A Time Calculus for Natural Language

Benjamin Han, Michael Kohlhase

Language Technologies Institute
Carnegie Mellon University
{benhdj|kohlhase}@cs.cmu.edu

Abstract

Automatic extraction and reasoning of temporal properties in a natural language discourse has not seen wide use in practical systems due to its demand of a rich and compositional, yet inference-friendly representation of time. We address the problem by proposing a time calculus within the framework of temporal constraint satisfaction problems (TCSP) [4], based on our study of the Penn Treebank corpora [11]. The formulation models temporal expressions as a two-level TCSP, thus enabling a system to answer a wide range of temporal-related queries.

1 Introduction

Time plays an active role in all facets of our lives, yet in many practical systems incorporating automatic analysis of natural language (NL), real time has long been a forgotten dimension. The problem is twofold: the nuances of time manifested in NL requires a rich yet compositional representation, and the need for advanced reasoning demands a more inference-friendly encoding of time. Over the years the problem has been addressed in a spectrum of works from heavily inference-oriented approaches to mostly NL-motivated methods. These include temporal logics [5,18,2], formal accounts of calendars [13,17,14], a theory for representing actions and time [1] and its continuation in the DAML Ontology of Time [9], annotation of temporal expressions in newswire texts [16] and the recent proposal of TimeML [15]. It is safe to say that the inference and algebra based approaches lack a principled account of how NL phenomena such as granularity and under-specification work, while the linguistically motivated methods need more formal rigor in order to facilitate a sophisticated inference mechanism.

In this paper we set out to address the problem by proposing a *time calculus* for NL, which will enable automatic extraction and reasoning of temporal properties of a discourse. Such systems could benefit numerous applications: question answering systems for texts could answer temporal or cause-effect questions; text summarization systems could provide a chronologically coherent account of events; an intelligence analysis system [8] could derive conclusions based on a set of known cause-effect relations, which may be automatically learned by observing recurring chronological patterns, etc.

In NL time is encoded via verb tense/aspect, and noun ("*Wednesday*"), prepositional ("*for a while*"), adjective ("*recent*"), and adverbial phrases ("*recently*"), etc. In this work we focus on representing temporal expressions (timex) involving calendric terms. At first sight timex may seem trivial to represent and to reason with, but our close study of the Wall Street Journal section in Penn Treebank corpora [11] (see [6] for details) has convinced us otherwise. Some examples are

(i) "*on Wednesday*": Most of the timex are under-specified at least in two ways: they are not directly anchorable on a timeline, and they do not commit themselves to either a time interval or a point.

(ii) "*last week*": More complex timex usually involve shifting from a temporal pivot (e.g., now) with a specified offset (e.g., a week); the offset not only specifies a duration but also implies the *granularity* of interest; e.g., regardless of the granularity of the current time, "*last week*" refers to a time at week granularity.

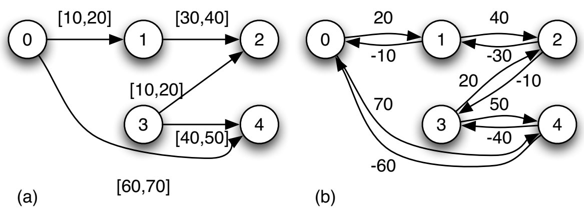
(iii) "*last Wednesday*": In contrast with the above, shifting with an offset that is not expressed in metric units behaves differently; e.g., the expression refers to the Wednesday in the last week.

(iv) "*the second Sunday in May*": An ordinal expression is specified by an ordinal and a range, possibly at different granularity.

Our calculus will need to capture all of the intricacies listed above (and beyond), and to enable a system to answer temporal-related queries such as when a specific event might happen. The design is based on the setting of solving a two-level *temporal constraint satisfaction problem* (TCSP) [4] (Sec. 2). Architecturally, real-world calendars are modeled as the lowest level CSP, and provide the basic vocabularies and services for meaning composition (Sec. 3). Timex are then captured using a typed formal language (Sec. 4), where the phenomena such as granularity conversion and re-interpretation are handled via type coercion. With the proposed operators and relations (Sec. 4.2), both qualitative and quantitative constraints among various temporal entities can then be expressed, and the resulting TCSP is solved using a modified all-pairs-shortest-path algorithm. As with the previous works on TCSP, this formulation will enable a system to answer a wide range of temporal-related queries.

2 Constraint-based Temporal Reasoning

Our view toward reasoning with time is in line with the framework formulated in [4], in which temporal assertions are represented as time-difference constraints, and the consistency of the resulting constraint networks is solved us-



	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]

(c)

Fig. 1. STP from Example. 2.1: (a) the constraint network (b) the distance graph (c) the minimal network

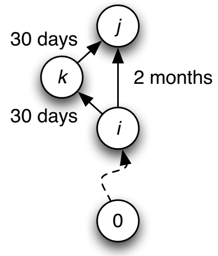
ing methods such as the conventional arc- and path-consistency methods [10]. For a *simple TCSP* (STP), the class of problems *without* disjunctive constraints, the constraint network can be converted to a simple flow network and its *consistency* derived by obtaining the *minimal network* using an all-pairs-shortest-path algorithm. In this section we will concentrate on solving an STP with only binary constraints, together with the necessary modification in order to make it work with NL.

