

The Challenges of Specifying Intervals and Absences in Temporal Queries: A Graphical Language Approach

Megan Monroe

Department of CS & HCIL,
University of Maryland
madejyay@umd.edu

Rongjian Lan

Department of CS & HCIL,
University of Maryland
rjlan@cs.umd.edu

Juan Morales del Olmo

HCIL & Universidad
Politécnica de Madrid
juanmoralesdelolmo@gmail.com

Ben Shneiderman

Department of CS & HCIL,
University of Maryland
ben@cs.umd.edu

Catherine Plaisant

Department of CS & HCIL,
University of Maryland
plaisant@cs.umd.edu

Jeff Millstein

Oracle
jeff.millstein@oracle.com

ABSTRACT

In our burgeoning world of ubiquitous sensors and affordable data storage, records of timestamped events are being produced across nearly every domain of personal and professional computing. The resulting data surge has created an overarching need to search these records for meaningful patterns of events. This paper reports on a two-part user study, as well as a series of early tests and interviews with clinical researchers, that informed the development of two temporal query interfaces: a basic, menu-based interface and an advanced, graphic-based interface. While the scope of temporal query is very broad, this work focuses on two particularly complex and critical facets of temporal event sequences: intervals (events with both a start time and an end time), and the absence of an event. We describe how users encounter a common set of difficulties when specifying such queries, and propose solutions to help overcome them. Finally, we report on two case studies with epidemiologists at the US Army Pharmacovigilance Center, illustrating how both query interfaces were used to study patterns of drug use.

Author Keywords

Query languages; temporal query; event sequences; query interfaces; electronic health records.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI)

INTRODUCTION

In 1992, a small class of elementary school students (including one of the authors of this paper) was given a simple writing assignment: describe how to make a peanut butter and

jelly sandwich. After a short period of fervent scribbling, papers were handed in to a teacher, who sat at the front of the room, surrounded by every tool and ingredient that one might need to construct the aforementioned confection. Then, one by one, the teacher acted out each set of instructions with the strictest adherence, and with total disregard for disaster or absurdity. Amidst the cackling laughter of the delighted students, peanut butter was spread onto walls, jelly onto desks, and bread was pressed into the carpet. The lesson: complexity can mask itself within a seemingly simple concept.

Nearly 20 years later, we found ourselves in the same position as we watched users specify temporal queries: they would begin with an idea that was so simple and purposeful, and end up with metaphorical peanut butter smeared onto metaphorical walls.

The difficulty with questions involving temporal event sequences is not necessarily understanding the underlying complexity, but articulating that understanding into meaningful queries. Users assume that the simplicity with which they perceive these events, will translate just as easily to the underlying search application. For queries involving patterns of point events (events that occur at a single point in time), this assumption typically holds true. However, such simple events do not adequately cover the complete range of both medical and real world phenomena. Our main partners, epidemiologists at the US Army Pharmacovigilance Center (PVC), are primarily responsible for conducting drug related studies involving prescription administration and medication interaction. Their data and their inquiries, are inherently interval-based. For example, they might need to know when two medications are being taken at the same time. Additionally, they frequently explore questions in which the absence of an event is the critical point of interest. For example, they might be looking for patients who did *not* experience a symptom after receiving a medication.

When these types of questions arise, user strategies for specifying queries tend to remain tethered to the simplistic logic of point events, despite the increase in the underlying complexity. The result is queries that fail to match the users' intention.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2013, April 27–May 2, 2013, Paris, France.

Copyright 2013 ACM 978-1-4503-1899-0/13/04...\$15.00.

This problem occurs across a wide range of both clinical and non-clinical domains.

In this paper, we present a two-part user study, designed to determine the central difficulties of temporal query specification, and alleviate these difficulties through graphical interaction. We report on the design of a basic, menu-based interface and an advanced, graphic-based interface, both of which are integrated into the LifeFlow visualization tool [25]. We then assess the performance of these query interfaces in assisting the PVC epidemiologists answer their evolving research questions. Our contributions are as follows:

1. An assessment of the primary user difficulties in specifying queries involving intervals and absences.
2. A discussion of the differences between interval event absences and point event absences.
3. Two novel temporal query interfaces, designed to offer intuitive access to a wide range of temporal relationships.

This paper is organized as follows: we begin by relating temporal search to a standard search model and discussing the critical factors of each stage. We then survey previous work in temporal query according to these critical factors. Next, we describe four common stumbling blocks that users encounter when specifying queries involving intervals and absences, and then introduce two query interfaces and discuss the design decisions behind them. We evaluate these interfaces in two case studies with PVC researchers, and finally, discuss future work and conclude.

A STANDARD SEARCH MODEL

As a basis by which to discuss and compare related work, our first step was to frame temporal query in the context of a standard search model. As many such models have been proposed [22, 18, 2], our purpose was not to evaluate these models against each other, but merely to select one that could serve as a straightforward guide. Hearst [10] breaks the search process down into three stages: Specification, Presentation of Results, and Reformulation. For each of these stages, we propose design goals, devised to facilitate the successful execution of temporal queries:

Specification

In the Specification stage, users construct and submit their initial query to the search application. Perhaps most important in this stage, is that a diverse array of potential users be able to complete the specification process. Users must be able to specify their query using a language that is both accessible and intuitive. Furthermore, the interface should provide users with as much information as possible about how the application is going to interpret their query. Errors should only arise through user misinterpretation of the dataset, not through user misinterpretation of the query interface.

Presentation of Results

In the Presentation stage, the application returns a set of results, based on the query. The critical component of this stage is a presentation that gives users enough information to refine their query if necessary. Clinical researchers typically group retrieval errors into two categories: false positives and false

negatives. An ideal interface should provide insight into both of these types of errors. Users should be able to look at the result set and quickly determine whether their query needs to be refined and if so, how.

