A Framework for Resolution of Time in Natural Language

BENJAMIN HAN and ALON LAVIE
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213

Automatic extraction and reasoning over temporal properties in natural language discourse has not had wide use in practical systems due to its demand for a rich and compositional, yet inference-friendly, representation of time. Motivated by our study of temporal expressions from the Penn Treebank corpora, we address the problem by proposing a two-level constraint-based framework for processing and reasoning over temporal information in natural language. Within this framework, temporal expressions are viewed as partial assignments to the variables of an underlying calendar constraint system, and multiple expressions together describe a temporal constraint-satisfaction problem (TCSP). To support this framework, we designed a typed formal language for encoding natural language expressions. The language can cope with phenomena such as under-specification and granularity change. The constraint problems can be solved using various constraint propagation and search methods, and the solutions can then be used to answer a wide range of time-related queries.

Categories and Subject Descriptors: I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods---Representation languages, Representations (procedural and rule-based); I.2.7 [**Artificial Intelligence**]: Natural Language Processing---Language parsing and understanding, Text analysis.

General Terms: Languages, Design, Theory.

Additional Key Words and Phrases: Temporal Information Processing, Temporal Reasoning, Computational Semantics, Knowledge Representation, Constraint Solving.

1. INTRODUCTION

Time plays an active role in all facets of our lives, yet in many practical systems that perform automatic analysis of natural language (NL), real time has long been a forgotten dimension. Incorporating time into these systems is by no means trivial: to start with, such systems need a rich yet compositional representation for encoding the nuances of time manifested in NL, and the encoding needs to be inference-friendly to facilitate sophisticated reasoning. Achieving these goals would require work on designing a syntax-semantics interface for time, crafting formal models for real calendars (including non-Gregorian ones), coping with *under-specification* and *granularity change* in temporal expressions, accounting for the effect of *temporal focus* shifting, and finally reasoning about a prescribed temporal scenario. The infrastructure thus provided can then serve to represent and reason about more complex phenomena in NL brought by tense, aspect, and discourse, and would greatly benefit numerous applications. For example, with a more complete understanding of the temporal aspect of a discourse, a question-answering system could answer temporal or cause-effect questions, a text summarization system could provide a chronologically coherent account of events, and an intelligence analysis system such as that of Hauck et al. [2002] could derive conclusions based on a set of known cause-effect relations, which may be automatically learned by observing recurring chronological patterns, and so on.

Over the years the problems of temporal analysis in NL have been addressed with a spectrum of approaches, ranging from heavily inference-oriented to mostly NL-motivated ones. These include temporal logics [Gabby et al. 1994; Wooldridge et al. 1998; Artale et al., to appear]; formal accounts of calendars [Ohlbach and Gabbay 1998; Wijsen 2000; Ning et al. 2002]; a theory for representing actions and time [Allen 1984] and its continuation in the DAML Ontology of Time [Hobbs et al. 2002]; annotation of temporal expressions in newswire texts [Setzer 2001]; TIMEX2 annotation scheme [Ferro et al. 2001]; and the recent proposal of TimeML [Pustejovsky et al. 2002]. Taking a position somewhere in the middle of this spectrum is research on temporal databases [Snodgrass 1995] and their natural language interfaces [Androutsopoulos 2002]. Although all of these approaches have their own strengths in providing the infrastructure required for representing and reasoning about time in NL, few of them can at the same time deal with common phenomena such as granularity change and under-specification or is rigorous enough to facilitate sophisticated inferences.

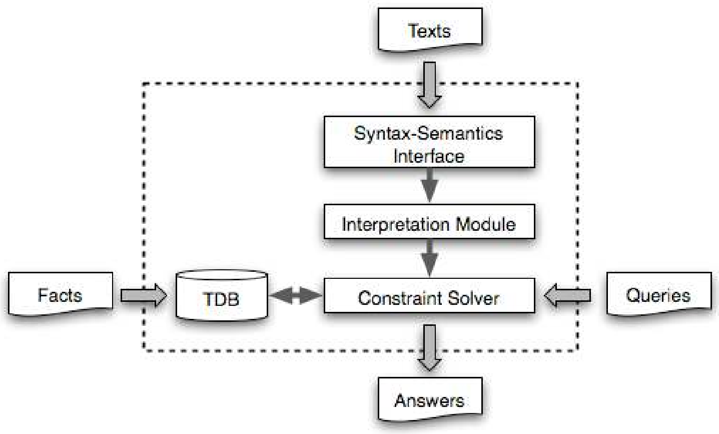


Fig. 1. A framework for resolution of temporal information in NL.

In this article we set out to address these requirements by proposing a constraint-based framework for resolution of temporal information in NL. In particular, we focus our effort in putting forward a practical way of modeling *temporal expressions*. Common temporal expressions in NL include noun phrases ("*Wednesday*"), prepositional phrases ("*in a week*"), adjectival phrases ("*current*"), adverbial phrases ("*recently*"), and subordinate clauses ("*…, when the market stabilized.*"). Within our framework these expressions denote *temporal objects* (timestamps of events), which are essentially sets of partial assignments to the variables of a calendar-constraint system. On a higher level, multiple expressions together describe a temporal scenario, which can be modeled as a temporal constraint-satisfaction problem (TCSP) [Dechter et al. 1991]. Figure 1 presents an overview of the framework: via the Syntax-semantics interface, temporal expressions are first extracted and translated into temporal objects encoded in a typed formal language, which is designed on the basis of our study of the Penn Treebank corpora [Marcus et al. 1994]. This representation can encode a wide variety of expressions and cope with phenomena such as under-specification and granularity change. The interpretation module is then called upon to rewrite certain temporal objects based on the contextual information, e.g., interpreting "*Wednesday*" as the nearest coming/past Wednesday, based on the corresponding verb tense if the expression is used in a non-habitual sense, or instantiating *temporal foci* throughout the discourse. The processed temporal objects together describe a TCSP, which is sent off to the constraint solver to find a set of complete and consistent assignments (solutions to the problem). During this stage, facts from a temporal database (TDB) can be retrieved to aid the solution process and the solutions can be stored in the TDB for future use. Various other NL applications can then access the framework by sending queries and obtaining answers.

Understanding the temporal aspect of a discourse obviously requires much more than just interpreting temporal expressions. Complicated problems such as modeling tense and aspect [Hwang and Schubert 1994], interpreting temporal prepositions and quantification [Pratt and Francez 2001], representing events [Steedman 1996], and dealing with discourse [Kamp and Reyle 1993] must also be tackled during the process. Addressing these problems, however, is not our main focus here. Our hope is that by providing a unified and principled treatment of temporal expressions (timestamps of events), solutions to these problems can be obtained more readily.

The rest of the article is organized as follows: Section 2 first motivates our approach by solving a historical puzzle; Section 3 then introduces the basic ideas behind the constraint-based temporal reasoning framework. Section 4 formalizes calendars as constraint systems. The details of our representation language for temporal expressions are then described in Section 5; in particular, it begins with our study of the Penn Treebank corpora that motivates the design decisions of our representation language and ends with derivation of meaning from example expressions. Section 6 gives a brief discussion on the related work. Finally, Section 7 concludes this article with a summary and plans for future work.

2. AN EXAMPLE: SOLVING A HISTORICAL PUZZLE

The possible meeting between Beethoven and Mozart has long been a fascinating puzzle: did they really meet? In this section we consider how a system utilizing our framework could at least confirm the possibility. To maintain our focus we assume that certain knowledge, such as two people can meet only if they are physically at the same location, is available to the system.