A binary STP involves a set of temporal variables $\{X_1, \ldots, X_n\}$, each of which represents a time point and has a continuous domain. A constraint between X_i and X_j is represented by a single interval $[a, b]$, and is interpreted as $a \leq X_j - X_i \leq b$. The entire STP can be viewed as a constraint network where vertices are variables and directed edges are constraints. Each directed edge can be further doubled by breaking the constraint $a \leq X_j - X_i \leq b$ into $X_j - X_i \leq b$ and $X_i - X_j \leq -a$, and the result is a *distance graph*. Intuitively, in such a graph the composition of a sequence of constraints between two variables is translated to the path distance along the corresponding edges between the variables, and the conjunction of all constraint compositions between them has to be the tightest one, or equivalently, the shortest path. The following example illustrates the idea (adapted from [4]):

Example 2.1 (Example of binary STP) *It takes John 30-40 minutes to commute to work, and it takes Fred 20-30 minutes to do so. Today John left home between 7:10 and 7:20, and Fred arrived at work between 8:00 and 8:10. We also know that John arrived about 10-20 minutes after Fred left home.*

Fig. 1 shows the corresponding constraint network, the distance graph, and the minimal network after finding the shortest paths between all pairs of the variables. The network is consistent, since there is no *negative cycle* detected. Note that variable X_0 is added as an arbitrary reference point so that a unary constraint can be converted to a binary one.

Adopting an STP framework has at least three advantages. First, 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. Second, both quantitative constraints (e.g., the ones in Example 2.1) and qualitative constraints (e.g., Allen's 13 relations [1]) can be converted into such a "metric" network and solved uniformly [12]. And third, the all-pairs-shortest-path algorithm required for obtaining such networks has polynomial time complexity, and assembling a feasible solution (a set of consistent variable assignments) is backtrack-free; although a more general TCSP involving disjunctions is still NP-complete, there are efficient methods exploiting special topologies of the constraint networks.



It is obvious that the STP formulation has made many convenient simplifications when solving such problems in a NL setting. For one, the different granularities often encountered in real-life scenarios are all left as part of the "preprocessing". To illustrate one implication, consider the distance updating equation at the heart of the all-pairs-shortest-path algorithm: $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$, where d_{ij} is the distance between X_i and X_j, and a distance graph depicted to the right (only the necessary edges are shown). In a non-leap year, when X_i is assigned with a time February, the shorter path from X_i to X_j would be via the "*2 months*" edge. However if X_i is assigned with March, the shorter path would be the composition of the two "*30 days*" edges. This complication arises because some metric unit does not have constant size in terms of a base unit. We can solve this problem, however, by using this distance updating equation instead: $d_{ij} \leftarrow \mathrm{argmin}_{\{\delta \big|_{d_{ij}, d_{ik}+d_{kj})}\}}(d_{0i} + \delta)$.

Still there are other complications. With the variety of metric units, the operator '+' and the relation '<' between distances must be defined based on our linguistic intuition. Since in NL times are usually not given with respect to an arbitrary reference point ('X_0'), the distinction between a time "quantity" (e.g., 2 months) and a "coordinate" (e.g., February) must be made, and a bridge between the two has to be provided. Our formulation forgoes the arbitrary X_0 and uses directly a time coordinate at each variable (with a modified all-pairs-shortest-path algorithm), and provides necessary operators for basic time arithmetics. These are the topics of the following sections.

3 Calendars

From a representational point of view, a calendar provides the basic vocabularies for composing the meaning of a timex. A "upward" relation between two metric units, such as days "in" weeks, allows us to interpret timex "*last Wednesday*" as the Wednesday in the last week, and a "downward" relation, such as days are "measured by" hours, allows us to interpret "*early Wednesday*" as, say, the first one-third of the day, expressed in hours.

Within the framework of TCSP, a calendar, formulated as a CSP, serves as a set of unary constraints for the individual temporal variables, explicated linguistically via timex; e.g., "*February 29*" rules out all of the non-leap years from the domain of the temporal variable. They also play the supporting role for defining a TCSP, as hinted in Sec. 2. For instances, we need services from a calendar to compare the *ordering* of either two time coordinates, or two time quantities, and we need to do *arithmetics* such as adding a coordinate and a quantity, or two quantities. Finally, all of the services must take into account the possible differences in granularities, and engage *granularity conversion* if necessary.

In this section we formalize calendars as constraint systems. Although different in approach, our goal is similar to [13,17,14] in that we want to abstract away from an actual calendar, so that new calendars can be readily constructed using the formal tools provided.

3.1 Calendars as Constraint Systems

Temporal entities used in timex can be categorized into two kinds: *units* and *values*. This dichotomy has been motivated by our linguistic observations (see example ii and iii in Sec. 1), and it is a natural formulation within a constraint-solving framework. We will first define a *unit* as analogous to a variable in a CSP, with the twist that the domain of our "variable" [1] is ordered:

Definition 3.1 (Unit) *A* **unit** *u is a totally ordered set $(V, \leq)$. We call V the set of* **values** *of u and denote it by V_u.*

Within this view, a "point" in time can be specified by a complete set of unit assignments, i.e., a solution to the CSP at hand. However to be able to compare two points, we need to provide a "global" ordering among the values in addition to their "local" ordering defined within a unit. Thus we depart from a traditional CSP again by prescribing an ordering among the units. For example, comparing "*2002 May*" and "*2002 June*" follows a *lexicographic* order: first compare both years, and then months following a tie in years. This is embedded in the definition of a *unit structure*:

Definition 3.2 (Unit Structure) *Let $\mathcal{U}$ be a finite set of temporal units and $\mathcal{M}$ a relation on $\mathcal{U}$. For $u_1 \dots u_n \in \mathcal{U}$ we write $u_1 \to_{\mathcal{M}} u_n$ if $\langle u_1, u_n \rangle \in \mathcal{M}$, and $u_1 \to^*_{\mathcal{M}} u_n$ if a sequence $\langle u_1 \dots u_n \rangle$ exists such that $u_i \to_{\mathcal{M}} u_{i+1}$.*

We call $(\mathcal{U}, \mathcal{M})$ a **unit structure** *iff all of the following hold:*

