Discrete Time Specifications In Temporal Queries

Philipp Eichmann
Andrew Crotty
Alex Galakatos
Emanuel Zgraggen
Brown University
Providence, RI 02912, USA
peichmann@cs.brown.edu
crottyan@cs.brown.edu
agg@cs.brown.edu
ez@cs.brown.edu

CHI'17 Extended Abstracts, May 06-11, 2017, Denver, CO, USA

http://dx.doi.org/10.1145/3027063.3053222

Abstract
Analysis, exploration, and visualization of time-oriented data are ubiquitous tasks in various application domains, all of which involve the execution of temporal queries. Prior research in interactively specifying the time component for such queries has been focused on defining temporal relationships in data, i.e., querying event sequences through ordinal patterns. However, there has been much less emphasis on how to specify time as a quantitative data dimension in temporal queries. Motivated by the advent of the *Internet of Things* (IoT), we present a formal model that can be used to represent complex time specifications. Our model is the first step in an effort to enhance temporal user interfaces that enables discrete time specifications through a visual query interface.

Author Keywords
Temporal queries; query interfaces; visual query language; query specification

ACM Classification Keywords
H.5.m [Information Interfaces and Presentation: Miscellaneous]:

Introduction

Various research efforts have identified and formulated both goals and tasks for information visualization, many of which include *interaction* in addition to visual tasks. Shneiderman [11], for instance, defined a task by data type taxonomy that comprises interaction tasks such as zoom, filter, and extract. Similarly, Yi et al [14] derived categories of interaction tasks, including "show me a different arrangement," "show me a different representation," and "show me something conditionally." While many interactive data exploration systems [2, 3] have built features to support these kinds of tasks for various data types, most offer only limited support for performing such tasks on time-oriented data.

An exception to this observation is the large body of work in the area of event sequence analysis, which has produced user interfaces that overcome the limited expressiveness of ordinal time specification techniques (e.g., [15, 8, 9, 13, 5]). However, such systems mainly focus on ordinal time, i.e., the relationship between events. Other applications treat discrete time as a regular, linear, and quantitative data dimension, where interactive parametrization is often limited to defining the start and end point of a time interval. Motivated by an IoT scenario, we argue that such systems are not suitable to express queries that exceed the complexity of simple interval- or calendar-based filters, since they either lack expressiveness or have a steep learning curve due to dialog-heavy interfaces or complex query languages.

As part of our endeavor to create a gestural interface for specifying time for temporal queries while also encouraging iterative and fluid exploration, this paper investigates a compact and expressive structure to parametrize discrete time using the basic time building blocks *instants*, *intervals*, and *spans*.

Motivating Examples

Our work is motivated by a recent experiment we conducted in the computer science building at Brown University. Throughout the building, we installed a number of IoT devices that record measurements and events, such as the current energy consumption of printers and vending machines, as well as motion in classrooms and when doors are opened or closed. An example of recorded data can be found in Table 1. In the following, we provide use cases to illustrate the importance of temporal interfaces that enable users to specify complex queries.

Time	Type	Value	Dev Id	Dev Type
1483030839180	open	NULL	d1:front	door
1483030842534	watts	669	v1:coke	outlet
1483030843573	closed	NULL	d2:front	door
1483030844573	watts	669.7	v1:coke	outlet
1483030852564	watts	668.7	v1:coke	outlet
1483030853525	watts	669.2	v1:coke	outlet
1483030857530	watts	668.2	v1:coke	outlet
1483030858033	open	NULL	d1:front	door
1483030858549	watts	669.1	v1:coke	outlet
1483030861628	watts	668.4	v1:coke	outlet
1483030863981	closed	NULL	d1:front	door
1483030866552	watts	669.4	v1:coke	outlet
1483030867478	watts	668.8	v1:coke	outlet
1483030873516	watts	667.7	v1:coke	outlet
1483030874589	watts	668.2	v1:coke	outlet
1483030876625	watts	667.7	v1:coke	outlet
1483030878471	watts	666.4	v1:coke	outlet

Table 1: A sample from our IoT dataset showing measurements made for two doors (d1,d2) and a vending machine (v1).

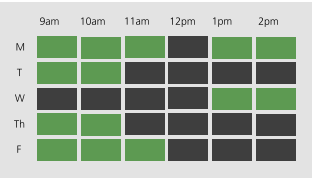

Figure 1: A sample visualization where a facilities manager executed a query to find time slots of two hours in length where the usage of a vending machine is below some threshold (shown in green).

