Visualizing Operations on Temporal Data

Steve B. Cousins and Michael G. Kahn
Medical Informatics Laboratory
Washington University School of Medicine
Saint Louis, Missouri

Abstract

The detection of temporal relationships among time-ordered patient data is an important, but difficult, clinical task. The ready availability of large volumes of computer-stored clinical data offers the possibility of early detection of undesired trends and untoward effects. Unfortunately, large volumes of data also can obscure relevant findings and relationships. To aid in exploring large data sets for complex temporal relations, robust and intuitive temporal visualization techniques are required. We are developing an interactive data browsing environment designed specifically to support temporal manipulations. The unique contribution is the combination of a formal system of temporal operations with a visual system for interactive temporal data display. The basic temporal construct is a time line. For each of the basic time line operations, SLICE, FILTER, OVERLAY, and NEW, we precisely define the effects of each operator, including exceptions and boundary conditions. For each formal definition, a precise visual analog of the operation is given. In addition, we describe a time line browser, an editor for time lines, which implements this formalism. We believe that the existence of formal definitions for all visualization operators results in more logically consistent operators, especially with respect to errors and exception conditions.

Introduction

Advances in instrumentation have resulted in the ability to observe, measure, and record vast quantities of biomedical data. The temporal nature of biological information and the presence of a large number of data types adds additional complexity to the understanding of these data. We are developing a formal methodology for visualizing temporal relationships among biological data, in order to assist in the interactive exploration of these data. Helman and Hesselink list three major components of a visualization system: analysis, display, and interaction [4]. We previously addressed the analysis component by introducing the use of temporal heuristics to aid in the selection of relevant data elements for display [3]. In this paper, we focus on the interaction component of the visualization task. We define a set of operators and their visual analogs that are used to manipulate temporal information.

Studies in computer-human interface have shown that consistency among interface operations is a key factor in the usability of software. Formal definitions are one method of ensuring logical consistency. With formal definitions of our operations, we can more carefully examine boundary cases and exceptions—the areas where most inconsistencies occur. Visualization researchers traditionally have not emphasized consistency, but have focused on developing new operations and algorithms instead. Our contribution is to emphasize consistency, while developing our visualization system.

For each temporal operator, we present a formal definition that completely describes the behavior of this operator in all situations. By defining the operations formally, we describe unambiguously the direct and indirect effects of each temporal manipulation. These careful definitions should help to eliminate the possibility of anomalous or unintuitive behavior.

Our formalization is based on the notion of the time line. Intuitively, a time line is a linear sequence of events ordered by time. The visual representation of a time line is a two-dimensional object, with one axis being used to denote time, and the other axis being used to label data classes (Figure 1). Labels vary by data class. For example, numeric data may be labeled by a plot or numeric display, whereas non-numeric data may be labeled as a string or an icon. The events on our time line are not restricted to simple points, but may be intervals or events of uncertain time or duration. In Table 1, we describe the default visual representation for all of these event types. A more detailed discussion of event types has been previously presented [3].

We give formal definitions for the following operations on time lines: SLICE, FILTER, OVERLAY, and NEW (Figure 2). Informally, SLICE corresponds to removing events from one or both ends of a time line, thereby reducing its size. Note that a time line can be copied by slicing from its first event to its last event. FILTER removes events satisfying an arbitrary predicate from the time line. OVERLAY corresponds to combining two time lines into one. One event in each time line is specified as the aligning event; the new time line contains all of the events of the old ones aligned on the specified events. Note that concatenation can be performed as a special case of OVERLAY by using the last event of the first time line and the first event of the second time line as the aligning events. NEW creates a new, empty time line. A time line browser is an editor for a collection of time lines (Figure 3). Time lines are arranged within the time line browser by the user in whatever way best helps him to visualize time. All four operations on time lines have visual analogs in the time line browser. Because each time line operation has a strict formal definition, the semantics of these operations are unambiguous. Likewise, the visual analogs to these operations are
precisely defined. We have defined a second class of operations that are applied only to time line browsers. These browser-level operators manipulate the visual characteristics of time lines, e.g., their arrangement, coloring, or labeling. The most interesting of these operators are ALIGN, SCALE, and MARK. ALIGN arranges a time line within the time line browser so that one of its events is always lined up horizontally with a specified event of another time line. SCALE changes the number of pixels per time-unit that are used for display. MARK adjusts the visual properties, such as the color or shape, of selected items.

### A Clinical Scenario

Using the analysis and display of diabetes patient data as our application area, we have implemented a time line browser prototype which displays a patient's medical history as a time line. This prototype system supports the time line manipulations we have described. One key feature for visualizing large data sets collected over a long period of time is the ability of our time line browser to allow the user to zoom in and out to different levels of temporal abstraction (Figure 4). By eliminating the visual clutter caused by unwanted detail, the user of the browser can perform time line manipulations with only those concepts that are required to solve his current analysis problem.