Consider the following short passage describing Beethoven's first trip to Vienna: the syntax-semantics interface identifies the temporal expressions (shown in boldface), and translates them into our temporal representation (shown in the parentheses):

> *"**At 14*** ($\mathtt{T1:\ \{T1'+|14_{year}|\}}$) *Beethoven was able to deputize for his teacher. **Three years later*** ($\mathtt{T2:\ \{\_+|3_{year}|\}}$)*, recognizing his talent, Prince Maxmilian Franz sent him to Vienna to further his education. He would soon return **within two weeks*** ($\mathtt{T3:\ \{\_+|(<2)_{week}|\}}$) *on the news that his mother was dying. She passed away **3 months later on July 17th 1787*** ($\mathtt{T4:\ \{\_+|3_{month}|,jul,17_{day},1787_{year}\}}$)*."*

The details of the representation language are given in Section 5; but suffice it to say that `T1'` to `T4` denote temporal objects; `{.}` represents a point (at certain granularity) in time; `|.|` denotes a temporal quantity; and '`_`' encodes an open temporal variable (temporal focus). The time `T1'` in particular denotes the birth of the composer. The Interpretation module then instantiates the temporal foci based on the context:

$\mathtt{T1:\ \{T1'+|14_{year}|\}}$	(deputizing-at-14)
$\mathtt{T2:\ \{T1+|3_{year}|\}}$	(off-to-Vienna)
$\mathtt{T3:\ \{T2+|(<2)_{week}|\}}$	(return)
$\mathtt{T4:\ \{T3+|3_{month}|,jul,17_{day},1787_{year}\}}$	(mother's-death)

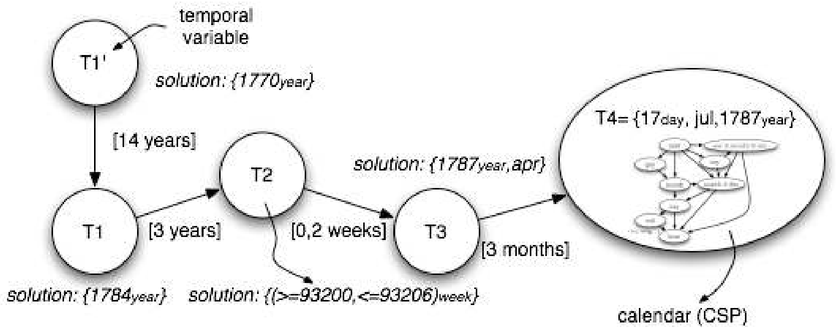


Fig. 2. TCSP depicting Beethoven's first trip to Vienna.

The entire set of temporal objects is then converted into temporal *variables* of a TCSP, shown in Figure 2[1]: each node in the figure represents one temporal variable, and the label on an edge represents the time difference between the adjacent variables, e.g., label `[0,2`$_{\texttt{weeks}}$`]` means the difference between `T2` and `T3` is from 0 to 2 weeks (`T2` is earlier). Inside each node is a calendar constraint system initialized by the expression which serves as the unary constraint for the node in solving the TCSP. Finally, the constraint solver takes over, decides that the particular TCSP is consistent, and gives an anchored time for each variable (also shown in Figure 2). At this point, the solution can be entered into a TDB for future use.

Suppose we are given another passage describing Mozart's activity in Vienna:

> "Mozart went to Munich to compose the opera **late in 1780** (`T5: {1780`$_{\texttt{year}}$`}`). **The next year** (`T6: {_+|1`$_{\texttt{year}}$`|}`), he was summoned from Munich to Vienna, where the Salzburg court was in residence on the accession of a new emperor. Mozart lived in Vienna for the rest of his life, until he died **in 1791** (`T7: {1791`$_{\texttt{year}}$`}`)."

Again, the interpretation module rewrites `T6` into `{T5+|1`$_{\texttt{year}}$`|}`, and a TCSP is formed. At this point we are interested in whether it is possible that Beethoven's stay in Vienna overlaps with Mozart's residence in the city, which requires tests of all possible interval-interval relations. In particular, the test that the interval from `T2` to `T3` is contained in that from `T6` to `T7` can be performed by inserting two hypothetical edges to relate the two TCSPs (shown in Figure 3). The constraint solver then confirms the consistency of the merged TCSP, and hence the facts support the possibility of a meeting.[2]

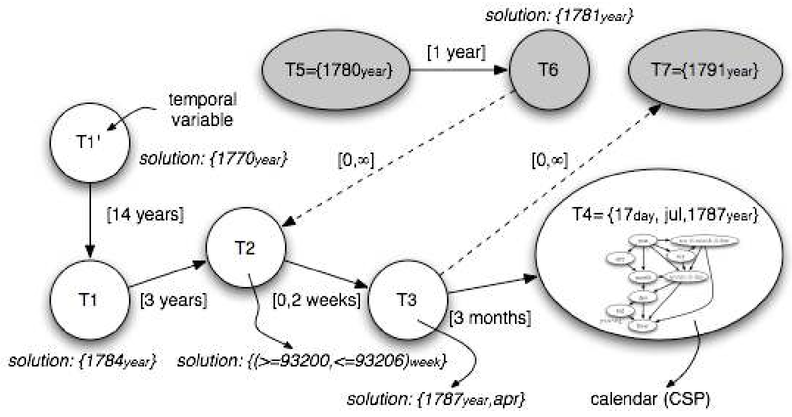


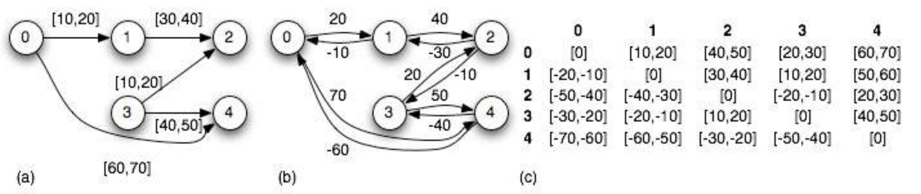
Fig. 3. Did Beethoven and Mozart meet in Vienna? Mozart's activity is shown in shaded graph.

[1] The week numbers are counted from January 1, 1 AD (the 1st week).
[2] Historically, whether the two met *anywhere* is still an unsolved puzzle, although Mozart did make comments on Beethoven's works.

3. CONSTRAINT-BASED TEMPORAL REASONING

Our framework essentially translates an NL discourse into a two-level constraint-satisfaction problem: at the higher level is a TCSP relating different temporal objects (variables), and within each temporal variable is a model of human calendars – a CSP describing constraints among different temporal units such as years and months. The overall TCSP is solved by conventional methods with the aid of the calendar model. In this section we briefly introduce the idea of constraint-based temporal reasoning.

A constraint-satisfaction problem (CSP) consists of a set of constraints over a set of *variables*, where each variable is associated with its *domain* of values. The goal is to find the assignments for variables such that none of the constraints is violated [Mackworth 1977]. Most CSPs involve only unary and binary constraints, and a constraint involving more than two variables can always be "binarized" [Bacchus et al. 1998]. CSPs can be solved by using a backtracking method to systematically search the solution space; but this is usually done with a combination of *constraint propagation* methods such as node-, arc-, and path-consistency methods, which can prune away a large portion of the search space. Other heuristics such as judiciously picking the right variable ordering are also helpful [Kumar 1992].



	0	1	2	3	4
0	[0]	[10,20]	[40,50]	[20,30]	[60,70]
1	[-20,-10]	[0]	[30,40]	[10,20]	[50,60]
2	[-50,-40]	[-40,-30]	[0]	[-20,-10]	[20,30]
3	[-30,-20]	[-20,-10]	[10,20]	[0]	[40,50]
4	[-70,-60]	[-60,-50]	[-30,-20]	[-50,-40]	[0]

Fig. 4. (a) an STP; (b) the flow network; (c) the minimal network.

A temporal constraint-satisfaction problem (TCSP), on the other hand, is a particular kind of CSP where domains are infinite (time) and constraints are allowed temporal relations [Dechter et al. 1991]. In particular, a binary constraint between variable X_i and X_j can be specified by an interval of time difference [a, b], which is interpreted as $\ldots a \leq X_j - X_i \leq b^3$. For a *simple TCSP* (STP) without disjunctive constraints, the problem can be converted into a flow network, and a minimal network can be found in polynomial time using an *all-pairs-shortest-path algorithm*. Sets of feasible assignments to the variables can then be obtained using a simple search on the edge labels of a minimal network. Figure 4 shows an example STP taken from Dechter et al. [1991]: in

(a) a node labeled *i* represents a variable X_i, and the edges are the time-difference constraints; (b) shows its converted flow network; and (c) is the minimal network. The variable X_0 in particular is added to represent the beginning of time (time 0), thus the minimal network indicates that, for example, the feasible range for X_3 is $20 \le X_3 \le 30$. A special solution to Figure 4 is $X_1 = 20, X_2 = 50, X_3 = 30, X_4 = 70$. A general TCSP can be decomposed into several STPs, and a backtracking search method can be used to find the solutions.