Example 1
The facilities department is responsible for management, maintenance, and scheduling of the building. In regular intervals, a facilities manager opens a web-based monitoring application, which displays information for all motion and temperature sensors as well as energy monitoring devices installed in the building. He is interested in finding anomalies in the recorded temperatures of all rooms in the building. While the application displays a linearly re-sampled representation of all temperature time series so they visually fit on the screen, he wants to zoom into last week's records in order to look at a more fine-grained resolution and detect unusual patterns.

Example 2
The facilities manager notices that the energy consumption of one of two co-located vending machines has remained steady over the course of the past 24 hours, indicating that no sales have been recorded. Since this is abnormal, he then decides to manually examine the machine and determines that there is indeed a technical issue. Because repair work on either vending machine can disrupt the service of the other, he wants to find suitable time slots for scheduling a repairman at times where the working vending machine is used the least. Knowing that repair work typically takes a maximum of two hours, he wants to find times on workdays between 9am and 3pm with low vending machine usage (i.e., low energy consumption). Based on the visualization shown in Figure 1, he schedules an appointment with the repairman.

Background

The goal of our work is to provide a formal description of a structure that can be used to parametrize time as an argument for temporal queries. We first recapitulate important time modeling design aspects that are typically considered when visualizing time-oriented data. We then give a brief description of a few selected systems that support data selection by time, before formally defining our model in the next section.

Time Modeling
A wide variety of models have been proposed in past research. In the following, we provide a brief overview (based on [1] and [4]) of time-related objects and the properties most relevant to our work.

Time Domain Time can be categorized into either an ordinal, discrete, or continuous domain. Ordinal time only defines relative order relations, such as before and after. Continuous time most closely models physical time, where an infinite number of points in time exist between any other two points. Discrete time uses a discrete time unit (e.g., a UNIX timestamp denoting the number of elapsed seconds since January 1, 1970).

Time Primitives Time Primitives are the building blocks of a time model. We distinguish between *instants*, *intervals*, and *spans*. An instant refers to a point in time with finite precision in the discrete time domain and infinite precision in the continuous time domain. We define an interval as a closed interval between two instants. Both instants and intervals are anchored, meaning that they have a fixed position in time. Spans represent a duration of time (e.g., three seconds) and are, therefore, unanchored.

Granularity Time can be represented at different levels of granularity. The granularity of time determines how temporal primitives are transformed to other conceptual units of time. For instance, the Gregorian calendaric elements Saturday and Sunday can be summarized as a weekend, or a day can be represented as 24 hours.

Rind et al. presented similar work on a software library called TimeBench [10], that provides data structures and algorithms for visual analytics of time-oriented data. In particular, they developed a data model that captures the complexity of time-oriented data by mapping data objects to a set of time primitives and vice-versa. Our work, in turn, focuses on a more formal and expressive notation for time specifications that parametrize time as an argument for temporal queries.

Time Specification Techniques
A common technique to drill down in time series data (used by systems such as ChronoLenses [17] and KronoMiner [16]) is to select regions of interest using rubber band selections. In both systems, selected time ranges can be shifted using a drag-and-drop gesture. With KronoMiner, start and end values can be adjusted precisely using a pop-up dialog. TimeSearcher [6] uses a concept called timeboxes, which are similarl to the rubber band selection. Timeboxes are rectangles that can be drawn directly on a line graph visualization of a time series. While the extent of a timebox along the horizontal axis defines the time range of interest, the location on the vertical axis and height of the rectangle specifies the desired amplitude range. Queries can be refined in either dimension using range sliders. A more recent example of an application that allows users to explore time-oriented data and offers similar basic operations is LiveRAC [7], a system tailored to the visual analysis of large collections of system management time series. Tableau [12] offers a way to create parametrization controls, which can be applied as filters. A parameter (e.g., a start date) can be defined with a minimum and maximum value as well as a step size. An interactive slider control can then be used to set the parameter.

Time Specifications

In this section, we introduce a model that can be used to parametrize temporal queries, such as the ones outlined in the previous section. Because time in information technology is usually represented in the discrete domain, we will assume discrete time for all primitives.