Reformulation

In the final stage of the search process, users reformulate and resubmit their query based on the information obtained in the Presentation stage. Here, it is important that the interface allows users to focus solely on the refinement. Users should not have to respecify their entire query; they should be able to modify only the components that need to be either added or removed. This allows users to maintain their thought process and continually move closer to their final result set.

RELATED WORK

Our work towards creating an intuitive temporal query system was rooted in three major spheres of prior work: temporal logic, temporal search, and visual query languages. They are discussed here in the context of Hearst's search model.

Temporal Logic

We draw first on methods to logically represent intervals, most notably the work of Allen et al. [1]. Allen identified 13 unique relationships between two intervals and developed a logic capable of representing each of them. This work has been substantially reinforced by work in the data mining community [14, 27, 19]. However, Allen's work does not address the relationships created by point *and* interval events, or the absence of an event, both of which our users have expressed to be critical components of their work.

Temporal Search

Querying temporal data has become increasingly important due to the rapid growth of time-related data from domains such as electronic health care, web logging, and financial analysis. Database researchers have produced numerous works on designing temporal algebras and languages to support access to temporal relationships. Snodgrass [21], Jensen et al. [12], and Das et al. [7] have all proposed extensions to the well-known database querying language, SQL, to encompass temporal relationships, including constructs for specifying both intervals and the absence of an event [8]. However, this work all centers around command-based query languages, which have a notoriously steep learning curve (see Figure 13). For most users, this renders the Specification stage of the search process infeasible. Researchers outside of the database community cannot be expected to learn, and thus utilize these complex languages. Thus, command-based query is not a tenable solution for widespread use.

Visual Query Languages

Just as the graphical user interface revolutionized the way that people interact with computers, visual query languages reduce complexity by conveying information with meaningful graphical representations [17, 6, 16]. Our focus, not surprisingly, was in surveying previous work in temporal visual query languages. The majority of these efforts have allowed users to query for patterns of point events [23, 9, 26], including languages that allow for the absence of a point event to be

specified [24]. Other work has included constructs for specifying interval events within a query [5, 4, 11]. However, none of these languages supported the specification of both intervals and absences, which greatly increase the complexity of queries both from a logical and a representational standpoint.

We extracted three guiding lessons from these surveyed works. The first is that the Specification stage is greatly facilitated when the graphical elements used in the query language closely resemble those that comprise the actual records. For example, Jin and Szekely [13] allow users to drag elements directly from the display of patient records into the query interface. Second, users should be able to directly manipulate query elements, exemplified nicely by the QueryLines system [20]. Many visual query languages use only form-based tools to modify the elements in the query display, which forces users to substantially backtrack when only minor changes are necessary. Finally, while visual query languages can make the Specification stage of the search process more accessible, they cannot single-handedly improve the Presentation or Reformulation stages. These languages must be accompanied by thoughtful strategies for result analysis.

THE CHALLENGES OF INTERVALS AND ABSENCES

Reasoning about intervals and absences is central to clinical research as well as many other domains of temporal search. As such, our first goal was to better understand the types of questions that users ask of their data, and the difficulties they encounter when trying to formulate those questions into meaningful queries.

Our understanding of temporal query construction was initially shaped by a series of tests and interviews with users of an early version of the LifeFlow visualization tool. We conducted both individual and group user sessions with seven researchers at the PVC, for a total of eight hours. In addition, we conducted similar one-hour sessions with many other clinical researchers, including researchers at Washington DC Children’s Hospital, Yale Medical School, and Harvard Medical School. From these sessions, we began to see four common difficulties emerging across users from multiple clinical domains.

(The graphics used in Figures 1 through 12, while rooted in those used in the advanced query interface, are adapted here to illustrate the difficulties being described.)






	Point Event
	Absence of Point Event
	Compacted Interval Event
	Expanded Interval Event
	Absence of Interval Event

Table 1: Event graphics, for reference throughout this section.

Difficulty 1: Specifying Intervals as Point Events

As mentioned previously, most users are able to easily understand temporal relationships involving point events, and specify successful queries to access them. What we frequently

saw then, was that users would attempt to apply these same querying strategies to questions involving intervals. That is, they would specify a query for an interval event as if it were two separate point events (one representing the interval’s start and one representing the interval’s end). Thus, they would try to specify the following event sequence, where a Stroke occurs *while* the patient is taking Drug A (Figure 1).

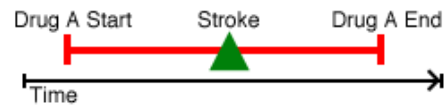


Figure 1: Intended query: Stroke occurs during Drug A.

Unfortunately, they would construct the query as three disjoint event points - paraphrased as:

Find patients for whom Drug A was started, who then had a Stroke, and who then stopped taking Drug A.

This query is analogous to the query in Figure 2:

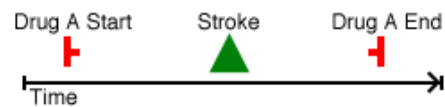


Figure 2: User specified query for a Stroke occurring during Drug A.

The problem is that users view the beginning of the Drug A prescription as a context change: the patient is now taking Drug A. They assume that this context remains in place until the end of Drug A is specified. They also assume that the underlying application will interpret this context in the same way, however this is not necessarily the case. Many databases and data structures store interval events as two separate point events, linked by a unique identifier. Thus, this query returns cases like Figure 3, where a Stroke occurs between two separate prescriptions.



Figure 3: False positive for “during” query.