Adopting a TCSP framework for processing temporal information in NL has at least four advantages. (1) the minimal network enables us to find the minimal set of feasible times of a variable and the minimal set of relations between any pair of variables; the consistency of the network can be readily checked by detecting the existence of a negative cycle. Together they encompass a wide range of possible queries. (2) Both quantitative constraints and qualitative constraints (such as the 13 relations proposed in Allen [1984]) can be converted into such a "metric" network and solved uniformly [Meiri 1992]. (3) Obtaining a minimal network in STPs takes only polynomial time and assembling a feasible solution from it is backtrack-free; although solving general TCSPs is still NP-complete, efficient methods for exploiting special topologies of the constraint networks exist. (4) Finally, treating a temporal discourse as a TCSP fits naturally with our formal calendar modeled as a CSP.

A TCSP in this original form requires significant simplifications about real-world scenarios, where time is described not by simple real numbers but by using quantities of various sorts. Adding the missing concept of real calendars back to TCSPs has one main implication: i.e., the distance updating equation at the heart of the all-pair-shortest-path algorithm: $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$, where d_{ij} denotes the distance between variable X_i and X_j. In the partial TCSP shown in Figure 5, the shorter path from X_i and X_j is contingent on the distance between X_0 and X_i: if X_i is assigned to February of a non-leap year, "2 months" would be the shorter path from X_i to X_j; however, if X_i is assigned to March, the other path is the shorter one. This complication arises because in real calendars some metric units do not have constant size in terms of a base unit. We can, however, work around this problem by selectively updating distances affected by changes in assignments in certain situations.

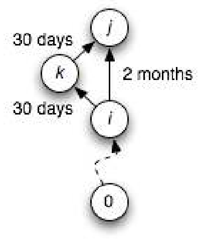


Fig. 5. A partial TCSP.

There are other complications as well. With the variety of metric units, the operator '+' and the relation '<' between distances must be defined on the basis of our

[3] When $a = b$, we simplify the notation to only $[a]$.

linguistic intuition, and the concept of calendars needs to be formalized (these are topics in the following sections).

4. MODELING CALENDARS [4]

Our goal in modeling calendars is similar to those of Ohlbach and Gabbay [1998]; Wijsen [2000]; and Ning et al. [2002], in that we want to find a principled way to formalize the structure of *any* human calendar. This would guarantee the necessary provisions for incorporating different kinds of calendars into our system. In our approach, however, the difference is that we not only view calendars as the ontology for representing temporal expressions, but also as *constraint systems* necessary for solving for the missing information in an under-specified expression. For example, given the expression "*February 29*", even without being told about the year, we can safely conclude that it cannot be the year 2003. The constraint-solving process can even be extended to convert expressions of a certain granularity ("*November 29, 2003*") to another ("*Sunday*"), and to answer questions about calendric arithmetics ("*what is the date of two days after February 27, 2000?*"). Ultimately, the calendar-constraint systems serve as unary constraints over the individual temporal variables in the higher-level TCSP.

4.1 Calendars as Constraint Systems

A calendar consists of a set of temporal *units*, and each unit can take on a set of possible *values* (e.g., unit `month` can be assigned to values `jan`, `feb`, ..., `dec`). This dichotomy naturally reflects the distinction between variables and domains in the setting of a CSP. Thus, a temporal *coordinate* expressed by an expression such as "*February 29*" essentially specifies an incomplete set of assignments to the units, and the omitted information can be derived (later) by solving the constraint system. In the rest of the article we shall denote a coordinate in a simple curly-bracketed format; e.g., $\{\texttt{feb}, 29_{\texttt{day}}\}$ represents the aforementioned date;[5] and we denote the set of units being assigned in a coordinate c as $D(c)$ (*domain* of c); e.g., $D(\{\texttt{feb}, 29_{\texttt{day}}\}) = \{\texttt{month}, \texttt{day}\}$. The additional requirement for a calendar, however, is that both values and units have to be ordered, so that two partial assignments in time can be compared (e.g., "*the first quarter of 1995*" is *earlier* than "*summer 1995*"). We therefore stipulate that values must be totally ordered, while the ordering among units can be partial. We denote the

[4] See Han and Kohlhase [2003] for a more formal treatment.
[5] Each element in the braces is a value assignment to a unit; units are shown in subscripts only when it is necessary for clarity.

latter ordering by the *measurement relation* $\rightarrow_M$; e.g., we could have `year` $\rightarrow_M$ `soy` (seasons)[6] and `year` $\rightarrow_M$ `qoy` (quarters of a year), and `qoy` and `soy` are not comparable. If u_1 is said to be *higher* than u_2 in $u_1 \rightarrow_M u_2$, then there could be more than one maximal/minimal unit in a calendar system. Note that although we give a notion of ordering to both units and values, it is somewhat different in nature for the two kinds of entities. As we shall see, the definitions essentially enable us to compare two coordinates lexicographically. Another point to note is that $u_1 \rightarrow_M u_2$ does not imply u_2 is *periodic* in u_1 – a concept we define in the next section.

To complete the definition of a calendar-constraint system, we must also specify the constraints among the assignments of the units. Naturally, each $u_1 \rightarrow_M u_2$ pair could be governed by a constraint, which we formalize by specifying a *cover function* from unit u_1 to u_2 as $C_{u_1,u_2} : V_{u_1} \rightarrow 2^{V_{u_2}}$, where V_{u_1} and V_{u_2} are values of u_1 and u_2.[7] For example, to represent the fact that the first *quarter* of a calendric *year* ranges from *January* to *March*, we write $C_{qoy,month}$(1) = {jan, feb, mar}. Using various conventional methods such as a combination of a constraint-propagation method (e.g., AC-3) and a backtracking search method [Kumar 1992], we can solve for a complete set of solutions based on a given coordinate c, which we call the *full extension* of c, denoted $\varepsilon(c)$. For example, assuming `year`, `qoy`, and `month` are the only units in our calendar, $\varepsilon(\{1_{qoy}, 1995_{year}\})$ then contains $\{jan, 1_{qoy}, 1995_{year}\}$, $\{feb, 1_{qoy}, 1995_{year}\}$, and $\{mar, 1_{qoy}, 1995_{year}\}$. The consistency of a coordinate c can then be defined straightforwardly: c is consistent *iff* $\varepsilon(c) \neq \varnothing$. Furthermore, the ordering of two coordinates c_1 and c_2 can also be defined based on the concept of full extensions: if $D(c_1) = D(c_2)$ and the units in the domain form a path from a maximal unit, they can be compared *lexicographically* (from the maximal unit to the lower ones). For example, $\{1_{qoy}, 1995_{year}\} < \{2_{qoy}, 1995_{year}\}$, since both coordinates are *anchored* on the same set of units that contains a maximal unit `year`. To compare two coordinates c_1 and c_2 that might be anchored on different units, e.g., $\{1_{qoy}, 1995_{year}\}$ and $\{apr, 1995_{year}\}$, we define $c_1 \leq c_2$ *iff* every coordinate c_1' in $\varepsilon(c_1)$ and every coordinate c_2' in $\varepsilon(c_2)$ such that $D(c_1') = D(c_2')$ (comparable), $c_1' \leq c_2'$ is true; $c_1 < c_2$ is true iff $c_1' < c_2'$ is true at least once. For example, we have $\{1_{qoy}, 1995_{year}\} < \{apr, 1995_{year}\}$.

[6] This particular modeling would interpret *Winter 2004* as two non-contiguous intervals (Jan. 1-Mar. 20 and Sept. 23-Dec. 31, 2004), which evaluates the assertion "*Winter 2004 is after Spring 2004*" to be false. An alternative would be to model seasons in a separate calendar with an alignment relation (later).
[7] *Notation*: 2^s is the power set of set s.

Multiple calendars can be coupled using a special *alignment* relation. This is useful in bringing together the calendar components of non-aligned cycles, such as *year*-based calendars and *week*-based calendars, as shown in Figure 6. The concept of full extensions can be naturally extended to coupled calendars, and granularity change for a coordinate involves shrinking/enlarging the domain of the coordinate (Section 5.3).