(i) *(infinite maximals) V_u is isomorphic to $\mathbb{N}$ for all $\mathcal{M}$-maximal units u* [2] *and V_u is finite otherwise;*

(ii) *(no cycles) for any u and $u' \in \mathcal{U}$ s.t. $u \to^*_{\mathcal{M}} u'$, there cannot be $u' \to^*_{\mathcal{M}} u$;*

(iii) *(connected) for any $u \in \mathcal{U}$ there must exist $u' \in \mathcal{U}$ s.t. $u \to^*_{\mathcal{M}} u'$.*

For instance, $(\{\texttt{month}, \texttt{year}\}, \{\langle \texttt{year}, \texttt{month} \rangle\})$ is a unit structure with the units $\texttt{month} := (\{\texttt{jan}, \texttt{feb}, \dots, \texttt{dec}\}, \{\texttt{jan} \leq \dots \leq \texttt{dec}\})$ and $\texttt{year} := (\mathbb{N}, \leq_{\mathbb{N}})$.

Definition 3.3 (Unit Chains, Coordinate) *Let $\mu = (\mathcal{U}, \mathcal{M})$ be a unit structure, we call a path in μ a* **unit chain** *in μ. Since $\mathcal{U}$ is finite, we can always denote a unit chain as $\alpha := \langle u_1, \dots, u_n \rangle$. We call α* **grounded** *in μ, iff the*

[1] Note that this is different from the "variable" described in Sec. 2.

[2] $u \in \mathcal{U}$ is $\mathcal{M}$-maximal if the only $u' \to_{\mathcal{M}} u$ implies $u' = u$ for all $u' \in \mathcal{U}$.

maximal element in α is maximal in μ.

We call a mapping c from units to values a **coordinate**, *iff $c(u) \in V_u$. Given a set of units α in μ, we say that c* **is based on** *α iff c is defined on all units in α, i.e., $\mathfrak{D}(c) = \alpha$, and c is* **complete** *iff $\alpha = \mathcal{U}$. The projection of c on α, written as $c|_\alpha$, is a coordinate whose domain is α.*

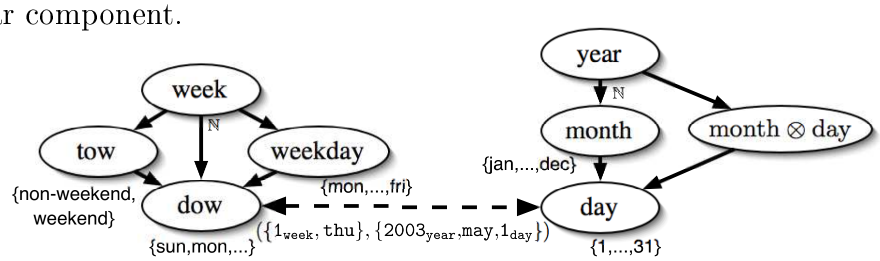
One problem remains for comparing two coordinates. For two coordinates based on the same grounded unit chain, comparison is straightforward. For two coordinates based on different chains, however, comparison requires "conversions" so that both can be based on the same chain. The required knowledge for such conversions is part of the definition of a *calendar component*:

Definition 3.4 (Calendar Component) *Let $\mu = (\mathcal{U}, \mathcal{M})$ be a unit structure and $u \rightarrow_{\mathcal{M}} u' \in \mathcal{U}$, then we call a total function $\mathcal{C}_{u,u'}: V_u \rightarrow 2^{V_{u'}}$ a* **cover function** *from u to u', iff $\mathcal{C}_{u,u'}(v) \subseteq \mathcal{C}_{u'',u'} \circ \mathcal{C}_{u,u''}(v) := \bigcup_{v'' \in \mathcal{C}_{u,u''}(v)} \mathcal{C}_{u'',u'}(v'')$ for each $v \in V_u$ and $u'' \in \mathcal{U}$ with $u \rightarrow u'' \rightarrow u'$. We call a complete set of cover functions $\mathcal{C}_{\mathcal{M}} := \{\mathcal{C}_{u,u'}: V_u \rightarrow 2^{V_{u'}} | u \rightarrow_{\mathcal{M}} u' \in \mathcal{U}\}$ a* **cover** *for μ.*

We call a tuple $k := (\mathcal{U}, \mathcal{M}, \mathcal{C}_{\mathcal{M}})$ a **calendar component**, *if $(\mathcal{U}, \mathcal{M})$ is a unit structure and $\mathcal{C}_{\mathcal{M}}$ is a cover for $(\mathcal{U}, \mathcal{M})$.*

Extending the unit structure above with a unit $\texttt{qoy} := (\{\texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}\}, \leq_{\mathbb{N}})$ for "quarter of year" and the relation $\mathcal{M}$ by $\texttt{year} \rightarrow \texttt{qoy} \rightarrow \texttt{month}$, the following cover functions give a calendar component: $\mathcal{C}_{\texttt{month}}(i_{\texttt{year}}) = \{\texttt{jan}, \texttt{feb}, \ldots, \texttt{dec}\}$, $\mathcal{C}_{\texttt{qoy}}(i_{\texttt{year}}) = \{\texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}\}$, and $\mathcal{C}_{\texttt{month}}(1_{\texttt{qoy}}) = \{\texttt{jan}, \texttt{feb}, \texttt{mar}\}, \ldots$.

It is obvious that calendar components are too simplistic to model real-world calendars. In [7] we have extended this notion to "full" calendars which are sets of calendar components that are "aligned" by a congruence relation. The following figure shows two aligned components: the week component and the year component.



The alignment relations essentially serve as "constraints by counting" which allows translation of coordinates among components. In this paper we will concentrate on the simpler version with calendar components, and refer readers to [7] for more details.