The lesson here is that the fundamental temporal relationship created by intervals, that of “during,” cannot be adequately specified using the same “before” and “after” strategies used to specify point event relationships. Interestingly, this issue only arose (but arose very frequently) during our initial tests with clinicians, who primarily use command-based querying tools. The graphical representation of intervals that was used throughout our design study completely eliminated this problem. This shows that a successful query tool can guide users away from these errors by explicitly allowing them to express temporal context in the query specification.

Difficulty 2: Specifying Absence as Non-Presence

Throughout our interviews, users often assumed that by not specifying the presence of an event, they were, by implication, specifying its absence. For example, a researcher looking for potential causal relationships may need to find all patients who were prescribed Drug A, and then had a Stroke after the prescription ended. The resulting query would frequently look something like Figure 4:

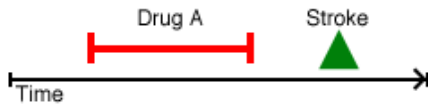


Figure 4: User assumes an implied absence of a Stroke during Drug A.

The assumption here, is that the absence of a Stroke while taking Drug A is implied by the fact that the presence of a Stroke was only specified after the Drug A interval. The problem with this assumption, is that it also necessarily implies that no other type of event occurred while taking Drug A. It also then, implies that no events occurred before Drug A, or between Drug A and the Stroke, or after the Stroke. Essentially, if not specifying the presence of an event implies its absence, then the querying system can only return exact matches. In this case, it can only return records that consist of a Drug A prescription, followed by a Stroke.

Suffice to say, this is what users typically report as their intention, nor is an exact match query system a particularly useful tool for searching real world datasets. Because of this, it is critical that the absence of an event be explicitly specified. The query interface must make this readily apparent either when users are specifying their query, or reviewing their results, or ideally, both.

Difficulty 3: Accessing “Does Not Occur” Relationships

Once users understood the need to explicitly specify an absence, they typically had very little difficulty integrating simple absence scenarios into meaningful queries. However, they frequently overlooked the fact that specifying the absence of individual events does not encompass the full range of possible “does not occur” queries. Consider a dataset of hospital transfers where the typical pattern of events is that the patient is admitted into the emergency room, then transferred to a specialty ward like neurology, then discharged (Figure 5).



Figure 5: Typical patient transfer sequence.

A researcher might be interested in patients who deviated from this pattern. Users would frequently attempt to answer this type of question using multiple queries involving different permutations of events or the absence of one or more of the individual events, such as in Figures 6 and 7.



Figure 6: Different permutation of typical sequence.



Figure 7: Event absent from typical sequence.

However, what they are really interested in is patient records in which the original sequence *does not occur*. Thus, the simpler approach is to search for the original sequence, and then select all the records that do not match that search. We found that users have difficulty transitioning to the mindset of using non-matches to their advantage. The query interface can play an integral role in whether this thought process shift is successful.

Difficulty 4: Understanding the Logic of Absences

From the onset, specifying the absence of an event cannot help but feel counterintuitive. This is because the absence of a point event functions much like an interval, while the absence of an interval event can function much like a point. Consider first, the absence of a point event. It might be important for a researcher to find all patients who did *not* have a Stroke after taking Drug A. A Stroke is a point event: it occurs at a single point in time. However, it does not make sense to search for the absence of a Stroke at any single point in time. There will, of course, be many points in time when the patient is not having a Stroke. The implication of this query is that a Stroke does not occur during the *entire* interval of time following the Drug A prescription. That is, the absence of a Stroke implicitly starts when the patient stops taking Drug A, and extends to the end of the patient record (Figure 8).

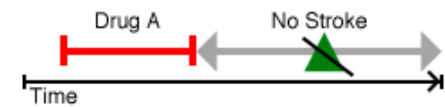


Figure 8: The implied end-points of a point event absence.

Counterintuitive? Yes, but users correctly report that this is how they expect the absence of a point event to behave, because this is the only way that the absence of a point event *can* behave. A point event absence is a duration of time, implicitly framed by the events specified on either side of it. Regardless of this, most users expressed this absence without a duration (Figure 9).

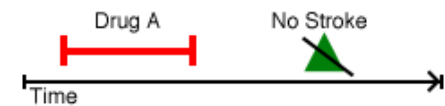


Figure 9: User specification of point event absence.

The problem with this specification is that, unlike the absence of a point event, the absence of an interval *can* be specified at a single point in time. For example, a researcher might be interested in patients who stopped taking their medication at any point after having a stroke. In this case, there need only be a single point in time when the interval is not taking place, such as in Figure 10.

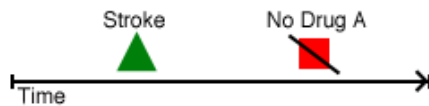


Figure 10: The absence of an interval, occurring at a single point.

The absence of an interval can also occur over a duration of time with implicit end points, just like a point event absence. For example, a researcher might be looking for patients who had a Stroke and then never took Drug A (Figure 11).

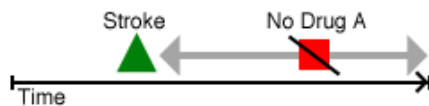


Figure 11: Interval absences can also have implied end-points.

Finally, the absence of an interval may only be relevant in the context of another event. For example, a researcher might be interested in patients who had a stroke when they were not taking Drug A (Figure 12).

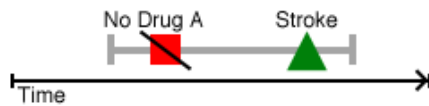


Figure 12: The absence of an interval occurs only during a specific event.

