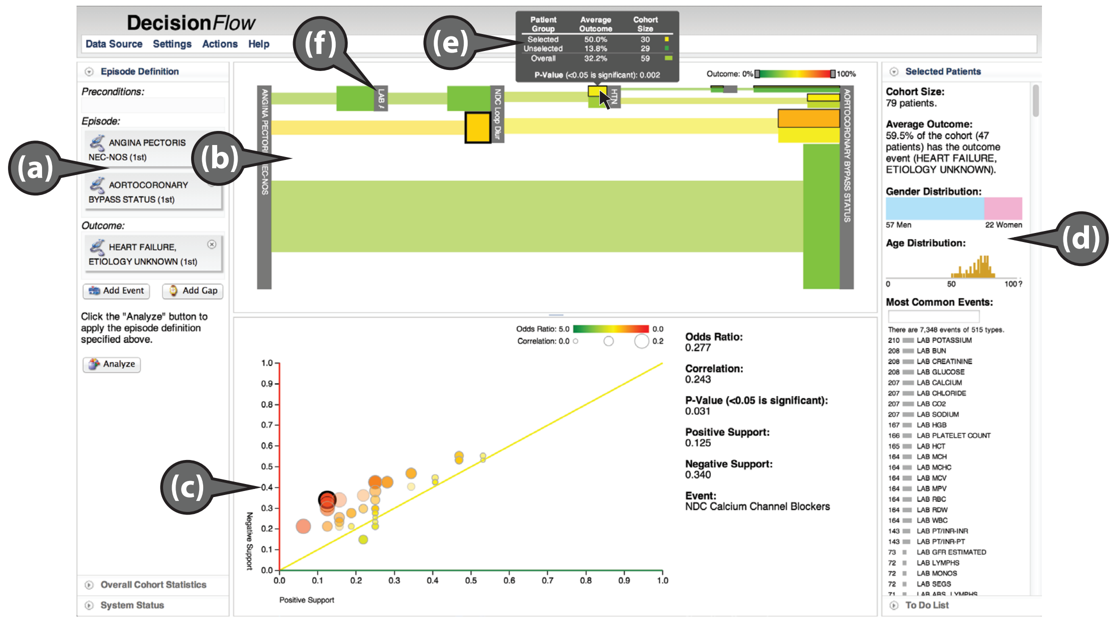
DecisionFlow: Visual Analytics for High-Dimensional Temporal Event Sequence Data

David Gotz and Harry Stavropoulos



Patient Group	Average Outcome	Cohort Size	
Selected	50.0%	30	🟨
Unselected	13.8%	29	🟩
Overall	32.2%	59	🟩

P-Value (<0.05 is significant): 0.002

Click the "Analyze" button to apply the episode definition specified above.

Odds Ratio: 0.277
Correlation: 0.243
P-Value (<0.05 is significant): 0.031
Positive Support: 0.125
Negative Support: 0.340
Event: NDC Calcium Channel Blockers

Cohort Size: 79 patients.
Average Outcome: 59.5% of the cohort (47 patients) has the outcome event (HEART FAILURE, ETIOLOGY UNKNOWN).
Gender Distribution: 57 Men 22 Women
Age Distribution:
Most Common Events:
There are 7,348 events of 515 types.

210	LAB POTASSIUM
208	LAB BUN
208	LAB CREATININE
208	LAB GLUCOSE
207	LAB CALCIUM
207	LAB CHLORIDE
207	LAB CO2
207	LAB SODIUM
167	LAB HGB
166	LAB PLATELET COUNT
165	LAB HCT
164	LAB MCH
164	LAB MCHC
164	LAB MCV
164	LAB MPV
164	LAB RBC
164	LAB RDW
164	LAB WBC
143	LAB PT/INR-INR
143	LAB PT/INR-PT
73	LAB GFR ESTIMATED
72	LAB LYMPHS
72	LAB MONOS
72	LAB SEGS
71	LAB ABS. LYMPHS

Fig. 1. A screenshot of DecisionFlow being used to analyze electronic medical data. This view (b-d) summarizes the medical records for 514 cardiology patients that match (a) the user-defined query. The query results include over 113,000 individual events of more than 1,600 distinct event types. (e) Highlighted is a subgroup of patients who developed heart failure at significantly higher rates ($p < 0.01$) than peers who received (f) a particular type of lab test earlier in the episode.

Abstract— Temporal event sequence data is increasingly commonplace, with applications ranging from electronic medical records to financial transactions to social media activity. Previously developed techniques have focused on low-dimensional datasets (e.g., with less than 20 distinct event types). Real-world datasets are often far more complex. This paper describes DecisionFlow, a visual analysis technique designed to support the analysis of high-dimensional temporal event sequence data (e.g., thousands of event types). DecisionFlow combines a scalable and dynamic temporal event data structure with interactive multi-view visualizations and ad hoc statistical analytics. We provide a detailed review of our methods, and present the results from a 12-person user study. The study results demonstrate that DecisionFlow enables the quick and accurate completion of a range of sequence analysis tasks for datasets containing thousands of event types and millions of individual events.

Index Terms—Information Visualization, Temporal Event Sequences, Visual Analytics, Flow Diagrams, Medical Informatics

1 Introduction

Temporal event data is nearly ubiquitous in this era of mobile devices, electronic communication, and sensor networks. It can be found in everything from social network activity, to financial transactions, to electronic health records. More than ever before, large collections of this sort of data are being recorded that capture (a) what type of event is happening, (b) when it happens, and (c) the entities (e.g., customer, bank account, or patient) involved.

Often, such temporal events are grouped by entity and chained together to form sequences sorted by order of occurrence. The resulting sequences provide a sampled record of what has happened to the corresponding entity over time. For example, in a patient's electronic medical record, we see a time-stamped sequence of diagnoses, treatments, medications, and lab tests.

Given the wide availability of such data sets and the valuable insights they potentially contain, a number of recent efforts have explored a range of visual analytic methods specifically designed for temporal event sequence data. These techniques typically begin by building an aggregate data structure that captures statistics (such as frequency and timing) for each of the various permutations of the event sequences found in the original data set. This structure is subsequently visualized, often using flow-based techniques reminiscent of Sankey diagrams [10, 19, 31, 32].

This aggregation approach can effectively handle large numbers of entities, and has proven powerful in a wide range of applications. It is, however, critically constrained in that it is only effective for small numbers of event types. For example, LifeFlow [32] uses colors to distinguish between event types and is shown being used with just six event types. Outflow [31], while avoiding color-coding, still has trouble as the number of event types grows (e.g., to over 20), causing the visualization to break down into a complex web of event pathways.

• *David Gotz is with the University of North Carolina at Chapel Hill. E-mail: gotz@unc.edu.*
• *Harry Stavropoulos is with the IBM T.J. Watson Research Center. E-mail: hstavrop@us.ibm.com.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014.
Digital Object Identifier 10.1109/TVCG.2014.2346682

Inconveniently, real-world data sets often contain a large variety of event types. For example, medical data can have thousands of event types, given the complex coding systems used for diagnoses, treatments, medications, and lab tests. For such data sets, which we call *high-dimensional*, users are often forced to work around this limitation by either pre-filtering to a small group of event types (e.g., a set of 18 specific heart failure symptoms [10]), or collapsing multiple low-level event types into a small number of high-level categories (e.g., by grouping specific drugs into more generic medication classes [20]).