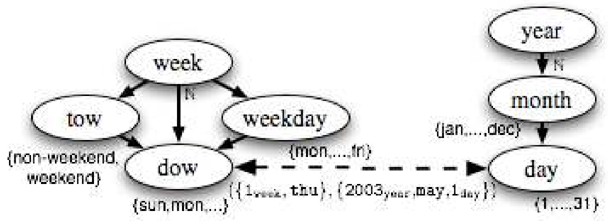


Fig. 6. Week-based calendar coupled with year-based calendar via an alignment relation (the measurement relation is shown by the solid arrows).

$$
\begin{aligned}
\mathcal{C}_{u_{\mathrm{top}},u\otimes_{\mathcal{C}_{\mathcal{M}}}u'} &\subseteq \mathcal{C}_{u,u\otimes_{\mathcal{C}_{\mathcal{M}}}u'} \circ \mathcal{C}_{u_{\mathrm{top}},u} \\
\mathcal{C}_{u,u\otimes_{\mathcal{C}_{\mathcal{M}}}u'}(v) &:= \{\langle v,w\rangle \mid \langle v,w\rangle \in V_{u\otimes_{\mathcal{C}_{\mathcal{M}}}u'}\} \\
\mathcal{C}_{u\otimes_{\mathcal{C}_{\mathcal{M}}}u',u'}(\langle v,w\rangle) &:= \{w\} \\
\mathcal{C}_{u\otimes_{\mathcal{C}_{\mathcal{M}}}u',u_{\mathrm{bot}}} &\subseteq \mathcal{C}_{u',u_{\mathrm{bot}}} \circ \mathcal{C}_{u\otimes_{\mathcal{C}_{\mathcal{M}}}u',u'}.
\end{aligned}
$$

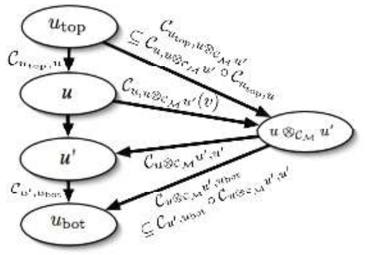


Fig. 7. Dependent unit product.

Note that the constraints defined above by means of specifying cover functions are all binary constraints. Non-binary constraints, however, are by no means rare in human calendars. For example, to determine how many days a particular month has, we also need to know the year, since February 29 only exists for leap years. To accommodate this, we adapt the well-known dual-graph approach for converting non-binary constraints into binary ones [Bacchus et al. 1998] and introduce *dependent unit products* (DUP) as complex units whose values are pairs of values of the existing units (Cartesian product). More formally, for units u_{top}, u, u', and u_{bot} such that $u_{\mathrm{top}} \rightarrow_M u \rightarrow_M u' \rightarrow_M u_{\mathrm{bot}}$, a DUP $u \otimes u'$ is defined as having values $\{\langle v,w\rangle | v \in V_u, w \in C_{u,u'}(v)\}$, where V_u is the domain of unit u. The topology shown in Figure 7 is also added into the calendar to ensure consistency. As an example, a partial model of a Gregorian calendar is shown in Figure 8, where

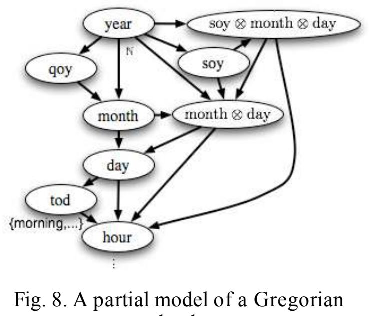


Fig. 8. A partial model of a Gregorian calendar.

DUP $\text{month} \otimes \text{day}$ models the ternary constraint among year, month, and day. We can then define $C_{\text{year,month}\otimes\text{day}}(i)$ to return $\{\langle i,\text{feb},1\rangle,\ldots,\langle i,\text{feb},29\rangle\}$ for leap years i, and return $\{\langle i,\text{feb},1\rangle,\ldots,\langle i,\text{feb},28\rangle\}$ for the others. Similarly, the quaternary constraint among year, season, month, and day is modeled by the DUP $\text{soy} \otimes (\text{month} \otimes \text{day})$.

4.2 Calendar Arithmetics and Periodicity Relation

A calendar system has to provide another important service, that is, doing arithmetics. For the most general cases, we need to define a predecessor/successor function for coordinate backward/forward shifting. The basic idea is to incrementally generate new instances of CSP and count the consistent instances to a certain number. For example, adding 1 day to $\{2003_{\text{year}}, \text{feb}, 28_{\text{day}}\}$ would first yield an inconsistent coordinate $\{2003_{\text{year}}, \text{feb}, 29_{\text{day}}\}$; the inconsistency then signals the successor function to invoke the carry propagation until the consistent $\{2003_{\text{year}}, \text{mar}, 1_{\text{day}}\}$ is generated, tested, and returned as the answer.

The general-case scenario requires linear time with respect to the quantity added.[8] In addition to the various caching/pre-computing tricks we could use to speed up the operation, in some cases we can take advantage of the *periodicity* relation between two units to achieve constant time complexity. We say that unit u_1 is periodic in unit u_2, denoted $u_1 \rightarrow_P u_2$, *iff* $u_2 \rightarrow_M u_1$ and every value of u_1 is in the cover of every value of u_2; e.g., $\text{month} \rightarrow_P \text{year}$, but month is not periodic in qoy. Because there is no "hole" in the cover C_{u_2,u_1} when $u_1 \rightarrow_P u_2$, adding l units u_1 to a coordinate c, written as $c \oplus_{u_1} l$, is reduced to a simple *modulo* operation: $c \oplus_{u_1} l := (c \oplus_{u_2} n)[u_1 \mapsto c(u_1) + m]$ where $l = n|V_u| + m$ and $c[u \mapsto v]$ is a coordinate identical to c except that value v is assigned to unit u.

5. REPRESENTING TEMPORAL EXPRESSIONS[9]

This section describes the meaning representation language for temporal expressions in our framework. The typed formal language consists of a set of temporal entities defined in a calendar constraint system (temporal units and values), and a set of operators and relations. The intensional meaning of an expression (see Section 5.1), be it a temporal point, a set of points, or a duration, is then encoded as a formula in this language. The omitted information in an under-specified expression can be automatically derived when

[8] In terms of number of consistency calls.
[9] Due to the limited space, we refer readers to Han [2003] for a complete description of the language.

necessary, thanks to the underlying calendar constraint system, and granularity change takes place in a transparent way via the built-in type coercion mechanism.

We start our discussion by first describing a corpus study based on the Penn Treebank corpora.

5.1 Temporal Expressions

As already mentioned in Section 1, we focus our study on non-verbal phrases, as they encode the majority of the complexity in temporal expressions. To better understand the variety of these expressions, we analyze the *Wall Street Journal* collection of the Penn Treebank corpora for the following reasons: (i) the Treebank annotation scheme already tagged the temporal expressions with *function tag* "-TMP", eliminating the need for us to prepare the tagged data; (ii) several highly accurate statistical parsers trained on the Penn Treebank corpora exist [Collins 1999; Charniak 1999], and these tools can be naturally incorporated within our syntax-semantics interface. Table I shows the statistics of the leading four categories of temporal expressions: the categories together represent 99.71% of all temporal expressions in the corpora, so we concentrate only on them. Also note that the percentage of the adverbial clauses used for temporal purposes (28.95%) is the highest among the four.

Table I. Temporal Expression Statistics in the Penn Treebank (*Wall Street Journal* Collection)

	ADVP-TMP	NP-TMP	PP-TMP	SBAR-TMP
Count	8303	5477	10573	2757
Percentage within all –TMP clauses	30.54%	20.14%	38.89%	10.14%
Percentage in the category	28.95%	1.35%	9.01%	8.65%

To better appreciate the complexity of temporal expressions, a list of representative expressions and their meanings is given below:

1. "*Sept. 9, 1987*": The simplest expressions are directly anchorable on a time line.
2. "*on Wednesday*": Most of the expressions are under-specified at least in two ways: they are not directly anchorable, and they do not commit themselves to either a time interval or a point.
3. "*Wednesday or Friday*": Logical disjunction can be expressed by using words such as "*or*".
4. "*4 o'clock*": Linguistic ambiguity occurs and is different from the logical disjunction; e.g., the expression could mean 4 in the morning or in the afternoon.
5. "*today*": Deictic expressions are anchorable only by implicitly making an external reference to a temporal focus (such as the utterance time).