Thus, in order to distinguish the type of interval absence that is being specified, the interface must clearly represent the duration, or lack thereof, of all absence specifications. It is not sufficient to represent the absence of a point event as simply a slashed icon, as this representation is used for the absence of an interval that occurs at a single point. Users frequently reported that they had not considered, or were unaware of these different incarnations of an absence. The interface then, is responsible for helping them generate a query that conveys the absence that they intend to capture. No previous work in either temporal logic or temporal querying has explicitly addressed this logical inconsistency that arises with absences, or the effect that it has on query construction.

Design Study

To determine how a graphical query language might help to eliminate these difficulties, we conducted a generative design study - the first phase of our two-part user study - in which 20 computer science graduate students were asked to draw out (with colored markers on paper) a series of temporal queries. The participants, all of whom self-rated as expert computer users, but with limited experience with databases, command-based query formulation, and temporal event data, were first

introduced to the basic event representation in LifeFlow (see the Timeline panel in Figure 14). They were then given a blank timeline and asked to draw out the following event sequences:

1. The patient is Admitted, then begins taking Drug A. The patient has a Stroke while taking Drug A.
2. The patient takes Drug A for at least 3 days.
3. The patient has a Stroke, then starts taking Drugs A within 5-10 days after having the Stroke.
4. The patient has a Stroke, then is Admitted.
5. The patient has a Stroke, then is Admitted, WITHOUT being Diagnosed in between.
6. The patient is Admitted, then has a Stroke. The patient is NOT taking Drug A when the stroke occurs.
7. The patient is not taking Drug A at some point before having a Stroke.

These queries were explicitly devised to target the problem areas that we encountered during our initial tests and interviews, in order to determine how users graphically distinguish between scenarios when they are made explicitly distinct. For example, it is impossible to express the absence of a Diagnosis in Query #5 as a non-presence without the query being identical to Query #4. The freehand graphics generated by the participants, were used to inform the visual query language used in the advanced query interface (discussed in detail in the Advanced Search section). Participants were also asked to explain their design choices during a subsequent debriefing.

BASIC SEARCH (MENU-BASED)

Our development process began with an interface for basic temporal search. The primary goal of this interface was to give users quick and easy access to three of the most fundamental temporal relationships: before, after, and during. The basic search is also designed to serve as an introduction to more complex temporal relationships. Use of this interface allows users to explore both interval events and absences without the complexity of additional temporal constraints. As such, this interface was integrated into the main control panel of the LifeFlow application. It is described here, in the context of Hearst's search model:

Specification

The basic search interface consists of two separate modules: the subsequence module and the overlap module. Both modules consist of a simple list of drop-down menus which allow users to specify a series of event types (Figure 14). In each menu, users can select either the presence (denoted by a either a triangle or square icon for point and interval events respectively) or the absence (denoted by a slashed icon) of any event type. For interval events, the menus only allow users to specify complete intervals. That is, users cannot specify only the start or the end of an interval. This prevents users from trying to access "during" relationships using a sequential strategy (Difficulty #1).

Additionally, since the absence option appears next to its corresponding presence option in each menu, users are introduced to the concept of explicitly specifying absences by sim-

ply opening the menus (Difficulty #2). To remain consistent across both point and interval absences, all absences are handled uniformly, using implicit end points, since that is the default for point absences (Difficulty #4). The other interval absence options can be accessed in both the overlap query module and the advanced query interface.

```

select distinct t1.patid ,
               t1.drug ,
               t1.dispense_date ,
               t1.next_drug ,
               t1.next_dispense_date
from (select distinct patid ,
                    dispense_date ,
                    lead(dispense_date ,1) over (order by
                                                  patid , dispense_date , drug)
                    next_dispense_date ,
                    drug ,
                    lead(drug ,1) over (order by patid ,
                                               dispense_date , drug) next_drug
from DRUG_TBL
where drug in('DRUG A' , 'DRUG B')) t1 ,
EVENT t2
where t1.patid = t2.patid and
      t2.ICD9 = 'STROKE' and
      ((t1.drug = 'DRUG A' and t1.next_drug = '
DRUG B') and
       (t1.dispense_date = t1.
next_dispense_date) and
       (t1.next_dispense_date < t2.
event_start or t1.
dispense_date > t2.event_end));

```

Figure 13: SQL specification for patients that did not have a Stroke while taking Drug A and Drug B (the same query being executed on the right side of Figure 14).

The subsequence module targets before and after relationships in temporal event data. That is, it searches for disjoint, sequential events. For example, a researcher might be looking for records in which the patient took Drug A, followed by Drug B, and did not have a Stroke between these two drugs. This query, specified using the subsequence module is shown on the left side of Figure 14.

By contrast, the overlap module targets “during” relationships, or events that are happening concurrently. The right side of Figure 14 depicts an overlap search for records in which the patient does not have a Stroke while taking both Drug A and Drug B. As opposed to the subsequence module, which can match events spanning across a patient’s entire record, the overlap module searches only for a single point in that record when the specified events are occurring concurrently. Again, by having two separate modules with specialized functionality, users must mentally shift their search strategies, based on the event relationships they are accessing.

To illustrate the complexity of this seemingly simple overlap query, our partners at Oracle constructed the same query using SQL (Figure 13). The overlap query module distills this entire block of code into three simple menu specifications.

Presentation of Results

As mentioned previously, the basic search interface is integrated into the LifeFlow visualization tool, which consists of two different panels for visualizing a set of temporal event records. One panel depicts a listing of each individual record in the dataset, laid out across its own timeline. A second panel depicts an aggregated view of the entire dataset, where records with the same event sequences are grouped together in a summarizing display. A full description of the LifeFlow application and display can be found in [25]. For the purposes of this paper, we only depict the individual record panel.