Recognizing this limitation, recent attempts have applied simplification algorithms to the high-dimensional problem. Approaches include event filtering, graph simplification [31], and pattern-based substitution [17]. These methods address the challenge by first presenting users with a complex, full-resolution visualization. They then provide interaction-driven techniques to help users simplify the high-dimensional type space. As a result, while they enable the effective analysis for marginally more complex data sets, they do little to address the fundamental problem: they do not allow for the visual analysis of data sets with thousands - or more - distinct event types because the initial starting condition is too complex.

In this paper, we present an alternative approach called *DecisionFlow*, that directly supports the visual analysis of high-dimensional temporal event sequence data with orders-of-magnitude more event types than previously achieved. For example, our user study shows that our prototype is effective with over 3,500 unique event types. Our scalable approach combines an incremental milestone-based data representation with on-demand statistical analysis and an exploratory flow-based visualization technique. Beginning with a relatively simple milestone-based view, direct interaction capabilities let users explore the high-dimensional event type space, test hypotheses, and make statistical comparisons between complex event pathways.

This paper describes the DecisionFlow technique in more detail, including our milestone-based aggregate data structure and corresponding temporal query methods. We also review our basic visualization technique and a set of incremental data manipulation operations provided to enable users to perform exploratory analyses. We then review a number of additional interaction capabilities and present the results from a user study documenting the benefits of our approach in an application from the healthcare domain. The key research contributions presented are:

- A dynamic, milestone-based approach to temporal event sequence query, aggregation, and visualization that scales effectively for datasets with large numbers of event types. Included are user interaction methods for visual exploration and comparison.
- A scalable data representation and an associated set of interactive *operators* that support ad hoc milestone-based pathway analyses. Associated with these operators are a set of algorithms for analysis and data structure manipulation.
- Results and discussion from a user study evaluating human performance with DecisionFlow on a set of event sequence analysis tasks.

The remainder of this paper is organized as follows. Section 3 provides an overview of related work. Section 4 then describes the DecisionFlow visualization while Section 5 presents additional algorithmic details. Evaluation results are discussed last in Section 6. We then conclude the paper with a summary and brief discussion of future work.

2 Motivation: Health Outcomes Analysis

While event sequence data can be found in a variety of domains, our work is primarily motivated by health outcomes research. In this field, analysts and epidemiologists study data from groups of patients to understand what factors may influence a particular outcome (e.g., readmission [3] or disease onset [25]). More specifically, such researchers perform a two stage process.

First, *selection* defines a cohort of patients that are to be included in a study. Selection often includes inclusion/exclusion criteria as well as an *episode* of interest (e.g., two years prior to diagnosis with heart disease). Selection defines a number of key concepts for a given study including which patients are included, the ”time zero” for the study (used to align patients around a cardinal event) and the period of time to study.

Second, *analysis* is performed over the data that meets the selection constraints. The results of this process inform the researcher as to which factors, clinical pathways, or other structures in the data are most associated with the outcome of interest.

Traditionally, the above steps are performed using structured query languages (e.g. SQL) and dedicated statistical software (e.g., SAS). However, this generally requires programming support and significant amounts of time, especially as data sets grow larger and more complex. This can make exploratory analysis extremely onerous, leading to an interest in more efficient and easy-to-use visual analytics tools.

3 Background

The visualization and analysis of temporal event data is a widely studied topic. Areas most relevant to the work presented here include temporal data representations and query models, event sequence visualization methods, and event sequence simplification/substitution techniques.

3.1 Temporal Data Representations and Query Models

Given the ubiquitous nature of temporal data and the complexity of the concept of time, a number of projects have explored the development of efficient visual data representations. The work of Rind et. al is a recent example of such work [23].

Many of these efforts have explored temporal models designed to support visual time-based query capabilities. These representations generally include key milestone events (e.g., temporal query constraints) which must be matched for an event sequence to satisfy a query, along with ways to specify intervals, absences, and other temporal constructs [12, 13, 18].

Perhaps most relevant is our own recent work on milestone-based pattern mining [9]. In this work, milestones are entered by users as query constraints to retrieve matching event sequences. The milestones are then reused to partition the returned data into sub-sequences, which are subsequently mined for frequent patterns. Finally, the milestones are incorporated into a static visual representation that is used to select a portion of the pre-computed mined pattern data for display. This approach does not support exploratory ad hoc analysis. Rather, it is analagous to the “mine-then-visualize” approach taken by Wong et al. [29] for pattern discovery in text corpora.

In contrast, our approach provides an exploratory visual environment that lets users incrementally build and visualize complex temporal constructs. This is done via a dynamic data representation that can be manipulated interactively using a set of operators that modify the structure and content of the representation. Our method performs on-demand analyses of the data structure during its evolution in response to these user-initiated operations.

3.2 Temporal Event Sequence Visualizations

Due to the variety of temporal data types and applications, a vast number of techniques have been developed [1]. Most relevant to the work presented here are methods devoted to temporal event sequence data. Historically, much of the work has focused on visualizing a single record [2, 4, 5, 7, 11, 14, 22].

Expanding on this work, methods for searching, filtering, and grouping multiple event sequences (or subgroups within a single large sequence) were developed [6, 8, 26, 27, 28, 33, 34]. Most recently, other researchers have supported much larger numbers of event sequences by visually combining individual records into an aggregate view [10, 30, 31, 32]. In theory, these techniques scale well to large numbers of event sequences, as long as the number of distinct event types is small (e.g., less than 20). In contrast, the work presented here is suitable for both large numbers of event sequences and large numbers of event types (e.g, in the thousands).

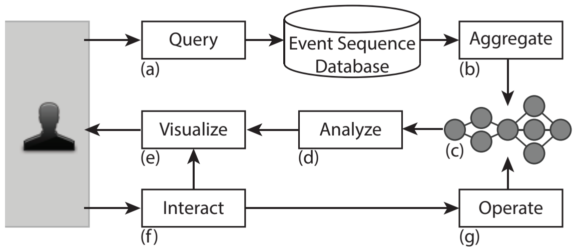


Fig. 2. DecisionFlow users first (a) issue a query to retrieve sub-sequences of interest. The matching data is then (b) aggregated to construct (c) a DecisionFlow Graph, G. G is then (d) analyzed to extract statistics and (e) visualized. (f) Interaction allows exploratory analysis. Certain interactions cause (g) transformation operations on G which, in turn, trigger ad hoc analyses over the underlying event data.

Variable	Description
t	The timestamp of an event
τ	The type of an event
$e = (\tau, t)$	An individual event
id	A unique event sequence identifier
$\vec{s} = (id, [e_0, e_1, \ldots, e_n])$	An event sequence of length n
$\vec{e} = (id, [e_i, e_{i+1}, \ldots, e_j])$	An episode, where $\vec{e} \subseteq \vec{s}$
$C = \{\vec{s}_0, \vec{s}_1, \ldots, \vec{s}_m\}$	A collection of event sequences
$G = (M, D)$	A DecisionFlow Graph
$m_i \in M$	A milestone node in G
$d_i \in D$	A directed edge in G

Table 1. A summary of the notation used throughout this paper.