6. "*last week*": Complex expressions usually involve shifting from a time with a specified offset; the offset not only specifies a duration to shift, but also implies *the granularity of interest*; e.g., no matter how fine the granularity of the current time, "*last week*" refers to a time down to the week granularity only.
7. "*last Wednesday*": In contrast to 6, above, shifting from a time that is not expressed in metric units behaves differently; e.g., the expression refers to the Wednesday in the last week.[10]
8. "*the second Sunday in May*": An ordinal expression is specified by an ordinal and a range.
9. "*an hour and 30 minutes*": Quantifier phrases with temporal units can denote a duration in time.
10. "*Tuesday and Thursday*": A set of temporal entities, not necessarily continuous, can be enumerated.
11. "*from now until 1995*": An interval, a specialized form of enumeration, can be specified by a starting and an ending point. Implicit granularity change works behind the scene to ensure both points are at the same granularity; e.g., the intended interval is to the last minute of the last day of 1994, provided the current time is at the minute granularity.
12. "*every week in May*": A recurrence expression is specified by a step size and a "pattern," possibly indicating a range. Again, "*every week*" and "*every Wednesday*" behave differently.
13. "*twice on Wednesday*": A rate expression can be specified by a frequency term and a range. This is also under-specified, since we do not known the specific time when each occurrence of the event takes place.

Based on this study, a desirable temporal representation for NL has to satisfy the following requirements: (i) it needs to provide an *intensional* representation for all of the expressions above, i.e., a single representation that can be evaluated in different contexts to give different interpretations; (ii) it must allow a natural representation and interpretation of under-specification and granularity change; (iii) it should facilitate a clean separation from discourse-level force (so discourse processing can act *on* the representation); and (iv) although it is not our focus here, the representation scheme should be readily integratable with a chosen event representation. These requirements are

[10] In some languages "*last Wednesday*" means the nearest Wednesday in the past (relative to the temporal focus) – this can be encoded in our representation using the *ordinal* `` `@' `` operator (Section 5.4).

satisfied in our constraint-based approach: since every temporal expression only serves as a timestamp (index) of an event, there is almost no restriction as to what kind of event representation we should adopt in our framework. The constraint-based approach also naturally accommodates the under-specified expressions. The representation language provides many operators, and allows referring to a discourse temporal variable ("temporal focus"), thus the intensionality and the separation from the discourse-level force are achieved. Finally, the granularity change is taken care of via the design of types and a mechanism of type coercion in the representation language.

5.2 Temporal Objects and Types

Every temporal expression is represented as a *temporal object* of the following three major types:

1. A *coordinate* (*C*) is a point in time. The simplest form is a conjunction of value constraints v_u where v is a value of unit u; e.g., "*Sep. 9, 1987*" is represented as $\texttt{\{}1987_{\texttt{year}}\texttt{,sep,}9_{\texttt{day}}\texttt{\}}$;
2. A *quantity* (*Q*) denotes a polarity-neutral duration of time. The simplest form is a conjunction of numeric constraints n_t, where n is a non-negative integer and t can be a unit or a set of values whose units form a path in the calendar; e.g., "*an hour and 30 minutes*" is represented as $\texttt{|}1_{\texttt{hour}}\texttt{,}30_{\texttt{min}}\texttt{|}$, and "*two Saturday nights*" is represented as $\texttt{|}2_{\texttt{(sat,night)}}\texttt{|}$.
3. An *enumeration* (*E*) is a set of coordinates. The simplest form is a list of coordinates; e.g., "*Tuesday and Thursday*" is represented as `[{tue},{thu}]`.

Connective "`,`" is used to conjoin two terms in both *C* and *Q* and to enumerate additional terms in *E*. For disjunctions, we distinguish between language ambiguity and genuine logical disjunction by using "`|`" for the former and "`;`" for the latter.

More complex objects can be built by using one of the infix operators and relations (Section 5.4). For example, using *backward fuzzy shifting operator* "`-`", the expression "*two weeks before Sept. 9 1987*" is represented as $\texttt{\{\{}1987_{\texttt{year}}\texttt{,sep,}9_{\texttt{day}}\texttt{\}-|}2_{\texttt{week}}\texttt{|\}}$; using a *point-interval* relation "`b`" (before), the expression "*sometime before Sept. 9, 1987*" can be encoded as $\texttt{\{b \{}1987_{\texttt{year}}\texttt{,sep,}9_{\texttt{day}}\texttt{\}\}}$. Also, *external temporal references* can be introduced in order to represent deictic expressions, e.g., "*yesterday*" is represented as $\texttt{\{\_-|}1_{\texttt{day}}\texttt{|\}}$: the underscore is an open variable representing the current *temporal focus*. Note that in order to keep the flexibility, we do not use the utterance time directly in this case, since it can be shifted within a discourse (e.g., reports from third parties in news articles). Instead, a discourse-level mechanism inside the interpretation module (Fig. 1) is responsible for instantiating the temporal focus. For under-specified expressions that are not deictic and not generic, another mechanism inside the interpretation module is also necessary for injecting a focus variable into the representation. For example, a non-generic expression "*Wednesday*" in "*the company will announce Wednesday*" is translated simply as {wed}, but will be rewritten into $\{|1_{\text{wed}}|@\{\text{bi_}\}\}$by the interpretation module because the corresponding verb is in the future tense (thus the formula means "*the nearest Wednesday in the future*", and bi is relation *after* [Allen 1984]).

As Examples 6 and 11 in Section 5.1 have shown, certain expressions can trigger implicit granularity change. We might suspect that cases of granularity change in NL are rare, since people usually do not mix expressions of different granularity together. However this is almost always not true for under-specified expressions, since their interpretations depend on an implicit temporal focus that speakers often make use of for economic reasons, and the temporal focus can be specified at any granularity possible. Even without referencing a temporal focus, interpreting expressions such as "*the summer following 1998*" still requires a treatment of granularity change (two other examples are shown at the end of Section 5.4). To model the phenomenon, we need to define the concept of *granularity*: our definition is via a function *g* recursively mapping a temporal object into a set of *minimal* temporal units, with respect to the measurement relation defined in the calendar. For example, $g(\{2003_{\text{year}},\text{nov}\}) = \min_M(\text{year}, \text{month}) = \{\text{month}\}$ (the granularity of "*November 2003*" is *month*), and $g(\{\{\_+|1_{\text{day}}|\},19_{\text{hour}}\}) = \min_M(g(\{\_+|1_{\text{day}}|\}), \text{hour}) = \{\text{hour}\}$ (the granularity of "*tomorrow at 7pm*" is *hour*). The granularity of an enumeration is simply defined as the set of minimal units over all granularities of its components.

To make the granularity change a transparent process, we decorate the major types with the granularity to become the type system for our language: we say that an object *o* is of type $T_{g(o)}$ when the major type of *o* is *T*. Under this definition, the type of $\{19_{\text{hour}},55_{\text{min}}\}$ is simply $C_{\{\text{min}\}}$. This gives us tremendous convenience: since every operator and relation in our representation is typed, *type coercion* will automatically kick in to bring the involved objects into the required types.

5.3 Type Coercion

Type coercion is possible within the same major type or among different major types. The former is realized by a granularity change function $\rightarrow_g$ over major type *C* and *E*, where *g* is the target granularity. Major type *Q* is excluded, since the use of a quantity never requires a granularity change for itself (instead, it drives the change of the others). Function $\rightarrow_g$ relies on a simple *projection* operation: $c|_\Theta$ gives a new coordinate of domain Θ from c. Operationally, this can be done by solving the underlying calendar CSP specified by the coordinate and then changing the domain. The function $\rightarrow_g$ over a coordinate can then be defined by specifying the set Θ:[11]

$$\Theta := D(c) \setminus \Phi \cup \Psi \cup g$$
$$\Phi := \{u' \mid u \rightarrow_M^* u', u \in g, u' \in D(c)\}$$
$$\Psi := \{u'' \mid u' \rightarrow_M^* u'' \rightarrow_M^* u, u \in g, u' \in D(c)\}$$

Intuitively, $\rightarrow_g (c)$ installs a new set of minimal units g in c; in particular, Φ represents granularity *promotion* (pruning) while Ψ represents *demotion* (adding information). A pictorial illustration of these processes is shown in Figure 9.