In Figure 3, we show a time line browser for a hypothetical diabetic patient. The uppermost time line in the time line browser represents the output of the patient's diabetic log book\(^1\). The run of markedly hyperglycemic readings (indicated by “x”s) on Monday through Wednesday. The second time line is a slice from the patient's personal calendar (a form of time line), which is temporally aligned with the log book. When available, a calendar can give a physician additional information about the patient's lifestyle that the medically-oriented log book does not include. In this case, the calendar indicates that the patient was travelling to visit family members during those three days, which in turn suggests the hypothesis that the patient had either more food or less exercise than usual due to the break in his normal schedule. The third time line is a modal day plot in Figure 3 is separated from the first two by a dashed line, indicating that it is not temporally aligned with them. This time line is a modal day—created by overlaying a whole week's worth of days. The modal day is popular in the domain of diabetes management because it gives an indication of the range of values at different times during the day. In the sections below, we give formal definitions for all the operations required in this scenario.

\(^1\)Most diabetic logbooks are recorded by hand today, and do not usually have this variety of information. However, hand-held electronic logbooks are being developed, which will make this information much more readily available to analysts in the near future.

---

#### Figure 1: A time line displaying various event classes that might be found in a diabetic patient record. The small circles represent blood glucose readings, positioned vertically by value. The horizontal positions of the X-ray icon and the speakers indicate when an X-ray was taken or when voice-commentaries were made to the record, respectively. The widths of the boxes denoting intervals indicate their temporal duration.

#### Figure 2: Operations on time lines. We use the more traditional visual representation for time lines in this figure for clarity.

### Time Lines

Formally, a time line is a tuple \(< E, M >\), where \(E\) is a finite set of events containing at least the special null event \(e_0\), and \(M\) is a measure function \(M : E \rightarrow R^+\). The measure function \(M\) assigns a temporal offset to each event of \(E\). By definition, \(M(e_0) = 0\). Another way to conceptualize the measure function is as a relation between events. In this case, \(M(e_i)\) is a shorthand for \(M(e_0, e_i)\). One special time line is the null time line, \(TL_0 = \{ e_0 \}, M >\).

Intuitively, a time line is a line segment with \(e_0\) as its leftmost boundary, some \(e_i\), such that \(M(e_i)\) is maximal as its rightmost boundary, and all other events placed in between according to the temporal ordering imposed by \(M\). The measure function implies a total ordering on \(E\). One widely used ordering function is calendar time, i.e., minutes, days, months, etc. Our definition does not require the existence of calendar time but does permit the mapping of time lines onto calendar time if required.

An essential capability of a time line definition is to make clear the distinction between a temporal ordering and calendar time. For example, the modal day plot in the scenario above contains a temporal ordering, but does not map onto any particular part of the calendar. In order to separate these ideas, we need to suspend the notion of calendar time, but keep the proper temporal ordering. Our basic time line formalism contains only the notion of temporal ordering of events. The concept of grounding time lines to "real time" on a calendar is a logical extension which is easily incorporated into our formalism.

In the following sections, we formally define the set of time line

---

\[\text{Input} \quad \text{Operation} \quad \text{Output}\]

| TL1 | Slice(TL1,[1,2]) | TL1[1,2] |
| TL1 | Fill(TL1,"a-test") | TL1 | TL1[1,2] |
| TL1 | Overlay(TL1,a1,TL2,a2) | TL2 | TL1[1,2] |

---

Hand-held electronic logbooks are being developed, which will make this information much more readily available to analysts in the near future.
A time line browser is an editor for time lines. This time line browser about a diabetic patient contains three time lines: (a) a diabetes logbook record; (b) a slice from the patient's personal calendar; (c) a modal day plot created from (a) by slicing it into days and overlaying them. "B L D N" is a short-hand for the time of day: Breakfast, Lunch, Dinner, or Nighttime.

**Slice**

Slice removes events from one or both ends of a time line. A time line $TL := <E, M>$, sliced from $e_1$ to $e_2$ ($e_1, e_2 \in E$ and $M(e_1) \leq M(e_2)$), yields a new time line

$$TL' := <E', M'> = \text{SLICE}(TL, e_1, e_2)$$

where

$$E' = \{e \in E | M(e_1) \leq M(e) \leq M(e_2)\} \cup \{e_0\}$$

$$M'(e) = \begin{cases} M(e) - M(e_1) & \text{when } e = e_0 \\ M(e_1) - M(e_1) & \text{otherwise} \end{cases}$$

Equation 1 describes the structure of a call to SLICE, using standard functional notation. Equation 2 adds all events whose measure function puts them between $e_1$ and $e_2$ inclusive, to the new time line. The way this definition is structured implies that the events between $e_1$ and $e_2$ will exist in both time lines—$E'$ is not made up of copies of the events from $E$. Equation 1 also guarantees that the null event $e_0$ is in $E'$. Equation 3 defines a new measure function for the new time line, which maintains the same temporal offsets in $TL'$ as in $TL$. Note that the definition of SLICE permits $e_1 = e_2$, and that $\text{SLICE}(TL, e_0, e_0) = TL_0$.