3.3 Temporal Event Simplification and Substitution

Recognizing the need to scale to large numbers of event types, there has been an interest in representing data at higher levels of granularity, focusing on patterns rather than low-level events [16]. Addressing this need, a variety of research projects have developed methods for simplification or substitution. For example, people have explored aggregation using the inherent structure of time (e.g., by day, week, month or year) [21], as well as the use of kernel density estimation to approximate frequency over time [15].

Outflow [31] includes a simplification feature based on hierarchical clustering to reduce visual complexity. However, the utility of this approach is limited, due in part to structural limitations in that it can only combine events in the same layer. Extensions of the Outflow technique have used domain-specific type hierarchies to group semantically similar event types [20]. However, this approach requires that the groups be statically defined in advance.

A more flexible substitution-based approach was proposed by Monroe et al [17]. It allows users to define individual meta-events that could be substituted in the place of longer, common sub-sequences to reduce visual complexity. However, this method begins by visualizing the data in full detail, then slowly reducing the data's complexity. Our approach is the opposite: we begin with a simplified visualization and provide users with the tools to dynamically identify, insert, and remove salient events during an analysis.

4 Description of DecisionFlow

DecisionFlow consists of three principal elements: (1) a baseline event sequence data representation that supports query and aggregation, (2) a set of dynamic operators that manipulate the baseline representation, allowing the ad hoc creation of temporal pathways during exploratory analysis, and (3) visual encodings and interactions that support dynamic user exploration and statistical comparisons. These elements combine to support the workflow illustrated in Figure 2. This section describes these elements in detail, beginning with some key definitions.

4.1 Temporal Constructs

A number of key temporal constructs, summarized in Table 1, are used throughout this paper. These terms are defined as follows.

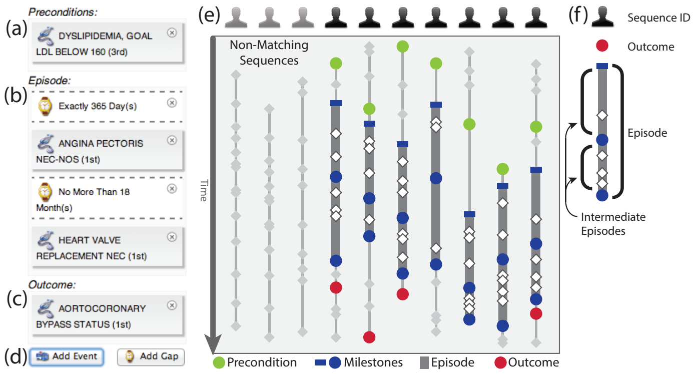


Fig. 3. Users specify a query by defining (a) preconditions, (b) milestones describing the episode of interest, and (c) an outcome measure. These are defined interactively through (d) dialog boxes and drag-and-drop interaction. Query execution (e) searches through C to find matching event sequences and, for each match, returns (f) the sequence ID, the outcome - if present - and all events that take place within the episode period.

Event Sequences. We define a *temporal event sequence* $\vec{s} = (id, [e_0, e_1, \ldots, e_n])$ as a unique identifier (id) and an ordered list of n distinct *events* $e = (\tau, t)$, where τ is the event's *type* and t is the *time* at which the event occurred. The events in s_i are ordered by time, such that $t_{e_j} \leq t_{e_{j+1}}$. For events in the same sequence that occur at the same time, we further sort by their types (i.e. $\tau_{e_j} \leq \tau_{e_{j+1}}$). This provides a total ordering for all events within $\vec{s}$. For example, one can treat a patient's electronic medical record as a temporal event sequence containing a unique time-sorted list of events from the patient's medical history (e.g., diagnoses, lab tests, medications, and treatments). Medical events occurring on the same day would be sorted by the type of event they represent (e.g., lab results first, followed by alphabetically sorted medications). Events are always represented as *point events* occurring at a single moment of time. Thus, *interval events* (i.e. those with a duration) are not represented. [1]

Episode. An *episode* $\vec{e}$ represents a specific portion of an overall event sequence (i.e., $\vec{e} \subseteq \vec{s}$). This construct is used to focus on a specific range of events (e.g., an episode in the medical record domain could be defined to include only events between a patient's initial admission to a hospital for surgery and her subsequent discharge).

Collection. A large number of individual temporal event sequences can be grouped to form a *collection*, $C = \{\vec{s}_0, \vec{s}_1, \ldots, \vec{s}_m\}$. The event sequences s_i can vary widely in both length (number of events) and content (the type and time for each individual event). For example, a hospital might maintain a collection of event sequences representing patient medical records. Within that collection, the number and types of diagnoses will likely vary widely from patient to patient.

4.2 Milestone-based Query, Aggregation, and Analysis

As illustrated in Figure 2, users begin by an analysis by querying the event sequence database to retrieve a set of episodes relevant to a given question. This is supported through a visual, milestone-base query capability. The episodes returned by the query are then transformed to build a data structure designed for interactive visualization.

4.2.1 Query

DecisionFlow employs a milestone-based query mechanism as illustrated in Figure 3. There are two parts in this component: (a) an interface widget allowing users to define the constraints of a query, and (b) a query engine that processes the query specification and retrieves matching event episodes from the overall event sequence collection.

The query widget allows the definition of three elements: the episode definition, its preconditions, and an outcome. Most significant

[1] For some applications, however, it is possible to transform interval events into pairs of independent 'start' and 'end' point events.

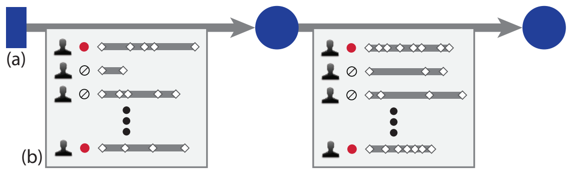


Fig. 4. (a) The initial aggregation process creates a graph G that includes a linear sequence of nodes (representing milestones) connected by directed edges that encode the order of occurrence. (b) Anchored to each edge are details describing the intermediate episodes that occur between the milestones.

is the episode definition. It is specified as an ordered list of *milestones*, each of which is associated with an event type and an ordinal (1st, 2nd, 3rd, etc.). The ordinal is used to allow users to specify which occurrence of an event type is associated with a milestone. In addition, *time gap constraints* can be added between milestones, or at the start of the episode definition, or at the end. When appearing at the start (or end), these gaps define *implicit milestones* corresponding to the start/end of the time window. Together, these elements define the query constraints that are used to specify which data is returned.

Figure 3(b) shows an example episode definition from a medical record use case. The user has created an episode definition that will match patients who have had a diagnosis of *Angina Pectoris*, followed by a *Heart Valve Replacement* within 18 months. The time gap of 365 days at the start of the episode definition (before *Angina Pectoris*) indicates that the user is interested in events happening up to one year in advance of the *first Angina Pectoris* diagnosis.

The query widget also allows for the specification of one or more *preconditions*. Like milestones, preconditions are defined by a type and an ordinal; they represent events that must occur within $\vec{s}$ prior to the start of the episode in order for the sequence to match the query. For example, Figure 3(a) indicates that only patients who have at least three diagnoses of *Dyslipidemia* should be returned by the query.

Finally, users are asked to define an *outcome* measure which is defined in the same ways as a milestone: an event type and an ordinal. Event sequences that have the outcome event after the end of the episode are so labeled. Otherwise, episodes are assigned an outcome of "not present." An example is shown in Figure 3(c).