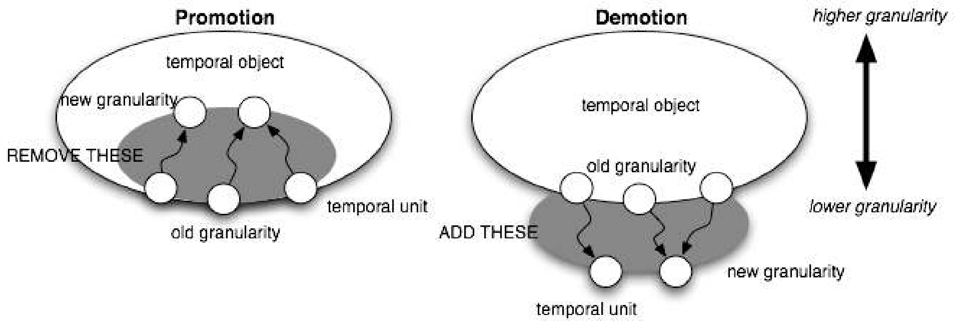


Fig. 9. Granularity change for a single temporal object: on each side, the biggest circle represents the set of assigned temporal units in a calendar constraint system (Fig. 8), and small circles are temporal units

Function $\rightarrow_g$ over E can be extended from the definition above. Examples of $\rightarrow_g$ are $\rightarrow_{\{\text{day}\}}$(`{may}`) = `{may, (>=1,<=31)`$_{\text{day}}$`}` and $\rightarrow_{\{\text{month}\}}$(`{31`$_{\text{day}}$`}`) = `{jan; mar; may; jul; aug; oct; dec}`.

For conversions among different major types, we only allow coercion from major type C to major type E via a re-interpretation function $C \rightarrow E_g$, which can be defined as (see Section 5.4 for the interval operator "`:`"):

1. $C \rightarrow E_g(c) := [\mathrm{i}\ [\min(\rightarrow_g (c)) : \max(\rightarrow_g (c))]]$ if $u' \rightarrow_M^* u$ for some $u' \in g(c)$ and $u \in g$;
2. $C \rightarrow E_g(c) := [\rightarrow_g (c)]$ otherwise.

An example is $C \rightarrow E_{\{\text{day}\}}$(`{may}`) = `[i [{may,1`$_{\text{day}}$`} : {may, 31`$_{\text{day}}$`}]]`.

5.4 Typed Operators and Relations

[11] $\rightarrow_M^*$ is the closure of the measurement relation.

Every operator and relation is typed in our representation, so that type coercion can transparently bring all of the involved objects into the required types beforehand. The type of an operator/relation, however, depends on the granularity of one of the involved objects: as shown in Example 6 of Section 5.1 ("*last week*"), the granularity of a shifting expression is determined by the quantity. To "factor out" the intended granularity from a quantity, we use a function $\mathrm{u}(q)$ to convert quantity q into its *pure-unit form* – i.e., every term in a pure-unit quantity must be of the form n_u, where n is a non-negative integer and u is a unit. The function $\mathrm{u}(q)$ is defined as follows:

1. change every n_{v_u} term into $n_{u'}$, where v_u is a value of unit u, and $u \to_P u'$ (u is periodic in u');
2. change every $n_{(v_1, v_2, ...)}$ term into n_u, where $v_1, v_2, ...$ is a set of values whose units $u_1, u_2, ...$ form a path in the calendar, and $\max(u_1, u_2, ...) \to_P u$.

For example, u(|2$_{\texttt{day}}$|) = |2$_{\texttt{day}}$|, u(|2$_{\texttt{morning}}$|) = |2$_{\texttt{day}}$|, and u(|2$_{\texttt{(sat,night)}}$|) = |2$_{\texttt{week}}$|.

Table II. List of the Typed Operators

Op	**Type**	**Meaning**	**Examples**
+/-	$E_{\to_{g(u(op_2))}} \times Q_{g(op_2)} \to C_{g(op_2)}$	forward/backward fuzzy shifting	{_+|1$_{\texttt{month}}$|} ("*next month*")
++/ --	$E_{\to_{g_m}} \times Q_{\to_{g_m}} \to C_{\to_{g_m}}$ $g_m = \min(g(op_1) \cup g(op_2))$	forward/backward exact shifting	{_++|1$_{\texttt{month}}$|} ("*exactly one month after*")
@	$Q_{g(op_1)} \times E_{\to_{g(op_1)}} \to C_{g(op_1)}$	ordinal	{|1$_{\texttt{wed}}$|@{bi _}} ("*the next nearest Wednesday*")
:	$C_{\to_{g_m}} \times C_{\to_{g_m}} \to E_{g_m}$ $C_{\to_{g(op_2)}} \times Q_{g(op_2)} \to E_{g(op_2)}$ $g_m = \min(g(op_1) \cup g(op_2))$	interval	[{may}:{jun}] [{may}:+|1$_{\texttt{month}}$|] [{may}:-|1$_{\texttt{month}}$|]
/	$E_{\to_{g(op_2)}} \times Q_{g(op_2)} \to C_{g(op_2)}$	arithmetic recurrence	[{may}:{aug}]/|1$_{\texttt{month}}$| ("*every month from May to August*")
^	$E_{\to_{g_m}} \times E_{\to_{g_m}} \to E_{g_m}$ $g_m = \min(g(op_1) \cup g(op_2))$	enumeration intersection	
\	$E_{\to_{g_m}} \times E_{\to_{g_m}} \to E_{g_m}$ $g_m = \min(g(op_1) \cup g(op_2))$	enumeration difference	

Table II gives a complete list of the typed operators: we use op_1 and op_2 to denote the two operands (from left to right), and $T_{\to_g}$ denotes an object of major type T converted to granularity g. The two sets of shifting operators move the ending coordinate of an enumeration with an offset specified by a quantity: the fuzzy version prunes away information below the target granularity, while the exact version preserves the granularity. The ordinal operator selects a coordinate from an enumeration; the interval operator forms an interval using either a pair of coordinates or a starting coordinate and a duration specified by a quantity; the arithmetic recurrence operator enumerates a set of coordinates within a range specified by a quantity, using a step-size specified by a second quantity[12]; and finally, the intersection and difference operators perform the respective set operation over enumerations.

When using a relation in the form of $r\ x_2$ in a hosting entity x_1, the intended meaning is $x_1\ r\ x_2$. There are two main categories for the relations: *value relations* and *object relations*. The former can only be used inside a value term, and it has the form r v, where v is a value. Value relations are <, >, and = with their usual semantics.[13] Examples of value relations are `{sun,(>19)`$_{\texttt{hour}}$`}` ("*Sunday after 7pm*") and `{_-|(<2)`$_{\texttt{day}}$`|}` ("*less than 2 days ago*"). For object relations there are two sets available: the *point-interval* relations can only be used inside a coordinate, while the *interval* relations can only be used inside an enumeration, as shown in Table III. For example, `{|1`$_{\texttt{summer}}$`|@{bi {1998`$_{\texttt{year}}$`}}}` ("*the summer following 1998*") and `{i [{jan,1998`$_{\texttt{year}}$`}:{mar,1998`$_{\texttt{year}}$`}]}` ("*sometime between January and March 1998*").

Table III. Object Relations

	Type	Relations	Remarks
Point-interval relations	$C_{\to_{\min}} \times E_{\to_{\min}}$ $\min = \min_M (g(op_1) \cup g(op_2))$	b, s, d, f, bi, i	Subset of the interval relations
Interval relations	$E_{\to_{\min}} \times E_{\to_{\min}}$ $\min = \min_M (g(op_1) \cup g(op_2))$	b, m, o, s, d, f, i, =, bi, mi, oi, si, di, fi, ii	From [Allen 1984]; i := (s;d;f)[14]

Before showing how an operator and type-coercion work together, we need to define another utility function. Function c(q), complementing u(q), produces the *implied constraints* from quantity q, and is defined as follows; initially we set $c(q) = \varnothing$:

1. For every n_v term, where n is a non-negative integer and v is a value, $\mathrm{c}(q) \leftarrow \mathrm{c}(q) \cup \{v\}$.
2. For every $n_{(v_1,v_2,...)}$ term, where $v_1, v_2, ...$ is a set of values whose units form a path in the calendar, $\mathrm{c}(q) \leftarrow \mathrm{c}(q) \cup \{v_1, v_2, ...\}$.