When queries are specified using either the subsequence or overlap query module, matching records are selected and moved to the top of the individual display panel (Figure 14). This allows users to quickly see not only the records that were matched to their query, but also the records that did not match. A count of the number of selected records is displayed at the top of the control panel. While there is no explicit direction towards leveraging non-matches into effective queries, this strategy of keeping both the matches and non-matches visible at least prevents them from being forgotten altogether. If users identify the non-matches as the primary point of interest, they can invert the selected records (thus selecting all the non-matches) in a single click (Difficulty #3).

Reformulation

In the basic search interface, queries are reformulated by simply reconfiguring the selected items in the drop-down menus of the appropriate module. The configuration from the initial query remains in place unless the module is explicitly reset. Users can quickly return to their original query in order to make the necessary adjustments. Users also have the option of removing either matches or non-matches from the display before they specify a new search. Thus, their view of the dataset can be narrowed down as their exploration progresses.

ADVANCED SEARCH (GRAPHIC-BASED)

While the basic search interface gives users easy access to either before and after relationships (Subsequence module) or during relationships (Overlap module), the advanced search allows them to specify these relationships in tandem, as well as access more complex temporal features such as absolute time constraints and the full range of absence scenarios. The advanced search interface revolves around a visual query language that is used to draw the desired sequence of event relationships. Wongsuphasawat et al. [26] found that users prefer this graphical method over the menu-based interface for specifying complex temporal relationships.

We began with an interface closely modeled after the search interface used in Similan [26], which we augmented to include an initial implementation of both absences and intervals. The interface was then updated incrementally with improvements gleaned from both the hand-drawn, generative phase of the user study and a second, *usability phase*:

In this phase, our 20 participants were asked to construct the same set of seven queries using the advanced query interface. We observed both the users’ understanding of the graphics and their interactions with the interface, loosely following

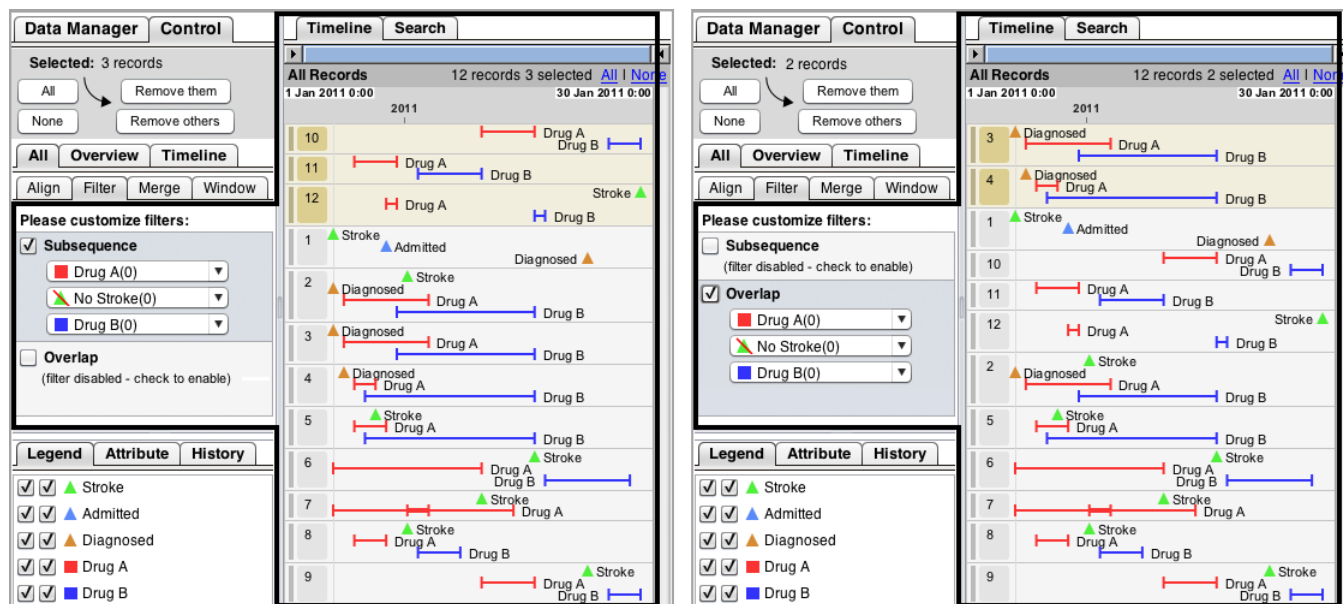


Figure 14: Basic Search: The Subsequence module (left) targets before and after relationships. The Overlap module (right) targets during relationships. Query results, with matching records highlighted at the top of the list, are displayed in the Timeline panel on the righthand side of both figures.

the Production-Preference-Performace program for developing graphical representations of abstract concepts [15]. Our findings are discussed throughout the remainder of this section.

Specification

Queries are specified by adding query elements to a designated canvas. Users click on the canvas to bring up the Element Editor Panel (Figure 15), where they can specify the element type, whether or not it occurs (Difficulty #2), and any additional time constraints. The hand-drawing phase of our user study greatly informed the graphics used in the advanced query interface. We initially displayed absences as individual gray boxes, interspersed between the presence events. However, multiple participants designated a separate section of the canvas for displaying absences, a technique that was very effective for keeping the display simple. This idea was quickly integrated into the evolving design. Additionally, interval events were initially displayed as double-sided arrows. However, many users felt that the pointed ends expressed the continuation of the interval, rather than its end-points. The interval graphic was thus changed to be a connected pair of bars. Similar to the concept behind the basic query interface, when users add intervals to their query, a fully-formed interval is added to the canvas. That is, users cannot add only an interval start-point or an interval end-point to their query (Difficulty #1).