Users can add new events or time gaps to these definitions via the controls located at the bottom of panel (see Figure 3d). Drag-and-drop capabilities let users quickly re-order milestones and time gaps, or move them between portions of the query panel.

Once the query has been specified, a query engine executes the constraints against C to retrieve a set of matching episodes. This is illustrated in Figure 3(e). As the figure shows, several $\vec{s} \in C$ are excluded because they don't match the precondition or episode definitions. For those $\vec{s}$ that match the query constraints, the query engine determines the episode $\vec{e} \subseteq \vec{s}$ that contains all events between - and including - the first and last milestones in the episode definition. This is indicated by the gray bars in the figure, which begin at the first implicit "start of time window" milestone (the blue rectangle) and end at the last *Heart Valve Replacement* milestone. The query engine returns a set of matching episodes, each of which is associated with both a sequence ID and an outcome (see Figure 3(f)).

4.2.2 Aggregation

After the query engine returns a set of episodes that match the user's constraints, an aggregation process transforms the individual lists of events into a data structure designed for scalable, interactive visualization. This aggregate structure is called the *DecisionFlow Graph*, which we note as G.

By definition, every episode returned by a given query contains the same set of milestones. The aggregation process begins by creating a simple directed graph with one node per milestone (which we note as $m_i \in M$). The milestones are linked with directed edges (noted as $d_i \in D$) which capture the order in which they occur. An example initial graph $G = (M, D)$, using blue milestone markers and gray arrows, is illustrated in Figure 4(a).

Next, each of the individual episodes returned is split into a set of *intermediate episodes* (see Figure 3(f)). Each intermediate episode contains the events occurring between a pair of milestones (exclusive of the milestones themselves). These intermediate episodes are then stored within the corresponding edge d_i (Figure 4(b)). The final step in the initial construction of G, performed after all episodes have been segmented and indexed within G, is the execution of edge analytics, described below.

4.2.3 Edge Analytics

Following the construction of G, two groups of statistics are computed for each $d_i \in D$. First, *edge summary statistics* are calculated, including average intermediate episode duration and the outcome rate (the fraction of intermediate episodes for which the outcome measure is present). Other statistics include age and gender distributions. Many of these statistics (e.g., age/gender distributions, outcome rate) are identical for all edges in G at the time of initial construction, as they are not time dependent. However, these measures can vary dramatically between edges once G is transformed by the operators described in Section 4.3.

Second, and most critical for DecisionFlow's approach to high-dimensional data, are *edge event statistics*. These are computed independently for each edge in G as a function of the intermediate episode data anchored to that edge (see Figure 4b). We separate the intermediate episodes for each edge into two groups: a *positive outcome* group and a *negative outcome* group. Then, for every event type observed at least once in the intermediate episode data for a given edge, we compute the following:

- **Event frequency:** the total number of times the event occurs across all intermediate episodes associated with the edge.
- **Positive support:** the fraction of intermediate episodes in the positive outcome group containing one or more occurrences of the event type.
- **Negative support:** the fraction of intermediate episodes in the negative outcome group containing one or more occurrences of the event type.
- **Correlation and odds ratio:** computed by comparing the positive and negative outcome groups, these measure the association between individual event types and the outcome measure; p-values are also computed.

As illustrated in Figure 2(d), the edge analysis process is first performed after the initial construction of G, and is repeated every time G is modified by one of the operations defined below.

4.3 Dynamic Manipulation of the DecisionFlow Graph

Upon initial construction, G exhibits a linear structure, as illustrated in Figure 4. This reflects the fact that, by definition, every episode retrieved from C contains each and every milestone used to specify the query. However, this is only the starting point; to enable exploratory analysis, DecisionFlow provides a small but powerful set of *operators* that dynamically manipulate G in response to user interactions. These operators can be chained together during user interaction to support a wide range of event sequence analysis tasks.

4.3.1 Milestone Promotion

As previously described, G is represented with a two-level data structure. The top level is defined by a relatively small number of milestones, connected by directed edges. The second level is the raw intermediate event data (and corresponding event statistics) that are anchored to each top-level edge. This structure is key to DecisionFlow's ability to scale to very large numbers of event types. However, to support exploratory analysis over the full scope of available data, users must be able to access the low level event data stored within the second level of our data model.

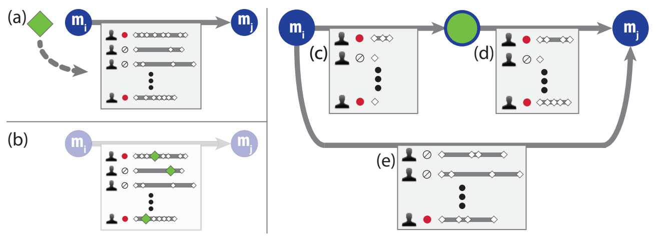


Fig. 5. The milestone promotion operator requires (a) an event type (shown in green) and an edge as inputs. (b) Intermediate episodes are checked for the target event type and, where found, split into "before" and "after" sub-episodes. The original edge is then replaced by a new milestone and three new edges corresponding to events (c) before, (d) after, and (e) without the new milestone.

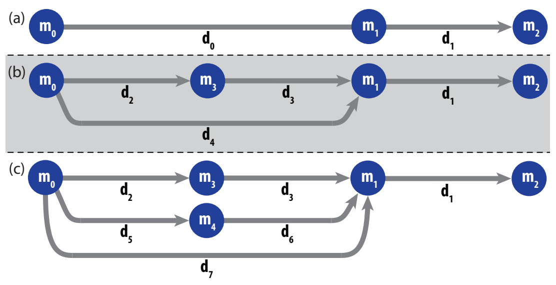


Fig. 6. (a) The original graph is first modified by (b) the promotion of m_3. Therefore, none of the intermediate episodes in d_4 contain m_3. (c) Next, the promotion of m_4 further modifies the graph. The edge d_5 contains intermediate episodes that have m_4 but not m_3. If they contained m_3, they would be in edge d_2. Edge d_7 only contains intermediate episodes that have neither m_3 nor m_4. This is represented by the order of edges leaving m_0 at the end of the progression: $[d_2, d_5, d_7]$.

Milestone *promotion* is the first operator designed to support this process. It converts an intermediate event type (selected via the Event Statistics Panel, Section 4.4.3) to a formal milestone, thereby inserting a new node m_i into G. This results in a new graph with a more complex structure. More formally, promotion is an operation performed on a specific edge d_j in a given graph G for a specific event type τ_i:

$$G' = Promote(\tau_i, d_j, G) \tag{1}$$

This process, illustrated in Figure 5, begins by searching each intermediate episode $\vec{e}_i$ in d_j for an occurrence of event type τ_i and dividing the episodes into two groups: those with τ_i and those without. The episodes within the τ_i group are then split into two derivative episodes: one for events taking place before τ_i and another for events occurring after τ_i. The result is that the original intermediate episodes for d_j are reorganized into three distinct collections: a "before milestone" collection, an "after milestone" one, and a "no milestone" one.

Once the episode event data has been processed, G is modified to reflect the new structure. Edge d_j is replaced by three new edges that connect the rest of the graph to a new milestone node representing the newly promoted τ_i. The three edges, shown in Figure 5(c-e), correspond to the "before milestone," "after milestone," and "no milestone" groups described above. In some cases, every intermediate episode in d_j contains an occurrence of the milestone event t_j. In this case, the "no milestone" group is empty and the corresponding edge is not added to G.