Given an incomplete coordinate of a calendar component, conventional constraint propagation methods can be used to derive complete and consistent coordinates; we then define the consistency of a coordinate accordingly:

Definition 3.5 (Coordinate Extensions and Consistency) *Let c be a coordinate of calendar component k. A complete coordinate $\bar{c}$ is an* **extension** *to c iff $c(u) = \bar{c}(u)$ and for all $u \rightarrow_{\mathcal{M}} u'$, $\bar{c}(u') \in \mathcal{C}_{u,u'}(\bar{c}(u))$ in k. We call the* *entire set of extensions to c, written as $\mathcal{E}_k(c)$, a* **full extension** *of c in k. A coordinate c in k is* **consistent** *iff $\mathcal{E}_k(c) \neq \emptyset$.*

Following the example calendar component above, $\{\mathtt{2003}_{\mathtt{year}}, \mathtt{jan}\}$ is consistent, but $c' = \{\mathtt{3}_{\mathtt{qoy}}, \mathtt{jan}\}$ is inconsistent, since $c'(\mathtt{month}) = \mathtt{jan} \notin \mathcal{C}_{\mathtt{month}}(c'(\mathtt{qoy})) = \{\mathtt{jul}, \mathtt{aug}, \mathtt{sep}\}$, hence there can be no extension to c'.

In principle we could maintain a full extension of a coordinate, thus a coordinate simply acts as an exposure to some of the calendar units, or a limited peek into the underlying calendar configurations.

We can now define the ordering between two coordinates:

Definition 3.6 (Coordinate Comparison) *Let c and c' be two coordinates of calendar component k. If both of them are based on the same grounded chain $\langle u_1, \ldots, u_n \rangle$, we say $c <_{\text{lex}} c'$ iff there exists $j = k+1$ such that $c(u_j) < c'(u_j)$ and $c(u_i) = c'(u_i)$ for all $i \leq k$, and $c =_{\text{lex}} c'$ iff $c(u_i) = c'(u_i)$ for all $i \leq n$.*

We say $c \leq_{\text{lex}} c'$ iff for all $\bar{c} \in \mathcal{E}_k(c)$ and $\bar{c'} \in \mathcal{E}_k(c')$ that are based on the same grounded chain, $\bar{c} \leq_{\text{lex}} \bar{c'}$; $c <_{\text{lex}} c'$ holds iff $c \leq_{\text{lex}} c'$ and $\bar{c} <_{\text{lex}} \bar{c'}$ holds at least once on the same grounded chain.

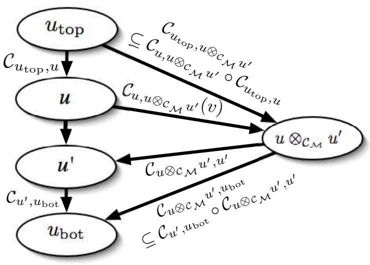
Following the example above, let $c_1 = \{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}\}$, $c_2 = \{\mathtt{1995}_{\mathtt{year}}, \mathtt{apr}\}$ and $c_3 = \{\mathtt{1995}_{\mathtt{year}}, \mathtt{feb}\}$. $c_1 <_{\text{lex}} c_2$ holds since all of the extensions in $\mathcal{E}(c_1) = \{\{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}, \mathtt{jan}\}, \{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}, \mathtt{feb}\}, \{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}, \mathtt{mar}\}\}$ are less then that in $\mathcal{E}(c_2) = \{\{\mathtt{1995}_{\mathtt{year}}, \mathtt{2}_{\mathtt{qoy}}, \mathtt{apr}\}\}$. However $c_1 \nleq_{\text{lex}} c_3$ since $\{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}, \mathtt{mar}\} \nleq_{\text{lex}} \{\mathtt{1995}_{\mathtt{year}}, \mathtt{1}_{\mathtt{qoy}}, \mathtt{feb}\}$; the reverse does not hold either.

3.2 Modeling Non-Binary Constraints

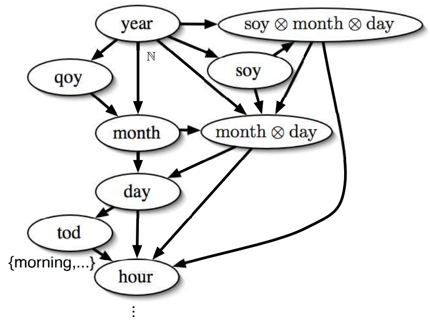
Not all temporal units in daily use can be easily modeled as in the examples above. In particular to account for `day` in a `month`, we have to consider not only the month but also the year since February 29 is only present in leap years. The constraint therefore is a ternary one and we need to convert it into a binary constraint. We will adapt from the well-known *dual-graph* approach for such conversions [3] and introduce complex units whose values are pairs of values of the existing units:

Definition 3.7 (Dependent Unit Products, DUP) *Let $k := (\mathcal{U}, \mathcal{M}, \mathcal{C}_{\mathcal{M}})$ be a calendar component and $u_{\text{top}} \rightarrow u \rightarrow u' \rightarrow u_{\text{bot}} \in \mathcal{U}$. We define the* **dependent unit product** *$u \otimes_{\mathcal{C}_{\mathcal{M}}} u' := (\{\langle v, w \rangle | v \in V_u \; w \in \mathcal{C}_{u,u'}(v)\}, \leq_{lex(V_u, V_{u'})})$, and add $u_{\text{top}} \rightarrow u \otimes_{\mathcal{C}_{\mathcal{M}}} u'$, $u \rightarrow u \otimes_{\mathcal{C}_{\mathcal{M}}} u'$, $u \otimes_{\mathcal{C}_{\mathcal{M}}} u' \rightarrow u'$ and $u \otimes_{\mathcal{C}_{\mathcal{M}}} u' \rightarrow u_{\text{bot}}$ to $\mathcal{M}$. The following cover functions are also added:*

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



It is straightforward to show that Def. 3.7 indeed constructs a calendar component based on Def. 3.4.