We first describe a time-oriented query function Q that we aim to parametrize.

$$Q(\{t_{s0}, ..., t_{sn}\}, \lambda, d) \tag{1}$$

Queries on time-oriented data are typically defined by temporal and non-temporal parameters, $\{t_{s0}, ..., t_{sn}\}$ and λ, respectively, as well as the data d for which the query is defined. We call each t_s a *Time Specification*. A *Time Specification* is a set of *Time Primitives* that each can represent an *instant*, *interval*, or *span*, as we will show. More formally, we define a *Time Primitive* t_p as a tuple $\langle S, \delta, \sigma \rangle$, where the start time S and duration δ represent a time interval $[S, S + \delta)$. σ indicates a stride or time step that we will use to support temporal partitioning. All values in t_s are defined in a time unit (e.g, milliseconds or ticks). Typically, S is defined in $\mathbb{N}$, δ is in $\mathbb{Z}$, and σ can be set to either a value in $\mathbb{N}^+$ or to ∞. The non-temporal argument λ is defined as a function that takes in a finite set of data satisfying the temporal constraint given by all t_s. This could be an aggregate function, such as an average for a particular data attribute. Conceptually, Q first partitions the data based on the temporal specifications given by t and then applies λ to each partition.

With t_p, an instant can expressed by setting S to the timestamp of the *instant*, the duration δ to zero, and the stride σ to ∞. To describe *intervals*, we set S to the start time, δ to the difference between the start and end times, and σ to

∞. By definition, spans are unanchored, i.e., they are not specified by a start time and end time, but rather only by duration. However, spans are typically used to parametrize windowed functions in order to extract intervals of a given span duration (time span) at discrete steps (temporal partitioning). This is reflected in our model by the stride parameter σ. Setting a stride to infinity, as done for instants and intervals, means that λ is only applied to a single partition of data (an instant or interval). Conversely, setting a stride to a positive integer σ, can lead to one or many potentially overlapping partitions, which can be seen as defining multiple Time Primitives at once, each of duration δ but with varying start times. We formally define t_p as

$$\langle S, \delta, \sigma \rangle = \bigcup_{i=0}^{n-1} \langle S + i \cdot \sigma, \delta, \infty \rangle \tag{2}$$

where

$$n = \begin{cases} \text{floor}(\frac{\delta}{\sigma}), & \text{if } \sigma \neq \infty \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

To illustrate the applicability of *Time Specifications*, we provide a number of examples that show how both simple and more complex time constraints can be represented using our model. For simplicity and readability, we do not use a particular time unit in the following specifications.

Examples

11am, January 1st, 2017

$$\downarrow$$

$$\{\langle \text{11am, January 1st, 2017, 0}, \infty \rangle\}$$

A single instant can be represented by a *Time Specification* with a single time primitive.

January 1st and January 3rd, 2017

$$\downarrow$$

$$\{\langle \text{January 1st, 2017, 0}, \infty \rangle\}$$

$$\cup$$

$$\{\langle \text{January 3rd, 2017, 0}, \infty \rangle\}$$

Consequently, n non-consecutive instants are represented by n time primitives.

10am to 2pm, February 20st, 2017

$$\downarrow$$

$$\{\langle \text{February 20st 10am, 4h}, \infty \rangle\}$$

An single interval is represented by setting the duration. Analogous to the previous example, non-consecutive intervals can be expressed by defining multiple time primitives.

Past hour

$$\downarrow$$

$$\{\langle \text{Current time, -1h}, \infty \rangle\}$$

Negative durations can be used to specify an interval from $S - \delta$ to S.

All weekends

$$\downarrow$$

$$\{\langle \text{First Saturday 1970 in Unix Time, 2d, 7d} \rangle\}$$

A span is defined by setting the stride to a positive integer, in this case seven days. Since “all weekends” does not constrain the time range in which weekend intervals are to be queried, the start time of this specification is set to the first occurrence of a Saturday in the given time unit.

All 24h windows with stride of 1h in the past week

$$\downarrow$$

$$\{\langle \text{Current Time, -7d}, \infty \rangle\}$$

$$\cap$$

$$\{\langle \text{First Unix day, 24h, 1h} \rangle\}$$

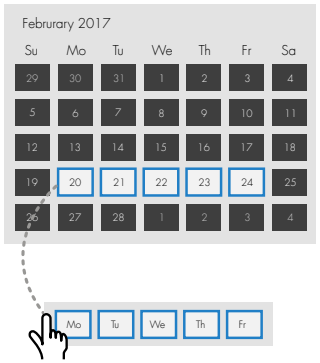

Figure 2: Depicts a possible scenario of how a time primitive can be turned into a new time specification. In this case, an interval (Mo, Feb 20 to Fr, Feb 24) is dragged out of its original context (the calendar) and thereby converted to *all weekdays* (query-by-example).

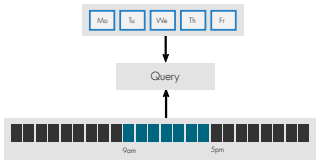

Figure 3: Shows two time specifications (Mo-Fr and 9am-5pm) that fully specify the temporal argument of a query.