Finally, once G has been updated, edge analytics (see Section 4.2.3) are automatically computed on each of the three new edges to generate updated statistics.

One important property of our data model is that the list of edges departing a milestone node is ordered. To accurately reflect the process of how event sequences are grouped during promotion, the "no milestone" edge is always inserted into G after the "before milestone" edge. This ordering is critical for correct interpretation of the data structure. Consider, for example, the DecisionFlow graph in Figure 6 which was created by a series of two promotions (first m_3, then m_4). The first milestone, m_0, has three departing edges in the following order: d_2, d_5, and d_7. These can be interpreted using "if, then, else" logic. If an episode contains m_3 after m_0 but before m_1, it will be included in d_2 regardless of whether it contains m_4. This is true because m_3 was promoted first. Otherwise, if an episode contains m_4 after m_0, it will be included in d_5. Only episodes that contain neither m_3 nor m_4 are included in d_7. Figure 1 shows an example produced via multiple promotions.

4.3.2 Milestone Demotion

Milestone *demotion* is the inverse of promotion: it converts a milestone m_i into an intermediate event and modifies G by merging the impacted edges. Demotion operations are used to simplify the data structure, and are typically employed together with promotions during exploratory analysis. More formally, a demotion is defined as:

$$G' = Demote(m_i, G) \tag{2}$$

In the most basic case, demotion is straightforward. Consider the graph shown in Figure 5(c-e). To remove the newly inserted node (shown in green), the inverse of the promotion process is performed. The "before" and "after" intermediate episodes are spliced together and merged with the set of "no milestone" episodes. The three edges and the demoted node are then replaced by a single edge (containing the combined set of intermediate episode data). Finally, the edge analytics from Section 4.2.3 are re-computed for the new edge.

While the above process works for simple cases, a demotion can be complicated when performed on more complex graphs such as the one illustrated in Figure 6. The fully generalized demotion algorithm is given in Section 5.1.

4.3.3 Episode Filters

The final operator is the episode *filter*, which can be used in various ways to modify the set of episodes stored in the data representation, thereby potentially changing the structure of G indirectly. First, the filter operator can be used to *exclude* a specific milestone m_i. This results in the removal of all episodes that include m_i. Alternatively, the filter operator can be used to *include* only a specific milestone, which results in the removal of all episodes *not* passing through m_i. Similarly, the filter operator can be used to exclude (or include) individual event types. The filtering of event types can either be applied to episodes associated with a specific edge, or with the overall graph G. In either case, the episodes that match the filter constraints are removed entirely from G.

In some cases, filters can end up removing all intermediate episodes from a given edge. For these cases, the filtering process prunes G of all empty edges. Finally, edge analytics (see Section 4.2.3) are re-computed for all edges whose intermediate episodes were modified during the filter.

4.3.4 Discussion

While the operators described above can be used to construct arbitrarily complex graphs during an unfolding analysis, there are certain invariants for the DecisionFlow Graph data structure.

Planar Invariant. The initial construction of G results in a linear chain of milestone nodes, which is by definition planar. Promotions, meanwhile, are performed by replacing a single edge with a group of three edges as shown in Figure 5. This process always results in a planar graph. Demotion, being the inverse operation, also maintains planarity. Finally, the removal of an edge is the only structural change to G that can be introduced by a filter operator. Therefore G will always be planar.

Acyclic Invariant. The initial construction of G creates a linear chain of milestone nodes with no cycles. Promotion, which replaces a single edge with an acyclic group of three edges which flow in the same direction, cannot introduce a cycle. Similarly, demotion, which

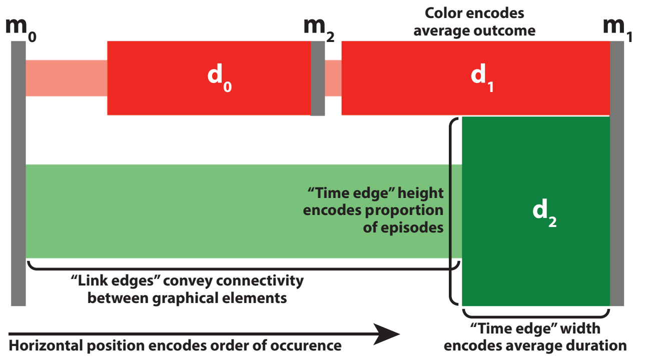
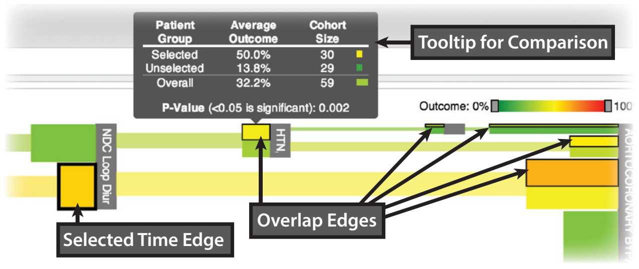

m0
m2
Color encodes average outcome
m1
d0
d1
d2
"Time edge" height encodes proportion of episodes
"Link edges" convey connectivity between graphical elements
Horizontal position encodes order of occurence
"Time edge" width encodes average duration

Fig. 7. The temporal flow panel uses gray rectangles to represent milestone nodes (m_i) which are arranged from left to right in temporal order. Edges d_i are represented using two different marks: time edges and link edges; they are color-coded to represent average outcome.



Patient Group	Average Outcome	Cohort Size	
Selected	50.0%	30	🟨
Unselected	13.8%	29	🟩
Overall	32.2%	59	🟩

P-Value (<0.05 is significant): 0.002

Fig. 8. A time edge is highlighted with a black outline after selection. Overlap edges are overlaid on top of the existing visualization to show how the episodes associated with the selected edge travel through the rest of the graph. Overlap edges are sized in proportion to the number of selected episodes and color-coded by outcome. Hovering over an overlap edge displays a tool-tip showing statistical comparisons between the selected and unselected episodes.

performs the inverse operation, cannot create a cycle. Finally, the removal of an edge in response to a filter will never result in the creation of a cycle. Therefore G will always be acyclic.

Singular Head and Tail Invariant. The initial construction of G produces a linear chain of milestones in which there is one head node and one tail node. Promotion can only insert new milestones between existing milestones. Similarly, demotion can only remove a milestone from the interior of G. Finally, because the head and tail milestones are created in the initial construction of G, all episodes pass through these milestones. Therefore, they cannot be removed by a filter unless all data is filtered out, resulting in an empty graph.

4.4 Interface Design and Visual Encoding

In addition to the visual query interface described previously, DecisionFlow includes three coordinated visualization panels. First, a temporal flow panel is used to visualize the top-level structure of G. User selection within the temporal flow view is linked to two additional panels: an edge overview panel and an event statistics panel.

4.4.1 Temporal Flow Panel

The primary view in DecisionFlow is rendered in the *temporal flow panel*. This panel visualizes the top-level structure of G: the directed graph of milestone nodes M connected by edges D. DecisionFlow employs a flow-based visual design that conveys a number of aggregate data properties.

Each milestone node m_i is represented as a fixed-width gray rectangle. Nodes that correspond to a query milestone (by definition including all episodes visualized), are rendered with a height of 100% (e.g., m_0 and m_1 in Figure 7). Because of the *Singular Head and Tail Invariant* (Section 4.3.4), the temporal flow visualization always begins and ends with full-height milestone rectangles.