Consider a more complete calendar component (shown to the right) with two DUPs: unit `month` $\otimes$ `day` models the ternary constraint among `year`, `month` and `day`, and unit `soy` $\otimes$ (`month` $\otimes$ `day`) models the quaternary constraint among `year`, `soy` ("seasons of year"), `month` and `day`. Although alternatively we could construct DUPs out of the maximal unit `year`, we shun away from such constructs to avoid infinite explosion of value sets. Our modeling of days of months boils down to the definition of $\mathcal{C}_{\texttt{month}\otimes\texttt{day}}(i_{\texttt{year}})$: it returns $\{\langle i, \texttt{feb}, 1\rangle \cdots \langle i, \texttt{feb}, 29\rangle\}$ for leap year i ($i \equiv 0 \bmod 4$, $i \not\equiv 0 \bmod 100$ and $i \equiv 0 \bmod 400$), and $\{\langle i, \texttt{feb}, 1\rangle \cdots \langle i, \texttt{feb}, 28\rangle\}$ for the others. A coordinate $\{2003_{\texttt{year}}, \texttt{feb}, 29_{\texttt{day}}\}$ is then inconsistent since no extension to it can exist. We skip the description of how to model the seasons for similarity.

3.3 Periodicity and Coordinate Arithmetics

Another interesting feature of a calendar is the periodicity of units in one another:

Definition 3.8 *Let $(\mathcal{U}, \mathcal{M}, \mathcal{C}_{\mathcal{M}})$ be a calendar component and $u' \to_{\mathcal{M}} u \in \mathcal{U}$, we say that u* **is periodic in** *u', written as $u \rightarrowtail u'$, iff for each $v \in V_u$, $v \in \bigcap_{v' \in V_{u'}} \mathcal{C}_{u',u}(v')$.*

Continuing the examples, we have `month` $\rightarrowtail$ `year`, `qoy` $\rightarrowtail$ `year`, etc.

Periodicity allows us to correctly model timex such as "*last/every v_u*" where v_u is a value of unit u: the interpretation is "*v in last/every u'*" with $u \rightarrowtail u'$ (see examples in Sec. 1). Another advantage for discovering periodicity is that it speeds up coordinate arithmetics. For the general case, we need a notion of successor and predecessor, which is more involved to define.

Definition 3.9 (Coordinate Arithmetics, Special Case) *Let $k = (\mathcal{U}, \mathcal{M}, \mathcal{C}_{\mathcal{M}})$ be a calendar component, then we define a function $\oplus$ as*

$$c \oplus_u l := (c \oplus_{u'} n)[u \mapsto c(u) + m]$$

where $l = n|V_u| + m$ such that $m \leq |V_u|$, and $u \rightarrowtail u'$. $c[u \mapsto v]$ is the coordinate that maps u to $v \in V_u$ and agrees with c for all units $u' \neq u \in \mathfrak{D}(c)$.

4 Representing Temporal Expressions

By using the "vocabularies" provided by a calendar, which we have formalized in Sec. 3, fairly complex timex can be constructed to represent a temporal point, a set of points, or a duration. The representation of these expressions, called *temporal objects*, are usually under-specified in many ways (see examples in Sec. 1), and they often embed implicit granularity conversion when certain operation is engaged. The under-specification aspect also calls for a representation which factors out the discourse dependency. In this section we will describe a formal language for representing timex. The language includes a set of operators and relations to construct temporal objects, and a granularity-refined type system to account for phenomena such as granularity conversion and re-interpretation. Due to the limited space, we refer readers to [6] for a complete description of the language.

4.1 Temporal Objects and Types

A *temporal object* can be of one of the three *major types*: *coordinate* ($\mathcal{C}$), *quantity* ($\mathcal{Q}$) and *enumeration* ($\mathcal{E}$):

(i) A **coordinate** is a point in time as defined in Def. 3.3. The simplest form is a conjunction of value constraints v_u where v is a value of unit u; e.g., "*Sept. 9, 1987*" is represented as $\{\mathtt{1987}_{\mathtt{year}},\mathtt{sep},\mathtt{9}_{\mathtt{day}}\}$;

(ii) A **quantity** denotes a polarity-neutral duration of time. The simplest form is a conjunction of numeric constraints n_t, where $n \in \mathbb{N}$, and t can be a unit or a set of values from a unit chain; e.g., "*an hour and 30 minutes*" is represented as $|\mathtt{1}_{\mathtt{hour}},\ \mathtt{30}_{\mathtt{min}}|$;

(iii) An **enumeration** is a set of coordinates. The simplest form is a list of coordinates; e.g., "Tuesday and Thursday" is represented as $[\{\mathtt{tue}\},\ \{\mathtt{thu}\}]$.

We use connective ',' to conjoin two terms in both $\mathcal{C}$ and $\mathcal{Q}$ and enumerate additional terms in $\mathcal{E}$. 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 constructed by using one of the infix operators (later) and relations. For the relations there are two kinds available. A *value relation rel* is used as *rel* v_u where v is a value of unit u and *rel* is one of $<$, $>$ and $=$ relations with their usual semantics[3]. An *object relation rel* is used in the form of *rel* o' in a hosting object o, with the intended reading o *rel* o'. We use the same set of value relations for $\mathcal{Q}$, but use Allen's 13 interval-based relations [1] for $\mathcal{C}$ and $\mathcal{E}$. Some examples of complex objects are $\{\{\mathtt{1987}_{\mathtt{year}},\mathtt{sep},\mathtt{9}_{\mathtt{day}}\}-|\mathtt{2}_{\mathtt{week}}|\}$ for "*two weeks from Sept. 9, 1987*" and $\{\mathtt{b}\ \{\mathtt{1987}_{\mathtt{year}},\mathtt{sep},\mathtt{9}_{\mathtt{day}}\}\}$ for "*sometime before Sept. 9, 1987*".