Participants had no difficulty using the advanced query interface to specify sequences of presence events, including more complex temporal constraints such as interval durations and gaps between events. Our initial implementation of absence specification, however, involved a series of options that allowed users to specify whether the absence should have implicit end-points, occur at a single point in time, or apply only to a specified presence event. Users had difficulty understand-

ing these options, regardless of the language used in the description. To mitigate this issue, we narrowed the absence specification form down to a single checkbox which controls whether the duration of the absence automatically spans out to the nearest events on either side. For interval absences, users can uncheck this box in order to have manual control of where the absence's end-points are placed (Difficulty #4).

Presentation of Results

Similar to the basic search result presentation, matching records are moved to the top of the record display. However, the advanced search explicitly separates matches and non-matches into two separate lists, and ranks records in each list by how closely they match the query (Difficulty #3). Matches are ranked by the number of extra events in the record (events that were not matched to an element in the query). Non-matches are ranked by how many query elements were matched within the record. A full display of the advanced query interface can be found in Figure 16.

These separate, ranked lists allow users to quickly find the most likely false positives (which will be towards the bottom of the matched list) and false negatives (which will be towards the top of the non-matched list). None of the participants in our study had any difficulty evaluating whether or not their query had returned correct results, indicating that the result presentation conveys enough information to accurately identify errors.

Reformulation

The advanced search interface makes it extremely easy for users to modify their queries. Elements on the query canvas can be dragged into new positions or deleted. Users can also click on any element to bring up its Element Editor, or hover over the element to view the event details. The only difficulty that participants encountered in the Reformulation stage was

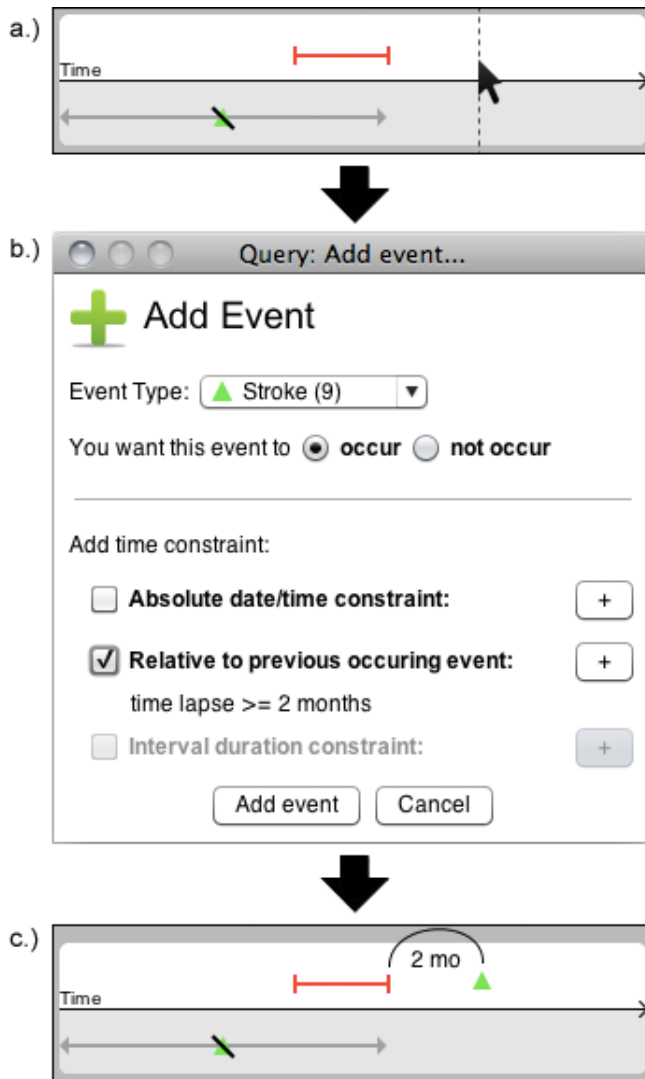


Figure 15: Specifying a query: Users add query elements by clicking the desired spot on the canvas (a), and adjusting its features using a pop-up form (b). The element is then added to the canvas, where it can be dragged, deleted, or modified (c). Event details can be seen on hover.

that, if they missed the element they were trying to click on and clicked on the background instead, the form to add a new query element would appear. This confused users, who were expecting to see the form to modify an existing element. This problem was solved by making the icons slightly larger and adding a more visible hover effect.

CASE STUDIES

Once the development of the basic and advanced search interfaces was complete, we spent one-on-one time with two different epidemiologists at the PVC, using LifeFlow and both temporal query interfaces to explore the evolving questions of their on-going research studies.

Opioid Use

The first epidemiologist was investigating patterns of opioid use. Opioids are typically prescribed to treat pain, and range in strength from low, over-the-counter doses, to high,

potentially-habit-forming prescriptions. Additionally, these medications can be prescribed as short-term, “acute” treatments, or long-term, “chronic” treatments. The epidemiologist was interested in the relationships between the dosage strength and the treatment duration. For example, are low doses of opioids being prescribed for acute treatments?

Her dataset was organized into two layers. The first layer consisted of each patient’s individual opioid prescriptions, categorized as either a “high,” “medium,” or “low” dose. In the second layer, individual prescriptions in close proximity to one another were combined into “eras,” and categorized as either an acute or chronic treatment, based on the duration. Thus, each prescription was represented twice, once as an individual dosage, and once as a component of a treatment. Hospital visits were also included in the dataset as potential catalysts for treatment changes. Overall, the dataset consisted of 500 patients and 2171 events.

The patients that the epidemiologist was initially interested in were those who received a high opioid dosage as part of an acute treatment. To isolate these patients, she used the overlap search module to select patients with concurrent “high” and “acute” intervals. From there, she was interested in cases where this combination was not closely preceded by a hospital visit. It was not readily apparent to her how to construct this query (Difficulty #3). We guided her to construct a query for cases where this combination *was* closely preceded by a hospital visit. From this hint, she quickly realized that the records not matching this query would satisfy her original query. She was able to isolate these records without further assistance.