Milestones introduced by promotion, such as m_2 in Figure 7, are represented with rectangles whose heights map to the fraction of the episodes that pass through the corresponding milestone. Because G is acyclic, milestone rectangles can be arranged along the horizontal axis from left to right based on the order in which they occur.

Edges, such as d_2 in Figure 7, are encoded using a pair of connected graphical marks: a link edge and a time edge. A *time edge* is encoded as a rectangle whose height (like that of a milestone rectangle) is proportional to the fraction of episodes that pass through the corresponding edge. The rectangle's width encodes the average duration of the intermediate episodes stored in the edge. Finally, time edges are color-coded based on the average outcome. A red-to-yellow-to-green color gradient is used by default as shown in the flow panel legend in Figure 1. Alternative color schemes can be used to support color-blind users. *Link edges*, meanwhile, are included in our visual encoding to convey connectivity between milestones and time edges. They are rendered as color-coded rectangles with reduced saturation and height, to set them apart from the corresponding time edge.

The above visual encoding draws upon a number of design elements from our previous work on Outflow [31]. However, in contrast to Outflow, DecisionFlow graphs are planar and encode logical information in the order of edges. These differences require a unique layout algorithm, described in detail in Section 5.2.

Interaction. Users can interact with the flow panel in several ways. First, they can hover over various graphical marks with a mouse to get precise metadata (such as average duration for edges, and event type information for milestones).

Second, they can click on time edges to perform *edge selection*. When an edge d_i is clicked (a) the corresponding rectangular mark is outlined in heavy black to visually convey the selection, (b) the edge overview and edge statistics panels are dynamically updated to display information about d_i, and (c) a set of *overlap edges* are overlaid on top of the visualization.

The overlap edge feature supports a number of additional analysis capabilities, including information about how episodes associated with d_i flow through the other edges in G. This helps convey how episodes "arrive at," and "depart from," a given point in G. The overlap edges, highlighted in Figure 8 and visible in Figure 1, are sized proportionally by height to show the fraction of episodes in a given edge that are selected. Color-coding is used to show differences in outcome for the selected subgroup, and a tool-tip can be activated by hovering over an overlap edge to see quantitative data (including p-values) comparing the selected subgroup with others in a given edge.

Finally, right clicking on either edges or milestones activates a pop-up context menu, which gets automatically populated with a context-specific list of operations (including milestone demotion and filtering commands). As previously discussed, these operators cause structural and content changes to G. When such changes occur, the flow visualization smoothly animates between G and G', to avoid disruptions to users' mental maps. This is made easier by the generally local nature of changes to G in response to any of the operators, as well as the stability of our layout algorithm.

4.4.2 Edge Overview Panel

The *edge overview panel* is the simplest component of the user interface. When there is no selection in the flow panel, it summarizes the full set of episodes returned by the active query. When an edge is selected, it displays the same categories of information, but includes only intermediate episodes associated with the selected edge.

As seen in Figure 1(d), this panel displays a number of aggregate statistics. First, basic information such as the number of episodes and the average outcome are presented. Summaries of application-specific metadata (such as gender and age distribution for the medical example in Figure 1) can also be included. Finally, a histogram is used to show the frequency of each event type in the selected portion of the dataset. Event types are sorted with the most frequent first, and by default the list includes all observed event types. A filtering text box is also provided through which users can explicitly restrict the event type spectrum. This is done by performing a sub-string match against the event type names. Users can also click on an individual event type to select it. The event type filtering and selection commands entered here are coordinated with the event statistics panel described below.

4.4.3 Event Statistics Panel

The *event statistics panel* is a critical element in the DecisionFlow design because it provides the place where users can explore the larger event space hidden from view in the flow panel, make comparisons between events over time, and select specific events for promotion to define new milestones. It displays statistics about whichever edge is selected in the flow panel, or, when no edge is selected, for the overall set of episodes returned by the query.

The panel design, seen in Figure 1(c), consists primarily of a color-coded bubble chart that encodes many of the statistics calculated by the edge analytics (see Section 4.2.3). Individual event types are represented with circles whose radius and color reflect correlation and odds ratio, respectively. The same red-to-yellow-to-green color coding scheme used in the flow diagram is adopted here. Each event type circle is positioned within a two-dimensional space defined by positive support on the X axis and negative support on the Y axis. Therefore, circles for event types that appear only in episodes with a positive outcome appear along the X axis, those that appear only in episodes with negative outcomes appear along the Y axis, and those that have equal support are positioned along the plot's diagonal. To aide interpretation of the chart, the X axis is displayed in green, the Y axis in red, and the diagonal in yellow. Given the nature of the odds ratio and correlation statistics, event type circles grow larger and more polarized (red or green) as they move away from the diagonal. Because many events appear with exceptionally low frequency, circles are omitted from the chart if both support levels fall below a threshold (e.g., both support levels below 0.05). We also sort the circles prior to rendering to place smaller circles on top. This helps limit problems due to over-plotting (which in practice was not found to be problematic).

This design makes finding an event type with strong correlation or odds ratio easy, but the statistical power of such measures depends upon factors such as sample size, which are less intuitive and less readily visible. Therefore, circles for event types with a statistically significant p-value (e.g., $p < 0.05$) are distinguished from the others with a thin black outline.

Coordination. The event statistics panel is closely linked to the temporal flow panel. In particular, selection of an edge in the temporal flow panel is used to determine which intermediate episode data is shown via the event statistics panel. Transitions in response to changes of selection in the flow panel are animated using a three-step, staged animation process. First, circles for event types that were only in the old selection are removed from the bubble chart. Second, event types that are in common between the old and new selections are animated to new positions, sizes, and colors to reflect the edge statistics for the new selection. Third, the process concludes by introducing new circles for all event types that are only present in the new selection.

The event statistics panel is also linked to the edge overview panel. The filtering text box used to limit the event types shown in the overview panel's histogram also restricts which circles are displayed on the event statistics panel. This allows users to quickly search for specific event types or categories. Moreover, selection in the overview panel's histogram is linked to selection in the bubble chart. This makes it possible to search for and select a specific individual event type by name.

Interaction. A user can interact with the event statistics panel in multiple ways. First, one can hover over individual event circles to see a tool-top that displays the event type name and more detailed statistical information in the sidebar located to the right of the bubble chart.

Second, one can select individual event types by clicking on the corresponding circle. Selected circles are highlighted with a heavy black outline, and the event type selection is maintained across selection changes in the flow diagram. Together with the animated transitions described earlier, this allows users to navigate from one edge to another in G to see how the significance of a given event type evolves across different milestones.

Finally, one can right-click on an event type to activate a pop-up context menu. This is automatically populated with a context-specific list of operations, including milestone promotion and filtering commands. Of particular importance is the promotion operation, as this is how users perform exploratory analysis. After selecting an edge d_j in the flow diagram and an event type τ_i in the event statistics view, a user can issue the $Promote(\tau_i, d_j, G)$ operator to dynamically insert a new milestone. In response, the graph is modified to G', the edge analytics are re-computed as needed, and the visualization panels are animated to reflect the changes between G and G'.