External temporal references are introduced in objects for deictic expressions; e.g., "*yesterday*" is $\{\_-|\mathtt{1}_{\mathtt{day}}|\}$, where the underscore represents the current *temporal focus*. A discourse-level mechanism for managing the temporal focus is therefore necessary but orthogonal to our discussions. Some of the under-specified objects ($\{\mathtt{wed}\}$ for example) will need to reference the temporal focus as well, however some will not when they are used as *generics*. Again

[3] For equality relation = the relation symbol is always dropped.

an independent mechanism must be provided for such decisions.

Before introducing the granularity-refined type system, we first define the *granularity* of a temporal object as a set of minimal units among all of the relevant units in the object.

Definition 4.1 (Granularity Function) *Let $(\mathcal{U}, \mathcal{M})$ be a unit structure. A total* granularity function $g\colon \mathcal{O} \to 2^{\mathcal{U}}$ *is a mapping from an object to a set of units:*

(i) *If $o \in \mathcal{C}$ or $\mathcal{Q}$: $g(o) := \min_{\mathcal{M}}(\bigcup_{term\ in\ o} \hat{g}(term))$ where $\min_{\mathcal{M}}$ returns a set of minimal units w.r.t. relation $\mathcal{M}$;*

(ii) *If $o \in \mathcal{E}$: assuming o to be* homogeneous, *meaning all of its terms must have the same granularity*[4], *then $g(o) := \hat{g}(term)$ where term is in o.*

The term-level granularity function $\hat{g}(term)$ is defined as:

(i) *if term is of the form $o_1\, op\, o_2$ then $\hat{g}(term)$ is the granularity of the resulting object;*

(ii) *if term is of the form rel v_u where rel is a value relation then $\hat{g}(term) := \{u\}$.*

We now construct a granularity-refined type system by decorating the major types with the granularity of an object:

Definition 4.2 (Temporal Types) *An object o is said to be of type $T_{g(o)}$ if o is of major type T, where T is one of $\mathcal{C}$, $\mathcal{Q}$ or $\mathcal{E}$.*

For example, the type of $\{1_{\texttt{hour}}, 30_{\texttt{min}}\}$ is $\mathcal{C}_{\texttt{min}}$.

The design of a granularity-refined type system enables us to define various typed operators and relations, and coercion kicks in to bring the involved objects into the required types. This not only can be used to model phenomena such as granularity conversion shown in the examples in Sec. 1 and certain re-interpretation, but can also simplify the formulation of various operators and relations.

Type coercion is possible within the same or among different major types. The former is realized by a *granularity conversion function* $\to_g$ over $\mathcal{C}$ and $\mathcal{E}$ with g being the target granularity. $\mathcal{Q}$ is excluded since the use of a quantity never requires a granularity conversion of itself (instead it drives the conversion of the others). For conversions among different major types we only allow coercion from $\mathcal{C}$ to $\mathcal{E}$ which we define as a *re-interpretation function* $\mathcal{C} \to \mathcal{E}_g$. The reasons for excluding the other possibilities are (i) coercions between $\mathcal{Q}$ and the other major types do not seem necessary for our purposes; (ii) for converting $\mathcal{E}$ to $\mathcal{C}$ it is usually not possible to find a uniform treatment[5], and it is often not necessary as we can canonicalize on $\mathcal{E}$ and use $\mathcal{C} \to \mathcal{E}_g$ instead.

---

[4] It might be worthwhile to loosen this definition to only consider the minimal granularity for better efficiency.

[5] Therefore it is better left to be defined explicitly whenever necessary.

We now give a definition for $\rightarrow_g(o)$ for object o, which basically installs a new set of minimal units into o, and makes sure that all of the "intermediate" units are also included.

Definition 4.3 (Granularity Conversion Function) *Let k be a calendar component, for coordinate c and the target granularity g (a set of incomparable units), the* **granularity conversion function** *$\rightarrow_g(c)$ returns all $\bar{c}|_\Theta$ where $\bar{c} \in \mathcal{E}_k(c)$ and Θ is defined as:*

$$\Theta := \mathfrak{D}(c) \setminus \Phi \cup \Psi \cup g$$
$$\Phi := \{u' | u \rightarrow^*_{\mathcal{M}} u',\ u \in g,\ u' \in g(c)\}$$
$$\Psi := \{u'' | u' \rightarrow^*_{\mathcal{M}} u'' \rightarrow^*_{\mathcal{M}} u,\ u \in g,\ u' \in g(c)\}$$

For enumeration e, $\rightarrow_g(e)$ is distributed onto each term: for $\mathcal{C}$ terms the definition above applies, for $\mathcal{E}$ terms or terms involving operators, $\rightarrow_g$ applies recursively on any non-$\mathcal{Q}$ object.

Two examples are $\rightarrow_{\texttt{day}}(\{\texttt{may}\}) = \{\texttt{may}, (>=1, <=31)_{\texttt{day}}\}$, and $\rightarrow_{\texttt{month}}(\{31_{\texttt{day}}\}) = \{(\texttt{jan}; \texttt{mar}; \texttt{may}; \texttt{jul}; \texttt{aug}; \texttt{oct}; \texttt{dec})\}$.

The re-interpretation $\mathcal{C} \rightarrow \mathcal{E}_g(c)$ for a coordinate c is basically an expansion of c into an enumeration of the designated granularity. We use utility function $\min(c)$ and $\max(c)$ below: they return the minimal/maximal possible coordinate from c; e.g., $\min(\{\texttt{may}, (>=1, <=31)_{\texttt{day}}\}) = \{\texttt{may}, 1_{\texttt{day}}\})$.