Asthma Medication Prescribing Practices

The second epidemiologist was working to understand the prescribing practices of asthma medications. Much like opioids, asthma medications range broadly in the strength of the prescription. One of the strongest classes of asthma medications, the Long-Acted Beta Agonist (LABA), should only be prescribed when other, weaker asthma medications have proven ineffective. Furthermore, when a LABA is prescribed, it should be followed by multiple prescriptions of successively weaker asthma medications, known as the “step-down.”

One of the queries that the epidemiologist constructed to explore these prescribing practices was to use the advanced search to look for patients who were prescribed a weaker asthma medication within 3 months of both the start and end date of the LABA prescription. However, this query also returned patients who had been prescribed a LABA either before or after the query sequence (Difficulty #2). The epidemiologist immediately noticed this error in the results display, and modified the query to include the specification that there were no LABA prescriptions before the initial, weaker prescription, or after the final prescription. The final query, and the results it produced, can be seen in Figure 16.

In both of these case studies, the epidemiologists remarked that LifeFlow, and both the basic and the advanced query interface were much easier to learn and use than the command-

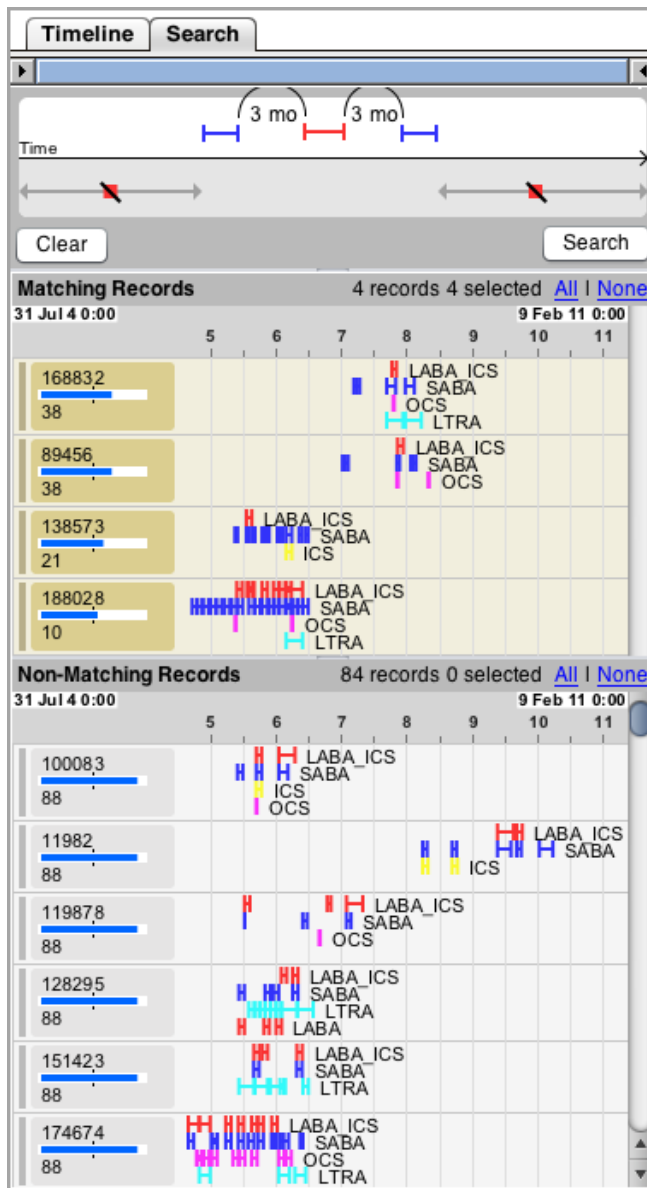


Figure 16: The Advanced Query Interface, including the query canvas (top), matching records (middle), and non-matching records (bottom). In this query, the researcher was looking for patients who received a weaker asthma medication (in blue) within 3 months of both the start and end date of a stronger asthma medication (in red). They also wanted to ensure that this sequence was neither preceded nor followed by the strong medication.

based, statistical software that they had been using previously. Both of our query interfaces prevented the epidemiologists from constructing queries in which intervals were treated as disjoint point events (Difficulty #1). However, we are still working to fully determine whether the interfaces are effective in distinguishing between different absence scenarios (Difficulty #4), as we have yet to encounter the full range of such scenarios within a single case study. At the prompting of these epidemiologists, we are currently working to install LifeFlow on the Pentagon servers to facilitate their use of the software. While this enthusiasm is encouraging, particu-

larly from such experienced researchers, more careful testing and long-term assessments are needed to understand when our querying interfaces are helpful compared to other tools, and what additional features are needed.

CONCLUSION

In this paper, we introduced two novel interfaces for constructing and executing temporal queries involving intervals and the absence of an event. These interfaces facilitate temporal search in four primary ways that set them apart from standard, command-based querying techniques:

1. Easy to learn and use.
2. Guide users away from common difficulties.
3. Offer access to a wide range of temporal relationships.
4. Address all three stages of Hearst's search model.

We illustrated the use of both interfaces in two case studies with epidemiologists at the US Army Pharmacovigilance Center, who used the query interfaces, as well as the encapsulating LifeFlow visualization tool, to answer questions involving prescribing practices and medication interactions. In these sessions, the epidemiologists were able to answer their evolving research questions using both the basic and advanced query interfaces. They described the entire query process as a dramatically simpler alternative to the command-based approach to which they had been previously limited. Though we only reported on these two case studies, multiple other studies are underway with researchers across a wide range of both clinical and non-clinical domains.

