNL representation of Han et al. (2006), represent the intensional meaning of a calendar expression. Moreover, it is unclear how TimeML would suit to the kind of reasoning that we have considered.

Finite-state methods have been explicitly used by a few people to represent or reason with temporal information, but the scope of their work is otherwise mostly rather different from that of ours. Karttunen et al. (1996) express the syntax of fairly simple dates as regular expressions to check their validity. Fernando (2004) uses regular expressions to represent events with optional temporal information, such as *(for) an hour*. Focusing on events, his examples contain only rather simple temporal expressions. Finally, Dal Lago and Montanari (2001) and Bresolin et al. (2004) present automata models that are suitable in particular for representing infinite granularities. Their models are based on Büchi automata.

5 Discussion and Further Work

In the following, we discuss some advantages and disadvantages of the bracketing FST model of calendar expressions presented in this paper. We also mention some directions for further research.

Compared our earlier FST-based temporal model (Niemi et al., 2006), the bracketing FST model is clearly more compositional and less procedural but less efficient in at least some aspects. For instance, it is an open question if it is possible to compute the denotation of an expression a granularity level at a time, or to expand on the timeline only the periods of time relevant to the expression, as in the earlier model. We thus use a pre-expanded timeline, which may be rather long in practical applications.

The FST compositions for calendar expressions can quickly result in very large automata unless we begin the composition sequence from the timeline. In some applications it might nevertheless be useful to combine the expression FSTs before composing them with a timeline, so we also intend to investigate other options for keeping the size of the FSTs manageable even for complex expressions.

Since we use a different marker bracket index for each subexpression, the number of different marker bracket symbols is proportional to the number of nodes in the expression tree of a term expression. This number may be fairly large for complex calendar expressions. An alternative could be to reduce the number of indices by reusing them; for example, an operation could use the brackets of one of its operands to mark the result. However, we probably should preserve the indices of reoccurring subexpressions to avoid having to recompute them.

We found it relatively easy to build FST expressions for the various calendar expression constructs, even though it might be partly explained by the fact that only few special cases were treated. To widen the coverage, we intend to try to represent in the bracketing FST model all the calendar expression constructs mentioned in Niemi and Carlson (2006), and relevant ones probably also from Endriss (1998) and Han and Lavie (2004). We should also test the scalability of the representation to longer and more complex calendar expressions.

Duration expressions are an area still to be investigated in the bracketing FST model, although they can be represented with finite-state methods, as seen in Niemi and Carlson (2006). The representation of durations in general may be intractable, since a disconnected duration, such as in *40 hours a week*, might be composed of an arbitrary number of arbitrarily short periods of time. However, a tractable subset should suffice in a practical application.

Even though finite-state methods can be used to represent many different types of calendar expression constructs, they are unable naturally to represent fuzzy or inexact expressions, such as *about 8 o'clock*, internally anaphoric expressions, such as *9.00 to 17.00, an hour later in winter*, fractional expressions, such as *the second quarter of the year*, or arbitrary repetition expressions, such as *two equally long periods of time*. However, it might be possible to compile at least some expressions of some of these types to a finite-state representation.

Furthermore, there are types of temporal expressions that we have not yet considered but that probably can be represented with finite-state methods. For example, one might want to find in an event database the days on which one could take a boat cruise followed by one to two hours before a theatre play.

We have used the Xerox Finite-State Tool (XFST) (Karttunen et al., 1997) as the main tool for experimenting with regular relation (FST) expressions. However, it would be essential to develop an accompanying compiler to compile term representations of calendar expressions into compositions of FST expressions, instead of translating them by hand.

In summary, although several issues should be investigated and solved before the bracketing FST model could be useful in a practical application. we find it more promising in many respects than that of Niemi et al. (2006). In general, we consider finite-state approaches fairly well suited to modelling the semantics of calendar expressions.

Acknowledgements

This paper represents independent work by the junior author based on the suggestions of the senior author. We are grateful to the anonymous reviewers for their valuable comments. We are also grateful to Nathan Vaillette (personal communication) for his original suggestion to use finite-state transducers in representing calendar expressions.

References

Branimir Boguraev, Jose Castaño, Rob Gaizauskas, Bob Ingria, Graham Katz, Bob Knippen, Jessica Littman, Inderjeet Mani, James Pustejovsky, Antonio Sanfilippo, Andrew See, Andrea Setzer, Roser Saurí, Amber Stubbs, Beth Sundheim, Svetlana Symonenko, and Marc Verhagen. 2005. TimeML: A formal specification language for events and temporal expressions. http://timeml.org/site/publications/timeMLdocs/ timeml_1.2.1.html, October.

Davide Bresolin, Angelo Montanari, and Gabriele Puppis. 2004. Time granularities and ultimately periodic automata. In José Júlio Alferes and João Leite, editors, *Logics in Artificial Intelligence: 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27–30, 2004, Proceedings*, number 3229 in Lecture Notes in Computer Science, pages 513–525, Heidelberg, September. Springer-Verlag.

Lauri Carlson. 2003. Tense, mood, aspect, diathesis: Their logic and typology. Unpublished manuscript, February.

Ugo Dal Lago and Angelo Montanari. 2001. Calendars, time granularities, and automata. In Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras, editors, *Advances in Spatial and Temporal Databases: 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12–15, 2001, Proceedings*, number 2121 in Lecture Notes in Computer Science, pages 279–298, Heidelberg, July. Springer-Verlag.

Ulrich Endriss. 1998. Semantik zeitlicher Ausdrücke in Terminvereinbarungsdialogen. Verbmobil Report 227, Technische Universität Berlin, Fachbereich Informatik, Berlin, August.

Tim Fernando. 2004. A finite-state approach to events in natural language semantics. *Journal of Logic and Computation*, 14(1):79–92.

Benjamin Han and Alon Lavie. 2004. A framework for resolution of time in natural language. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):11–32, March.

Benjamin Han, Donna Gates, and Lori Levin. 2006. From language to time: A temporal expression anchorer. In James Pustejovsky and Peter Revesz, editors, *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 196–203. IEEE Computer Society.

L[auri] Karttunen, J[ean]-P[ierre] Chanod, G[regory] Grefenstette, and A[nne] Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–328, December.

Lauri Karttunen, Tamás Gaál, and André Kempe. 1997. Xerox finite-state tool. Technical report, Xerox Research Centre Europe, Grenoble, France, June. http://www.xrce.xerox.com/ competencies/content-analysis/fssoft/docs/fst-97/xfst97.html.

Jyrki Niemi and Lauri Carlson. 2006. Modelling the semantics of calendar expressions as extended regular expressions. In Anssi Yli-Jyrä, Lauri Karttunen, and Juhani Karhumäki, editors, *Proceedings of the FSMNLP 2005*, number 4002 in Lecture Notes in Artificial Intelligence, pages 179–190. Springer.

Jyrki Niemi, Kimmo Koskenniemi, and Lauri Carlson. 2006. Finite-state transducers and a variable-granularity timeline in modelling the semantics of calendar expressions. In Hans W. Guesgen, Gérard Ligozat, Jochen Renz, and Rita V. Rodriguez, editors, *ECAI 2006 Workshop. Spatial and Temporal Reasoning*, pages 56–62, August.

Hans Jürgen Ohlbach and Dov Gabbay. 1998. Calendar logic. *Journal of Applied Non-classical Logics*, 8(4):291–324.

Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Artificial Intelligence. Springer, Berlin.