Definition 4.4 (Re-interpretation Function) *Let c be a coordinate and g be the granularity, the* **re-interpretation function** *$\mathcal{C} \rightarrow \mathcal{E}_g(c)$ is defined as:*

(i) $\mathcal{C} \rightarrow \mathcal{E}_g(c) := [\texttt{i}\ [\min(\rightarrow_g(c)) : \max(\rightarrow_g(c))]]$ *if there exists $u \in g$ and $u' \in g(c)$ s.t. $u' \rightarrow^*_{\mathcal{M}} u$* [6] *; and*

(ii) $\mathcal{C} \rightarrow \mathcal{E}_g(c) = [\rightarrow_g(c)]$ *otherwise.*

An example is $\mathcal{C} \rightarrow \mathcal{E}_{\texttt{day}}(\{\texttt{may}\}) = [\{\texttt{may}, 1_{\texttt{day}}\} : \{\texttt{may}, 31_{\texttt{day}}\}]$.

4.2 Typed Operators and Relations

The operators and relations are typed in our calculus so that coercion can enforce a type uniformity among the involved objects. To assign types we use a utility function $\mathrm{u}(q)$ to convert a quantity q into a *pure-unit quantity* - a quantity with all of its terms being in the form n_u where $n \in \mathbb{N}$ and u is a unit. Function $\mathrm{u}(q)$ is defined as:

(i) changing n_{v_u} terms into $n_{u'}$, where $n \in \mathbb{N}$, v_u is a value of unit u, and $u \rightarrowtail u'$; and

(ii) changing $n_{(v_1, v_2, \ldots)}$ terms where $(v_1, v_2, \ldots)$ are values of a unit chain $\langle u_1, u_2, \ldots \rangle$, into n_u where $u_1 \rightarrowtail u$.

---

[6] Relation `i` (in) is an abbreviation of the disjunction of `s`, `f` and `d` in Allen's interval algebra [1].

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

The table below shows a complete list of the typed operators, where op_1 and op_2 are (from left to right) the two operands, 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, operator @ selects a coordinate from an enumeration, operator : forms an interval using either a pair of coordinates or a starting coordinate and a duration specified by a quantity, operator / enumerates a set of coordinates within a range specified by a coordinate, using a step-size specified by a quantity [7], and operator $\wedge$ and $\backslash$ are essentially set intersection/difference operators.

	Type	Meaning	Example
$+/-$	$\mathcal{E}_{\to_{g(\mathrm{u}(op_2))}} \times \mathcal{Q}_{g(op_2)} \to \mathcal{C}_{\to_{g(op_2)}}$	forward/backward fuzzy shifting	{_+|$1_{\texttt{month}}$|} (“*next month*”)
$++/--$	$\mathcal{E}_{\to_{g(op_2)}} \times \mathcal{Q}_{g(op_2)} \to \mathcal{C}_{\to_{g(op_2)}}$	forward/backward exact shifting	{_++|$1_{\texttt{month}}$|} (“*exactly one month after*”)
@	$\mathcal{Q}_{g(op_1)} \times \mathcal{E}_{\to_{g(op_1)}} \to \mathcal{C}_{\to_{g(op_1)}}$	ordinal	[|$2_{\texttt{sun}}$|@{may}] (“*the second Sunday in May*”) [|$1_{\texttt{wed}}$|@{**bi** _}] (“*the next nearest Wednesday*”)
:	$\mathcal{C}_{\to_{\min}} \times \mathcal{C}_{\to_{\min}} \to \mathcal{E}_{\to_{\min}}$ $\mathcal{C}_{\to_{\min}} \times \mathcal{Q}_{\to_{\min}} \to \mathcal{E}_{\to_{\min}}$ $\min = \min_{\mathcal{M}}(g(op_1) \cup g(op_2))$	interval	[{may}:{jun}] [{may}:|$1_{\texttt{month}}$|]
/	$\mathcal{E}_{\to_{g(op_2)}} \times \mathcal{Q}_{g(op_2)} \to \mathcal{C}_{\to_{g(op_2)}}$	arithmetic recurrence	[[{may}:{aug}]/|$1_{\texttt{month}}$|]
$\wedge$	$\mathcal{E}_{\to_{\min}} \times \mathcal{E}_{\to_{\min}} \to \mathcal{E}_{\to_{\min}}$ $\min = \min_{\mathcal{M}}(g(op_1) \cup g(op_2))$	enumeration intersection	
$\backslash$	$\mathcal{E}_{\to_{\min}} \times \mathcal{E}_{\to_{\min}} \to \mathcal{E}_{\to_{\min}}$ $\min = \min_{\mathcal{M}}(g(op_1) \cup g(op_2))$	enumeration difference	[{_+|$0_{\texttt{year}}$|}\\ [{|$1_{\texttt{month}}$|@{_+|$0_{\texttt{year}}$|}}:_]] (“*the rest of the year*”)

We also assign types to object relational terms: for $\mathcal{Q}$ the relations are all assigned type $\mathcal{Q}_{\to_{\min}} \times \mathcal{Q}_{\to_{\min}}$, where min is the set of minimal units from the granularities of the two related objects; for $\mathcal{C}$ and $\mathcal{E}$ we assign type $\mathcal{E}_{\to_{\min}} \times \mathcal{E}_{\to_{\min}}$ to all of the possible relations.