Moving forward, we plan next to integrate more flexibility into the advanced query interface, allowing users to construct queries where certain events can appear in permuted orderings. We are also testing different, more granular ways of classifying the search results, much like the work done by Chen and Dumais [3]. As we continue our collaboration with all of our case study partners, our hope is that their datasets and research questions will guide the development of future feature sets.

ACKNOWLEDGMENTS

We appreciate the partial support of the Oracle Corporation and NIH-National Cancer Institute grant RC1-CA147489, Interactive Exploration of Temporal Patterns in Electronic Health Records. We appreciate the collaboration of researchers at the US Army Pharmacovigilance Center and the Univ. of Maryland Human-Computer Interaction Lab.

REFERENCES

1. Allen, J. F., and Ferguson, G. Actions and events in interval temporal logic. In *Journal of Logic and Computation*, vol. 4 (1994), 531–579.
2. Bates, M. The design of browsing and berrypicking techniques for the on-line search interface. In *Online Review*, vol. 13 (1989), 407431.
3. Chen, H., and Dumais, S. Bringing order to the web: automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human*

- factors in computing systems*, CHI '00, ACM (New York, NY, USA, 2000), 145–152.
4. Chittaro, L., and Combi, C. Visualizing queries on databases of temporal histories: new metaphors and their evaluation. In *Data & Knowledge Engineering - Special issue: Temporal representation and reasoning*, vol. 44 (2003), 239–264.
 5. Combi, C., and Oliboni, B. Visually defining and querying consistent multi-granular clinical temporal abstractions. In *Artificial intelligence in medicine*, vol. 54 (2012), 75–101.
 6. Cruz, I. F. Doodle: a visual language for object-oriented databases. In *Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, SIGMOD '92, ACM (New York, NY, USA, 1992), 71–80.
 7. Das, A. K., and Musen, M. A. A temporal query system for protocol-directed decision support. In *Methods of information in medicine*, vol. 33 (1994), 358–70.
 8. Dunn, J., Davey, S., Descour, A., and Snodgrass, R. T. Sequenced subset operators: Definition and implementation. In *IEEE International Conference on Data Engineering (ICDE2002)*, IEEE (2002), 81–92.
 9. Fails, J. A., Karlson, A., Shahamat, L., and Shneiderman, B. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, IEEE (2006), 167–174.
 10. Hearst, M. A. *Search User Interfaces*. Cambridge University Press, 2009.
 11. Hibino, S., and Rundensteiner, E. A. User interface evaluation of a direct manipulation temporal visual query language. In *MULTIMEDIA '97 Proceedings of the fifth ACM international conference on Multimedia*, ACM (1997), 99–107.
 12. Jensen, C. S., Cliord, J., and et al., S. K. G. The tsql benchmark. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases* (1993), QQ1–QQ28.
 13. Jin, J., and Szekely, P. Interactive querying of temporal data using a comic strip metaphor. In *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, IEEE (2010), 163–170.
 14. Kam, P.-S., and Fu, A. W.-C. Discovering temporal patterns for interval-based events. In *DaWaK 2000 Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery*, Springer-Verlag (2000), 317–326.
 15. Kessell, A., and Tversky, B. Visualizing space, time, and agents: Production, performance, and preference. In *Cognitive Processing 12* (2011), 43–52.
 16. Li, W.-S., Candan, K., Hirata, K., and Hara, Y. Ifq: a visual query interface and query generator for object-based media retrieval. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems* (jun 1997), 353–361.
 17. Lin, J., Thomsen, M., and Landay, J. A. A visual language for sketching large and complex interactive designs. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, CHI '02, ACM (New York, NY, USA, 2002), 307–314.
 18. Marchionini, G., and White, R. Find what you need, understand what you find. In *Journal of Human-Computer Interaction*, vol. 23 (2008), 205237.
 19. Patel, D., Hsu, W., and Lee, M. L. Mining relationships among interval-based events for classification. In *SIGMOD '08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM (2008), 393–404.
 20. Ryall, K., Lesh, N., Lanning, T., Leigh, D., Miyashita, H., and Makino, S. Querylines: approximate query for visual browsing. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, ACM (New York, NY, USA, 2005), 1765–1768.
 21. Snodgrass, R. The temporal query language tqel. In *ACM Transactions on Database Systems (TODS)*, vol. 12, ACM (1987), 247–298.
 22. Sutcliffe, A., and Ennis, M. Towards a cognitive theory of information retrieval. In *Interacting with Computers*, vol. 10 (1998), 321351.
 23. Vrotsou, K., Johansson, J., and Cooper, M. Activitree: Interactive visual exploration of sequences in event-based data using graph similarity. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 15 (2000), 945–952.
 24. Wang, T. D., Deshpande, A., and Shneiderman, B. A temporal pattern search algorithm for personal history event visualization. In *IEEE Transactions on Knowledge and Data Engineering* (2012), 799–812.
 25. Wongsuphasawat, K., Gómez, J. A. G., Plaisant, C., Wang, T. D., Taieb-Maimon, M., and Shneiderman, B. Lifeflow: Visualizing an overview of event sequences. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI'11)*, ACM (2011), 1747–1756.
 26. Wongsuphasawat, K., Plaisant, C., Taieb-Maimon, M., and Shneiderman, B. Querying event sequences by exact match or similarity search: Design and empirical evaluation. In *Interacting with Computers*, vol. 24 (2012), 55–68.
 27. Wu, S.-Y., and Chen, Y.-L. Mining nonambiguous temporal patterns for interval-based events. In *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, IEEE (2007), 742–758.