[12] *Pattern recurrence* is another construct for representing recurrence [Han 2003]; e.g., "*every Wednesday from 3pm to 5pm*" can be represented by `[{*wed,15`$_{\texttt{hour}}$`}:{17`$_{\texttt{hour}}$`}]`, and "*every 4 years since 1896*" is `[{*/4`$_{\texttt{year}}$`,bi {1896`$_{\texttt{year}}$`}}]`.
[13] = is usually dropped when there is no ambiguity.
[14] The relations are *before* (b), *meets* (m), *overlaps* (o), *starts* (s), *during* (d), *finishes* (f), 6 of their inverse relations, and *equal* (=). The relation *in* (i) and its inverse are added as a convenience.

To illustrate, we now define the semantics of fuzzy shifting +. Let *e* and *q* be an enumeration and a quantity, respectively, and assume $\mathrm{u}(q) = \left|n^1_{u_1}, \ldots, n^m_{u_m}\right|$, the operator is defined as $e + q \coloneqq \{\max(e[-1]) \oplus_{u_1} n^1 \ldots \oplus_{u_m} n^m, \mathrm{c}(q)\}$, where $e[-1]$ is the *last* coordinate in enumeration *e*, and function max(*c*) returns the "latest" possible coordinate from *c*. We first give a more involved derivation for expression "*two nights after the accident on March 28*":

$$\mathrm{u}(\left|2_{\text{night}}\right|) = \left|2_{\text{day}}\right|,\ g(\left|2_{\text{day}}\right|) = \{\text{day}\}$$

$$\begin{aligned}
\{\text{mar}, 28_{\text{day}}\} + \left|2_{\text{night}}\right| &= \{\max(C \rightarrow E_{\{\text{day}\}}(\{\text{mar}, 28_{\text{day}}\})[-1]) \oplus_{\text{day}} 2, \mathrm{c}(\left|2_{\text{night}}\right|)\} \\
&= \{\{\text{mar}, 28_{\text{day}}\} \oplus_{\text{day}} 2, \{\text{night}\}\} \\
&= \{\text{mar}, 30_{\text{day}}, \text{night}\}
\end{aligned}$$

The derivation for expression "*tomorrow,*" on the other hand, is simpler (assuming the temporal focus is $\{2003_{\text{year}}, \text{feb}, 28_{\text{day}}, 8_{\text{hour}}\}$):

$$\begin{aligned}
\_ + \left|1_{\text{day}}\right| &= \{2003_{\text{year}}, \text{feb}, 28_{\text{day}}, 8_{\text{hour}}\} + \left|1_{\text{day}}\right| \\
&= \{\max(C \rightarrow E_{\{\text{day}\}}(\{2003_{\text{year}}, \text{feb}, 28_{\text{day}}, 8_{\text{hour}}\})[-1]) \oplus_{\text{day}} 1\} \\
&= \{\{2003_{\text{year}}, \text{feb}, 28_{\text{day}}\} \oplus_{\text{day}} 1\} \\
&= \{2003_{\text{year}}, \text{mar}, 1_{\text{day}}\}
\end{aligned}$$

6. RELATED WORK

There are essentially three components in our framework: the calendar models, the representation language for temporal expressions, and the overall reasoning mechanism based on solving TCSPs. In this section we briefly discuss the work related to each of the components, and to contrast the previous approaches with our framework.

In modeling calendars, Ohlbach and Gabbay [1998] described a multi-modal logic for reasoning about calendric time. The many-sorted term language in their calendar logic formalizes calendars. The logic overall is decidable (via translation to propositional logic or via a tableaux decision procedure), but in the worst case is still exponential. One limitation in their representation is that every temporal unit has to be defined as a convex set using a base unit, and this makes defining units with "holes" difficult. In Wijsen [2000] a finite string-based representation is proposed to describe infinite granularities based on the idea that infinite patterns can be generated by fairly simple and recurrent patterns. The approach is much more efficient than the conventional methods where every temporal granule is defined by a smallest granule. Although it is not clear how under-specification can be dealt with, this is indeed an interesting idea that our constraint-based approach could incorporate to further restrict the formulation of cover functions. Another approach in formalizing calendars is that of Ning et al. [2001], where an algebra-based method is proposed. Every granularity is represented against a "bottom" granularity, and algorithms for granularity change are also presented. Although the representation is fairly flexible (e.g., capable of finding all business days in a given month), it is again not clear how under-specification is addressed. Overall, we believe our constraint-based calendars is a natural formalization that addresses particular needs in NL discourse, and fits well with the rest of the framework.

In representing temporal expressions, TimeML [Pustejovsky et al. 2002] is one of the most recent efforts in standardizing temporal annotations. The proposal is based on earlier work such as the temporal annotation scheme proposed in Setzer [2001] and TIMEX2 [Ferro et al. 2001]. The scope of TimeML extends beyond temporal expressions into the realm of eventuality. Due to its design as a markup language, the representation has a closer affinity to NL rather than to an inference formalism (such as first-order logic assumed in Hobbs et al. [2002]). Although the reasoning aspect of the representation was addressed in Setzer [2001] in the form of simple closure computation of temporal relations, it is not clear how more complex queries can be handled. The effect of granularity change is also not clear in the current proposal. However, the annotation scheme itself, due to its closer tie to surface texts, can be used as the first pass in the syntax-semantics interface of a temporal resolution framework such as ours. The more complex representation, suitable for more sophisticated reasoning, can then be obtained by translating from the annotations.

In terms of reasoning with time, many variants of temporal logics were proposed before [Gabbay et al. 1994]. Most of the approaches treat time as uniformly distributed clicks, and phenomena such as under-specification and granularity change are ignored. The meager set of operators also presents a problem in NL discourse; for example, a propositional logic based on a standard modal logic is proposed in Wooldridge [1998] for reasoning about knowledge and belief among multiple agents, and it has only two elementary operators: *Next-time* and *Until* – the former asserts the truth of a proposition at the next time click, and the latter asserts the truth until a certain time click. Some of the work on temporal reasoning, however, did have a closer tie with NL, such as Allen [1984] and its continuation in the DAML Ontology of Time [Hobbs et al. 2002]. In this line of work, an extensive axiomatization of time is proposed in first-order logic, taking into account the "real" time used in NL discourse. It also encompasses ideas such as granularity, convexity, and extensional collapse of time. In comparison, our framework addresses many of the same issues, albeit formulated in a constraint-solving formalism. We believe, in line with the authors of the SNARK system [Stickel et al. 2001], that instead of using a general-purpose inference engine based on first-order logic for temporal reasoning, time has special properties that justify the use of a special-purpose reasoner. But this does not imply that it is difficult to integrate external knowledge into our framework. For example, in the TCSP of Beethoven's visit to Vienna in Section 2, the Constraint Solver could add in additional constraints such as `T3` is before `T4`, based on the fact that "dying" is an event *culminating* in death[15].

The design of NL interfaces for temporal databases is another important line of work relating to temporal processing in NL. In Androutsopoulos [2002], a temporal intermediate representation language called TOP is proposed, and a parser driven by HPSG grammar is used to map time-related questions into the representation. The resulting TOP expressions are then translated into equivalent queries in TSQL2 [Snodgrass 1995] to query the database. The language TOP partitions NL verbs over the domain into a set of relevant aspectual categories, and provides a set of operators dealing with tense, aspect, interrogatives, and event identifications (using episode identifiers). However, the underlying modeling of time appears to be over-simplistic and fixed (Gergorian). It is not clear how phenomena such as under-specification and granularity change can be handled with customized calendars. The integration with TSQL2, nevertheless, is an important step in bringing the NL interface to a production system, and it is also a direction we would like to explore in the future.