**Filter**

Filter removes all events not satisfying an arbitrary predicate $P$ from a time line $TL := <E, M>$. By definition, the null event $e_0$ cannot be removed by any predicate. The new time line is

$$TL' := <E', M'> = \text{FILTER}(TL, P)$$

where

$$E' = \{e \in E | P(e)\} \cup \{e_0\}$$

Equation 5 applies the predicate $P$ to all events except $e_0$. The null time line $TL_0$ is generated whenever a predicate $P$ removes all events (other than $e_0$) from the original time line $TL$.

**Overlay**

Overlay puts all of the events in two time lines into a single one, so that an event from the first time line and one from the second have the same measure function in the new time line. A subtle complication occurs in the OVERLAY operation that does not occur in any other temporal operation. If both original time lines have an event in common, it is undesirable to have that event duplicated in the new time line at the same point, since then even simple operations such as counting the events in the time line would not act as the user expects. On the other hand, if the alignment offsets the common event, the measure function could not have two values, and in this case duplication would be in order (Figure 5). In order to enable this, we use an operation called copy to do the necessary duplication in Equation 10. We define copy for events as $\text{copy}(e) = e'$ such that $\{P(e')\} = P(e)$ and $e \neq e'$ where $P$ are arbitrary predicates defined over events.

$\text{OVERLAY}$ combines two time lines

$$TL_1 := <E_1, M_1>$$

$$TL_2 := <E_2, M_2>$$

into a new time line,

$$TL' := <E', M'> = \text{OVERLAY}(TL_1, e_1, TL_2, e_2)$$

aligning the time lines on $e_1 \in E_1$ and $e_2 \in E_2$, where

$$E' = E_1 \cup E_2 \cup \{\text{copy}(e) | (e \in E_1 \cap E_2 \setminus \{e_0\}) \land (M_1(e_1) - M_1(e) = M_2(e_2) - M_2(e))\}$$
At any given time, a subset of the objects in a time line browser (time lines and the events they contain) may be selected [2]. In general, objects are selected by clicking on them. An entire time line may be selected by clicking on its border. Multiple events may be selected by holding the add to selection modifier key down (typically the shift key). An entire class of events may be selected by clicking on a prototype event of the class with the class modifier key down.

**Visualizing Time Line Operations**

In this section, we discuss the visual analogs of the time line operations.

**Visually**, **SLICE** is performed by selecting a range of events and choosing the SLICE option from a menu. The first event in the selection is taken as \( e_1 \) and the last event selected is taken as \( e_2 \). The new time line is inserted in the time line browser immediately after the one from which it was sliced.

**Visualizing FILTER** is complicated by the need to specify the predicate \( P \) in Equation 4. Because all events are members of some event class, this problem is solved best in general at the event-class level. However, we do provide shortcuts for a few simple predicates. In particular, the predicate \( P_{del}(e) \equiv (e \notin C) \) (for an arbitrary event class \( C \)) which removes all events in class \( C \), is performed by selecting a class of events, and then striking the delete key. Similarly, a specific event is removed by selecting it and striking the delete key, which invokes the operation \( del(e') \equiv \text{FILTER}(TL,P) \) where \( P_{del}(e') \equiv e' \notin C \). In general, however, selecting an event and choosing FILTER from a menu invokes a class-specific method for constructing predicates relevant to the event's class.

**OVERLAY** is performed by dragging the aligning event of one time line onto the aligning event of a second time line. The resulting time line replaces the second time line in the time line browser.

Finally, a new (null) time line is created by choosing the new command from a menu. A new time line can also be created from existing time lines by taking an arbitrary SLICE from a time line, and then using FILTER to remove all events.

**Visualizing Time Line Browser Operations**

The utility of the time line browser is greatly enhanced by adding a few browser-level operations to the collection of time line operations defined above. The following operations allow the user to align time lines in the time line browser, to change the time SCALE (temporal granularity), and to MARK events.

**Align** ALIGN is used to arrange two adjacent time lines so that appropriate events are lined up vertically (Figure 3). Typically, ALIGN is used to give two time lines a common temporal basis. For example, two time lines representing the same period of calendar time would be aligned on a common day, or two pregnancies may be aligned on the dates of conception.

**Scale** SCALE refers to the amount of time represented by a unit of space on the horizontal axis (Figure 4). To avoid visual confusion, all time lines within a single time line browser are drawn to the same temporal scale. For example, the scale in Figure 1 is approximately 1 day = 1 inch. Zooming out to a coarser temporal granularity and zooming in to a finer temporal granularity are implemented as changes in the scale of a time line browser.