Algorithm 1 Base case milestone demotion algorithm

```
procedure BASEMILESTONEDEMOTION($m_i, G$)
    $DemotedArrEdge$ = Edge arriving at $m_i$
    $DemotedDepEdge$ = Edge departing $m_i$
    $G$ = $G$.remove($DemotedArrEdge$, $m_i$, $DemotedDepEdge$)
    $e$ = splice($DemotedArrEdge$, $m_i$, $DemotedDepEdge$)
    $PrecedingMilestone$ = Origin of $DemotedArrEdge$
    $BypassEdge$ = Last departing edge for $PrecedingMilestone$
    if $BypassEdge$ != $DemotedArrEdge$ then
        $G$ = $G$.remove($BypassEdge$)
        e = merge(e, $BypassEdge$)
    end if
    $G$ = $G$.insert(e)
    return $G$
end procedure
```

5 METHODS

This section provides greater detail on both the milestone demotion algorithm and the flow visualization layout algorithm.

5.1 Milestone Demotion Algorithm

As described in Section 4.3.2, the milestone demotion algorithm is straightforward when applied to the base case illustrated in Figure 7. The base case is formally defined with the following criteria: First, the milestone to be demoted (e.g., m_2 in the figure) must have exactly one arriving edge and one departing edge (d_0 and d_1, correspondingly). We refer to these as the *demoted arriving edge* and the *demoted departing edge*, respectively. Second, the milestone at the origin of the demoted arriving edge (e.g., m_0) can have at most one additional departing edge (e.g., d_2) that appears *after* the demoted arriving edge (e.g., d_0) (the reader is reminded that edges at a milestone are ordered - see Section 4.3.1). Finally, the aforementioned additional edge must arrive at the destination milestone for the demoted departing edge (m_1). We refer to these last two constraints as the *single bypassing edge requirement*. Only when all of these conditions are satisfied, can we apply Algorithm 1.

For cases that do not satisfy the above conditions, we apply recursion to trigger a wave of demotions to first enforce the base case constraints. This is done by first demoting milestones preceding m_i until a single demoted arriving edge is obtained. This repeats for subsequent milestones to obtain a single demoted departing edge. Finally, bypassing milestones are demoted until the single bypassing edge requirement is satisfied. Once complete, the base algorithm is performed and recursion completes. This is outlined in Algorithm 2, where we define a *bypass edge* as any edge that departs the preceding milestone after the demoted arriving edge, regardless of destination.

5.2 Layout Algorithm

The horizontal layout of milestone nodes in the temporal flow panel is a critical element of the DecisionFlow design. This is due to the logical dependencies that exist between milestones as described in Section 4.3.1. However, these logical relationships cannot always be determined solely based on the structure of G. For example, consider the graph in Figure 9. Based on the structure of G alone, it is not possible

Algorithm 2 Milestone demotion algorithm

```
procedure MILESTONEDEMOTION($m_i$, $G$)
    while $m_i$ has multiple arriving edges do
        $G$ = MilestoneDemotion(Origin of first arriving edge, $G$)
    end while
    while $m_i$ has multiple departing edges do
        $G$ = MilestoneDemotion(Origin of first departing edge, $G$)
    end while
    while $m_i$ has multiple bypass edges do
        $G$ = MilestoneDemotion(Origin of first bypass edge, $G$)
    end while
    return BaseMilestoneDemotion($m_i$, $G$)
end procedure
```

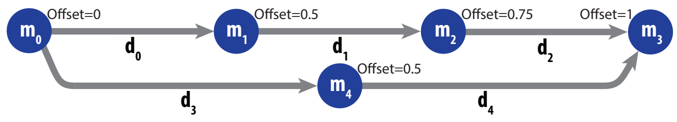


Fig. 9. The logical relationship between milestones can be ambiguous when considering only the structure of G. For example, does m_4 align with m_1 or m_2? Knowledge of the order of promotion is required to correctly answer this question. DecisionFlow uses offset values assigned at the time of promotion for this alignment task.

to know if m_4 should align with m_1 or m_2. However, this relationship is essential for the correct interpretation of the graph.

To enable the proper horizontal layout of milestone nodes, we assign an *offset* value to each milestone at the time it is inserted into G. By convention, we begin by assigning an offset of zero to the head node and an offset of one to the tail node. All other nodes are inserted into G using the *Promote* operator and assigned an offset that is the average of the "before" and "after" milestones' offsets. In Figure 9, we see that m_1 and m_4 share the same offset value and should therefore be aligned when visualized.

Given these offset values, the layout algorithm for the flow visualization of G is as follows. First, G is traversed to obtain a sorted list of all unique offset values in the data structure. (It is also possible to maintain a list of offsets as G evolves over time to avoid the time spent on this traversal. However, the relatively small size of G makes the cost negligible.) The total number of unique offset values (e.g., there are 4 unique offset values in Figure 9: 0, 0.5, 0.75, and 1) determines the number of milestone layers to use when rendering the temporal flow visualization. Second, we map the index of the offset value in the sorted list of unique offsets to a horizontal position using a linear scale. This produces an evenly spaced set of layers, one layer for each unique offset value in G. The end result is a neatly aligned visualization display as seen in the six layer example shown in Figure 1.

6 EVALUATION

We performed a user study to evaluate DecisionFlow's ability to support a range of event sequence analysis tasks in the presence of large numbers of event types. This section presents the study's design and reports its results.

6.1 Design

We asked 12 users (9 males and 3 females) to participate in a study of DecisionFlow using a dataset of electronic medical records. None of the subjects had prior experience with DecisionFlow, though all users were adult professionals who were comfortable with technology. The medical background of the users varied broadly, but study tasks were designed to require no medical knowledge. Study tasks focused on the use and interpretation of the visualization rather than on clinical significance. One limitation of this study design is that it does not evaluate DecisionFlow's capabilities for long-term exploratory analyses of the type that domain experts might undertake. We therefore plan long-term case studies as future work [24].

Each subject participated in one 60 minute study session. Each session began with a brief orientation in which a moderator reviewed the concept of temporal event analysis, introduced the medical domain of the dataset, and provided a short tutorial covering DecisionFlow's design and interactions. Participants were then given time to experiment hands-on with the prototype system.

Once familiar with the basic operation of the system, users were asked to perform a series of practice tasks using data from a practice query. This helped users further acclimatize to the system and gain experience with the specific types of questions that would be asked during the formal study period. Following the practice tasks, users were given a new temporal query to enter and asked to answer 12 study tasks. The moderator recorded both accuracy and time to completion for each of these tasks. Finally, users were given time for free exploration and asked to complete a post-study questionnaire/debriefing.

The dataset used for this study included de-identified medical event data for over 32,000 individual patients from a US-based care provider. There were over 8,000 unique event types in the overall database, including diagnoses, lab tests, and medications. The specific queries used for training and experimental tasks during the study returned between 514 and 2,899 patients. Those cohorts contained between 113,189 and 1,074,435 individual point events. Most important for our evaluation, the query result sets contained between 1,656 and 3,531 unique event types. This represents a two orders of magnitude increase in the number of event types compared to previous methods.