7. CONCLUSIONS AND FUTURE WORK

In this article we have proposed a comprehensive framework for representing and reasoning about time in natural language. Our approach is capable of representing a wide variety of temporal scenarios described in natural language and translating them into temporal constraint-satisfaction problems (TCSPs). With respect to syntax, we have designed a representation for temporal expressions based on our study of the Penn Treebank corpora: the representation features sets of operators and relations to encode the intensional meaning of the expressions. Its type system together with the type coercion mechanism provides a transparent way to deal with phenomena such as granularity change and re-interpretation. In terms of semantics, real calendars are modeled as constraint systems and serve as unary constraints over individual temporal variables in the TCSP. Solving this two-level TCSP using conventional methods such as the well-known all-pairs-shortest-path algorithm combined with a backtracking search method, the

[15] However, in this case the added constraint has no additional effect on the final solutions. In fact, we could also add additional constraints to reflect the tense of the verbs in the narrative (past), but in this case they would have no effect on the solutions either.

resulting minimal representation of the scenarios can answer a wide range of time-related queries.

Currently, we are developing a prototype system that features two types of syntax-semantics interfaces for two different applications (in time we will report on the results). The first application is in the email domain (project RADAR[16] at CMU), where emails sent for scheduling meetings are identified and parsed into a machine-readable form to enable further scheduling negotiations among the participants and the agents (programs). For this application we used a parsing system based on semantic grammars [Gavaldà 2000] to extract and translate temporal expressions into our representation language, since a vast number of grammars were developed before for conversational speech in the travel-scheduling domain. The second application we are working on is answering time-related queries in newswire texts, where the open-domain nature requires a more shallow but effective parsing strategy. Our approach focuses on broad coverage and uses a simple event representation, recording only a predicate (verb) and its tense, the timestamp, and the event participants (any nominals involved in an SBAR/S clause). The system uses Charniak's maximum-entropy-inspired parser (trained on the Treebank corpora) [Charniak 2000] to provide CFG parse trees and uses named entities produced by BBN's identifinder [Bikel et al. 1999] to further decorate the trees. For each ADVP, NP, PP, and SBAR node in a parse (Section 5.1), we use a statistical classifier to decide whether it is a temporal clause, thus effectively recovering the TMP function tag that is missing from the output of the parser. The temporal nodes identified are then translated into our representation using a finite-state parser driven by hand-crafted grammars.

In the future we would like to investigate the effectiveness of various formal constructs in our framework on the basis of the empirical evidence. We would also like to extend the framework by considering other temporal phenomena such as tense and aspect, and investigating the effect of temporal focus-shifting in real-world scenarios such as emails and newswire texts. We are interested in using machine-learning techniques to automate grammar development required for building the syntax-semantics interface. Finally, we would like to investigate various methods to further enhance the efficiency of the underlying constraint solver, such as introducing a more restrictive constraint language for modeling calendars.

ACKNOWLEDGMENTS

[16] http://www.radar.cs.cmu.edu/external.asp

The earlier iteration of the work was done with Michael Kohlhase, who helped to shape many of the key ideas discussed in this article. The authors also wish to thank the reviewers for their extensive and helpful comments.

REFERENCES

ALLEN, J. F. 1984. Towards a general theory of action and time. *Artif. Intell. 23* (1984), 123–154.

ANDROUTSOPOULOS, I. 2002. *Exploring Time, Tense and Aspect in Natural Language Database Interfaces*. John Benjamins.

ARTALE, A. AND FRANCONI, E. To appear. Temporal description logics. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. M. F. Dov Gabbay and L.Vila, eds. MIT Press, Cambridge, MA.

BACCHUS, F. AND VAN BEEK, P. 1998. On the conversion between non-binary and binary constraint satisfaction problems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98*) and the *10th Conference on Innovative Applications of Artificial Intelligence* (IAAI-98), 311–318.

BIKEL, D. M., SCHWARTZ, R., AND WEISCHEDEL, R. M. 1999. An algorithm that learns what's in a name. *Machine Learning 34,*1-3 (1999), 211–231.

CHARNIAK, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000.*

COLLINS, M. 1999. Head-driven statistical models for natural language parsing. PhD dissertation, University of Pennsylvania.

CRISTIANINI, N. AND SHAWE-TAYLOR, J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

DECHTER, R., MEIRI, I., AND PEARL, J. 1991. Temporal constraint networks. *Artif. Intell. 49* (1991), 61–95.

FERRO, L., MANI, I., SUNDHEIM, B., AND WILSON, G. 2001. TIDES temporal annotation guidelines. MITRE Tech. Rep.

GABBAY, D., HODKINSON, I., AND REYNOLDS, M. 1994. *Temporal Logic: Mathematical Foundations and Computational Aspects*. Oxford University Press.

GAVALDA, M. 2000. SOUP: A parser for real-world spontaneous speech. In *Proceedings of the 6th International Workshop on Parsing Technologies* (IWPT-2000, Trento, Italy).

HAN, B. 2003. Time calculus for natural language - tagging guidelines. Unpublished draft, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA.

HAN, B. AND KOHLHASE, M. 2003. A time calculus for natural language. In *Proceedings of the 4th Workshop on Inference in Computational Semantics* (Nancy, France).

HAUCK, R. V., CHAU, M., AND CHEN, H. 2002. COPLINK: Arming law enforcement with new knowledge management technologies. In *Advances in Digital Government: Technology, Human Factors, and Policy*. W. McIver and A. Elmagarmid, eds. Kluwer Academic, Dordrecht, The Netherlands.

HOBBS, J. R., FERGUSON, G., ALLEN, J., HAYES, P., NILES, I., AND PEASE, A. 2002. *A DAML Ontology of Time*. http://www.cs.rochester.edu/~ferguson/daml/.

HWANG, C. H. AND SCHUBERT, L. K. 1994. Interpreting tense, aspect and time adverbials: A compositional, unified approach. In *Proceedings of the 1st International Conference on Temporal Logic* (ICTL 94, Bonn, Germany, July 11-14), 238-264.

KAMP, H. AND REYLE, U. 1993. *From Discourse to Logic*. Kluwer Academic, Dordrecht, The Netherlands.

KUMAR, V. 1992. Algorithms for constraint satisfaction problems: A survey. *AI Mag. 13,* 1 (1992), 32–44.

MACKWORTH, A. K. 1977. Consistency in networks of relations. *Artif. Intell. 8* (1977), 99–118.

MARCUS, M., KIM, G., MARCINKIEWICZ, M., MACINTYRE, R., BIES, A., FERGUSON, M., KATZ, K., AND SCHASBERGER, B. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop.*

MEIRI, I. 1992. Temporal reasoning: A constraint-based approach. PhD dissertation, UCLA.

NING, P., WANG, X. S., AND JAJODIA, S. 2002. An algebraic representation of calendars. *Ann. Math. Artif. Intell. 36,* 1-2 (2002), 5-38.

OHLBACH, H. AND GABBAY, D. 1998. Calendar logic. *J. Appl. Non-classical Logics 8,* 4 (1998), 291-324.

PRATT, I. AND FRANCEZ, N. 2001. Temporal prepositions and temporal generalized quantifiers. *Linguistics and Philosophy 24* (2001),187-222.

PUSTEJOVSKY, J., SAURÍ, R., SETZER, A., GAIZAUSKAS, R., AND INGRIA, B. 2002. *TimeML Annotation Guidelines*. http://www.cs.brandeis.edu/~jamesp/arda/time/documentation/AnnotationGuideline-v0.4.0.pdf.

SETZER, A. 2001. Temporal information in newswire articles: an annotation scheme and corpus study. Ph.D. dissertation, University of Sheffield.

SNODGRASS, R. T. ed. 1995. *The TSQL2 Temporal Query Language*. Kluwer Academic, Dordrecht, The Netherlands.

STEEDMAN, M. 1996. Temporality. In *Handbook of Logic and Language*. J. van Benthem and A. ter Meulen eds. Elsevier, London, 895-935.

STICKEL, M. E., WALDINGER, R. J., AND CHAUDHRIM, V. K. 2001. *A Guide to SNARK*. Tech. Rep. http://www.ai.sri.com/snark/tutorial/tutorial.html.

WIJSEN, J. 2000. A string-based model for infinite granularities. In *Proceedings of the AAAI-2000 Workshop on Spatial and Temporal Granularity*, 9-16.

WOOLDRIDGE, M., DIXON, C., AND FISHER, M. 1998. A tableau-based proof method for temporal logics of knowledge and belief. *J. Appl. Non-Classical Logics 8* (1988), 225–258.

Received December 2003; revised April 2004; accepted April 2004