**Mark** Selecting a set of events and choosing MARK temporarily alters the visual characteristics of the selected events. For example, the marked events may have their shapes modified, or on a color monitor the events may all be drawn in a new color. In Figure 3, vacation events are marked with an "x" rather than a "o." Although a MARK is an attribute of a time line browser, its effect is to change the visual characteristics of a time line.

**Other operations** Time lines may be selected and saved to a file individually, or time line browsers may be saved. If the entire time line browser is saved, all positioning information is retained in the file. Similarly, time lines can be loaded into new time line browsers, or time line browsers which have been saved can be restored.
Clinical Example Revisited

We now return to the clinical scenario described earlier. One possible time line browser describing this scenario is shown in Figure 3. We begin by opening a diabetes log book data file containing all events stored as a single time line. Opening the time line data file creates a time line browser containing a single time line. A second time line, representing the patient’s personal calendar, is loaded into the same time line browser, and aligned on the start of Monday. The third time line, the modal day, is formed by filtering all events except blood glucose readings from the first time line, marking the values on the three-day weekend, slicing the resulting time line into single day time lines, and then overlaying the single day time lines onto each other. At this point, the physician may choose to save the time line browser configuration, or may continue to pursue hypotheses suggested by the current display.

Future Work

Since the measure function we define is real-valued, a typical numerical methods issue arises: “When are 2 real-valued times represented in the computer considered equal?” The standard answer is to use the inequality $|t_1 - t_2| < \epsilon$, but with temporal equality, it becomes clear that the concept of temporal granularity is an important component in determining if two points in time should be considered the same. Temporal granularity is the unit of a time scale appropriate for a given problem-solving context [5]. Exact seconds do not matter if the concept of interest ranges over years, but seconds become important when that concept evolves over minutes. Therefore, epsilon is actually a function, dependent on the temporal granularity of the time line. Although we have not yet experimented with different functions for epsilon, good candidates seem to be logarithmic and few functions based on time units (seconds, minutes, hours, days, months, years...).

We have not dealt with incorporating temporal uncertainty into our time line operations. Others have shown the general solution to temporal uncertainty to be an NP-hard problem [1; 6]. The framework we provide allows one to specify wide bounds on the occurrence of events, but we have not dealt with how to resolve that temporal uncertainty. For example, assume we have two events regarded as having occurred at 11:55 AM. When viewed at a fine temporal granularity where 1 inch on the screen represents 1 micro-second, it becomes clear that the events could have occurred at different times. In these cases, we add error bars to the event’s visual representation to indicate that, at this temporal granularity, the exact location of the event on the screen is not indicative of its exact temporal location.

There are other visual representations of time in common use which we believe our system will be able to handle, but which we have not yet implemented. For example, we have examined a large number of temporally-oriented statistical graphics from Tufte’s book [5], and are confident that our framework can handle them. One type of visual representation that presents difficulties is the “traditional” time line: A single line with other lines perpendicular to it, such as in Figures 2 and 5. One way to handle this visual representation is to require event classes to respond to a new message, “traditional draw”. Unfortunately, this requirement would add to the burden of programmers of event classes, making them define two separate draw routines (or in the worst case, one for each possible visual representation). Another useful visual representation displays time lines vertically (as in a patient chart or date book). We have a tentative design which we believe will allow us to draw these alternative time line formats without placing such a burden on event class programmers. At the current time, our prototype draws time lines in the format used in Figures 1 and 2.

We believe the time line browser will become a very useful tool, both inside and outside the domain of medicine. However, before it becomes really useful, we need to develop thoroughly the classes of events and their behaviors. In addition, we believe time lines should be saved in databases rather than in files, so that sets of events can be shared and more readily manipulated. These features remain future work.

Conclusions

It will take more than fast display algorithms to satisfy the needs of physicians to visualize patient data. We believe that having a powerful, concise, and intuitive set of operations is absolutely necessary to allow clinicians to perform more thorough analysis of temporal data. Therefore, we have developed formal specifications that precisely define the temporal characteristics of all time line operations. Each definition describes both the alterations in a time line imposed by the operation and a description of a visual format for displaying the results of the operation. We seek to develop a complete calculus of time line manipulations and visualizations that can be generalized to other sources of time-oriented biological data. With precise semantics and a complete toolbox of operators, we believe the time line concept will become a useful visualization approach to browsing biomedical data.

Acknowledgements

This work was partially supported by the Washington University Center for Intelligent Computer Systems, and by a generous hardware grant from Apple Computer, Inc. We sincerely appreciate the comments of Charlene Bloch Abrams, Mark E. Frisse, Guillermo R. Simari and Rose F. Gamble on previous versions of this manuscript, which have made it much better.

References