We are now ready to give the semantics of fuzzy shifting $+$. Let $c = e+q$ where c, e and q are a $\mathcal{C}$ $\mathcal{E}$ and $\mathcal{Q}$ respectively. The result, assuming $\mathrm{u}(q) = |n^1_{u_1}, \ldots, n^m_{u_m}|$, is defined as $c := \{\max(e[-1]) \oplus_{u_1} n^1 \ldots \oplus_{u_m} n^m, c_q\}$ where $e[-1]$ is the last coordinate in e under $\leq_{\mathrm{lex}}$ (defined in Def. 3.6), and c_q is the implied

[7] A simpler *pattern recurrence* is also included in [6]; e.g., [[{*wed, $15_{\texttt{hour}}$}:{$17_{\texttt{hour}}$}]] for “*every Wednesday from 3pm to 5pm*”, and [{*/$4_{\texttt{year}}$, **bi** {$1896_{\texttt{year}}$}}] for “*every 4 years since 1896*”.

constraints of q, which is defined as follows: for n_u terms there is no implied constraint; for n_v the constraint is $\{\mathtt{v}\}$; and for $n_{(v_1,v_2,\ldots)}$ the constraint is $\{\mathtt{v}_1,\mathtt{v}_2,\ldots\}$. An example is given below:

$$\begin{aligned}\{\mathtt{may}\}+|2_{\mathtt{morning}}| &= \{\max([\{\mathtt{may},\ 1_{\mathtt{day}}\}{:}\{\mathtt{may},\ 31_{\mathtt{day}}\}][-1]) \oplus_{\mathtt{day}} 2,\ \{\mathtt{morning}\}\} \\ &= \{\{\mathtt{may},\ 31_{\mathtt{day}}\} \oplus_{\mathtt{day}} 2,\ \{\mathtt{morning}\}\} \\ &= \{\mathtt{jun},\ 2_{\mathtt{day}},\ \mathtt{morning}\}.\end{aligned}$$

Due to space restrictions we refer readers to [7,6] for the definitions of the other operators.

5 Summary and Future Works

In this paper we have proposed a flexible and inference-friendly time calculus motivated by our study of the Penn Treebank corpora. Our approach views the task as a temporal constraint satisfaction problem, which consists of two sets of constraints: those from within a calendar (unary constraints) and those from the inter-event temporal relations (binary constraints in the time-difference form). We have provided a principled account of calendars, on top of which we have built representations for a variety of temporal expressions. A granularity-refined type system with coercion allows us to model the phenomena of implicit granularity conversion and re-interpretation. The resulting TCSP can then be solved using a modified all-pairs-shortest-path algorithm, and the derived minimal network can then be used to answer a wide range of temporal-related queries.

We are currently implementing a system utilizing the formulation and will in time conduct empirical tests. Other future works include investigating the complexity impact brought by the various operators, and possible efficiency improvement via more restrictive cover mappings, such as the string-based representation proposed in [17] , and modeling the movement of temporal focus within this framework, etc.

Acknowledgments

The authors are grateful for the numerous valuable comments received from the anonymous reviewers.

References

[1] Allen, J. F., *Towards a General Theory of Action and Time*, Artificial Intelligence **23** (1984), pp. 123–154.

[2] Artale, A. and E. Franconi, *Temporal description logics*, in: M. F. Dov Gabbay and L. Vila, editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*, MIT Press, to appear .

[3] Bacchus, F. and P. van Beek, *On the conversion between non-binary and binary constraint satisfaction problems*, in: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)* (1998), pp. 311–318.

[4] Dechter, R., I. Meiri and J. Pearl, *Temporal constraint networks*, Artificial Intelligence **49** (1991), pp. 61–95.

[5] Gabbay, D., I. Hodkinson and M. Reynolds, "Temporal Logic: Mathematical Foundations and Computational Aspects," Oxford University Press, 1994.

[6] Han, B., *Time Calculus for Natural Language - Tagging Guidelines*, Unpublished draft, Language Technologies Institute, Carnegie Mellon University (2003).

[7] Han, B. and M. Kohlhase, *A time calculus for natural language*, Unpublished draft, Language Technologies Institute, Carnegie Mellon University, Pittsburgh (2003).

[8] Hauck, R. V., M. Chau and H. Chen, *COPLINK - Arming Law Enforcement with New Knowledge Management Technologies*, in: W. McIver and A. Elmagarmid, editors, *Advances in Digital Government: Technology, Human Factors, and Policy*, Kluwer Academic Publishers, 2002 .

[9] Hobbs, J. R., G. Ferguson, J. Allen, P. Hayes, I. Niles and A. Pease, *A DAML ontology of time* (2002).

[10] Mackworth, A. K., *Consistency in networks of relations*, Artificial Intelligence **8** (1977), pp. 99–118.

[11] Marcus, M., G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz and B. Schasberger, *The Penn Treebank: Annotating predicate argument structure*, in: *ARPA Human Language Technology Workshop*, ARPA Human Language Technology Workshop, 1994.

[12] Meiri, I., "Temporal Reasoning: A Constraint-based Approach," Ph.D. thesis, UCLA (1992).

[13] Ohlbach, H. and D. Gabbay, *Calendar logic*, Journal of Applied Non-classical Logics **8(4)** (1998), pp. 291–324.

[14] Peng Ning, X., S. Wang and S. Jajodia, *An algebraic representation of calendars*, Annals of Mathematics and Artificial Intelligence **36(1-2)** (2002), pp. 5–38.

[15] Pustejovsky, J., R. Saurí, A. Setzer, R. Gaizauskas and B. Ingria, *TimeML annotation guidelines* (2002).

[16] Setzer, A., "Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study," Ph.D. thesis, University of Sheffield (2001).

[17] Wijsen, J., *A string-based model for infinite granularities*, in: *The AAAI-2000 Workshop on Spatial and Temporal Granularity* (2000), pp. 9–16.

[18] Wooldridge, M., C. Dixon and M. Fisher, *A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief*, Journal of Applied Non-Classical Logics **8** (1998), pp. 225–258.