In this example, the user is looking for a span (24h) within a certain interval (past week). Since the query has to satisfy each of the time constraints imposed by a time specification, we can limit the temporal search space for a span by adding an additional interval primitive to the time specification.

Discussion and Future Work

Time Specifications compactly describe discrete time instants and intervals as well as periodic intervals and can easily be translated into queries over data (e.g., using SQL). Their expressiveness has been particularly useful to formulate sample queries, which we used to determine and compare the capabilities of existing interactive data exploration systems that support time-based data. Consider our motivational examples: a possible time specification for *last week* in Example 1 could be $\langle$Current time, -7d, $\infty\rangle$. The time constraints in Example 2 could be described by first specifying weekdays as $\langle$First Monday in Unix Time, 5d, 7d$\rangle$ and then intersecting it with the daily time range $\langle$First Unix day 9am, 8h, 1d$\rangle$.

Although our model makes it easy to reason about and express time-dependant queries, there are many open questions that must be answered so that users can easily and naturally express queries in a visual interface. While we showed that a wide set of time-based constraints can be specified using just three parameters, a remaining challenge is to determine how users can define the parameters to construct a time specification in a user interface, and which operations they can use to derive parameters from existing specifications. A key aspect thereof is the notion of granularity, since humans use abstractions when they talk about time. The parameters in the example time specifications shown in the previous section are defined in different time units such as hours, days, weekends, and years. Although many systems provide mappings from higher level granularities to a discrete time domain (e.g., Unix time and vice versa), many support only a fixed set of pre-defined granularities, as Rind et al. [10] point out. While useful in some some cases, we believe that an expressive visual query language should facilitate intuitive ways to define, modify, and derive the parameters used in our model at both pre-defined as well as custom granularities. Figure 2 schematically depicts a case where an anchored interval can be turned into a span by moving it out of its original context.

Based on this work, we are currently designing a novel gestural interface that allows for incremental specifications of time-oriented queries. We envision an iterative query building workflow that resembles the iterative build-up of formal specifications. As described in the previous paragraph, a user with the intention to constrain a temporal query to hours 9am - 5pm on weekdays would split up the query into (1) specifying weekdays, (2) specifying the hours, and (3) connecting the two specifications to fully define the temporal arguments of the query. Figure 3 schematically illustrates such a scenario.

References

[1] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. 2011. *Visualization of time-oriented data*. Springer Science & Business Media.

[2] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2015. Vizdom: Interactive Analytics through Pen and Touch. In *PVLDB*.

[3] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald A. Metoyer, and George G. Robertson. 2012. GraphTrail: analyzing large multivariate, heterogeneous networks while supporting exploration history. In *CHI*.

[4] Iqbal A Goralwalla, M Tamer Ozsu, and Duane Szafron. 1998. A framework for temporal data models: exploiting object-oriented technology. In *Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings*. IEEE, 16–30.

[5] David Gotz and Harry Stavropoulos. 2014. Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 1783–1792.

[6] Harry Hochheiser and Ben Shneiderman. 2001. Interactive exploration of time series data. In *International Conference on Discovery Science*. Springer, 441–446.

[7] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. 2008. LiveRAC: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1483–1492.

[8] Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. 2013a. Temporal event sequence simplification. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2227–2236.

[9] Megan Monroe, Rongjian Lan, Juan Morales del Olmo, Ben Shneiderman, Catherine Plaisant, and Jeff Millstein. 2013b. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2349–2358.

[10] Alexander Rind, Tim Lammarsch, Wolfgang Aigner, Bilal Alsallakh, and Silvia Miksch. 2013. Timebench: A data model and software library for visual analytics of time-oriented data. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2247–2256.

[11] Ben Shneiderman. 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, 336–343.

[12] Tableau 2017. Tableau. http://www.tableau.com. (2017). Accessed: 2017-01-01.

[13] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. 2011. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1747–1756.

[14] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. 2007. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1224–1231.

[15] Emanuel Zgraggen, Steven M Drucker, Danyel Fisher, and Robert Deline. 2015. (s| qu) eries: Visual Regular Expressions for Querying and Exploring Event Sequences. (2015).

[16] Jian Zhao, Fanny Chevalier, and Ravin Balakrishnan. 2011a. Kronominer: using multi-foci navigation for the visual exploration of time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1737–1746.

[17] Jian Zhao, Fanny Chevalier, Emmanuel Pietriga, and Ravin Balakrishnan. 2011b. Exploratory analysis of time-series with chronolenses. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2422–2431.