6.1.1 Tasks and Questionnaire

Each user was asked to perform 12 formal tasks designed to test users' abilities to interpret and interact with the DecisionFlow design. Five of the tasks focused on interpretation of the temporal flow visualization (T1-T4, T6). Another three focused on the use of the event statistics panel, in conjunction with the flow visualization, to analyze properties of intermediate events (T5, T7-T8). The final four tasks focused on the use of milestone operators and user interpretation of the resulting changes (T9-T12). The specific goals of the 12 tasks were as follows:

T1. Identify a specified intermediate episode in the flow view, given a set of temporal clinical constraints
T2. Compare proportion of population across flows
T3. Compare outcomes across flows
T4. Compare speed of progression across flows
T5. Understand the association between events and outcomes
T6. Understand temporal flow relationships (e.g., A before B)
T7. Identify intermediate events with the strongest associations to outcome
T8. Compare changes in event statistics over time
T9. Milestone promotion and time comparison
T10. Milestone demotion (base case)
T11. Milestone demotion (complex case)
T12. Logical interpretation of flow graph

At the end of the study session, users were given the opportunity to explore the data set without specific instructions. They were then asked to complete a post-study questionnaire with 13 questions answered on a 7-point Likert scale (1=very easy, 7=very hard). The questionnaire also included three free-response questions to gather more subjective feedback. The Likert-scale questions asked if it was easy or hard to:

Q1. Learn how to use DecisionFlow
Q2. Use once you've been trained
Q3. Specify a query
Q4. Interpret the proportion of patients exhibiting a specific set of events
Q5. Compare the average outcome between to pathways
Q6. Know which events have happened to a group of patients in the flow visualization
Q7. Remove a milestone

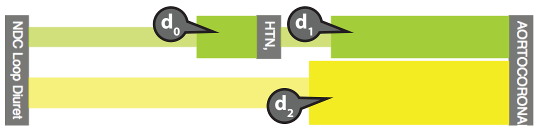


Fig. 10. In task T9, two users incorrectly reported that d_2 was the slower path through this portion of the flow graph after seeing that d_2 was longer than d_1. In fact, d_2 is the faster path because its length is smaller than the combined lengths of edges d_0 and d_1.

Q8. Understand the changes that take place when removing a milestone

Q9. Get an overall understanding of the intermediate events

Q10. Find a specific event in the event statistics panel

Q11. Promote an event to form a new milestone

Q12. Compare individual event statistics over time

Q13. Compare overall event set trends over time

6.2 Results

We report two results sets from our evaluation. First, we describe accuracy and time-to-completion measurements for the 12 tasks. We then summarize the feedback obtained via our post-study questionnaire.

6.2.1 Accuracy and Speed

Overall, participants were able to complete the study tasks with a high degree of accuracy. Of the 144 individual tasks observed during our evaluation, 141 were answered correctly - an accuracy rate of 98%.

Of the three recorded errors, two occurred during task T9. This task required participants to (a) promote a new milestone, and (b) identify which path through the modified flow visualization occurred most quickly on average (see Figure 10). While most people answered correctly, two made mistakes after quickly comparing the lengths of d_1 and d_2. However, because d_2 bypasses the middle milestone, users needed to compare d_2's length against the total combined lengths of edges d_0 and d_1 to obtain the correct answer. Both users recognized the mistake during their debriefing and attributed it to inexperience and answering too quickly.

The one other error was made on task T12. This question asked about patients having a *NDC Loop Diuretics* event. The user, however, reported looking simply for "Loop Diuretics" and therefore skipped over the milestone that should have been the focus of the question because its label started with "NDC."

6.2.2 Speed

Study participants generally performed the tasks very quickly, with completion times averaging between 1.7 for T2 ($\sigma = 0.7$) and 23.6 seconds for T8 ($\sigma = 18.5$). The variation in times reflects, in part, the differences in complexity between tasks. We also observed that for some of the more challenging tasks, certain users answered very quickly while others felt the need to double or triple check their answers. This led to longer averages and higher standard deviations (σ) in task completion time for tasks T8 and T12 ($mean = 13.7$, $\sigma = 13.5$), in particular. While it is difficult to draw too many conclusions from task completion time alone, these results show that DecisionFlow allows users to quickly navigate high-dimensional event data to answer rather subtle questions about event timing, outcomes, and population proportions.

6.2.3 Questionnaires

The results of the post-study questionnaire are shown in Figure 11. As the responses show, users generally found the system easy to use. The most difficult rating was given to Q1, which asked about ease of learning the DecisionFlow system. However, participants still performed strongly in both accuracy and speed measurements despite limited training time. Meanwhile, Q2 shows strong agreement that after training, DecisionFlow was easy to use.

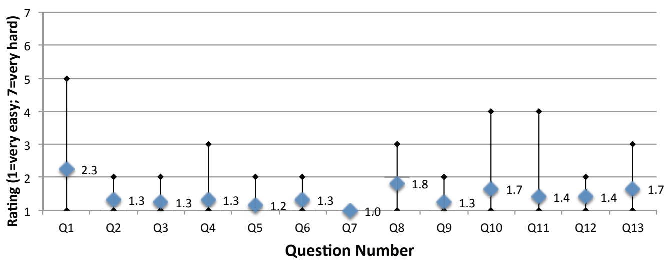


Fig. 11. Post-study questionnaire results for the 13 ease-of-use questions answered by study participants on a 7-point scale. The chart shows the average, minimum, and maximum responses.

Questions Q4-Q8 focused primarily on the flow visualization technique. Results show that users found this easy for both interpretation and interaction. Questions Q9-Q13, meanwhile, focused on analyzing intermediate events. The responses showed still low, but somewhat higher scores. This reflects, we believe, the more challenging cognitive task of sifting though large numbers (sometimes thousands) of event types. Yet, we consider these scores a sign of success given the inability of previous methods to handle such high-dimensional data sets.

The free response questions and our debriefing conversations yielded additional insights about what users perceived as the strongest and weakest aspects of our approach. Most commonly reported as confusing was the two-part edge design, with users often looking for meaning in the length of the link edge instead of the time edge. One user even stated that the link edge was "quite disturbing." In response to this feedback, we added a mode that omitted the time edge in favor of a simple one-piece link edge. This simplifies the visualization, but fails to convey the duration information, which may be important in some contexts.

The most frequent positive comments focused on the ease of finding strongly correlated events or milestone pathways. "I can easily compare the outcome of different groups of patients with vs. without a specific event." Users also found it "intuitive," "easy to use," and "very nice visually." They also reported that that the system was "very responsive" and provided immediate analytic results in response to interactive exploration.

7 Conclusion and Future Work

This paper presented DecisionFlow, a technique designed to support the visual analysis of high-dimensional temporal event sequence data. We described the key elements of the DecisionFlow technique, including: (a) the DecisionFlow graph, a two-level milestone-based data structure used for temporal query, aggregation, analysis, and visualization; (b) a set of operators that support dynamic manipulation of the graph in response to user interaction; and (c) the interactive interfaces and visual encodings adopted in our design. We also presented details on key algorithms and the results from a 12-participant user study. Speed and accuracy measurements recorded during the study demonstrate that DecisionFlow effectively supports the interactive ad hoc analysis of high-dimensional event sequence data. Qualitative feedback shows that users were generally happy with the design and felt that it was easy to use - despite the large number of event types in the study dataset.

While these initial results are promising, there are many possible topics to explore in future work. First, the DecisionFlow technique is limited to events which occur at a specific moment in time. Developing methods that also support interval events would significantly expand the space of possible applications for this technology. We also plan to conduct more comprehensive evaluations which go beyond learnability and usability. In particular, we plan to conduct longitudinal evaluations, observing experts working with our techniques in specific applied domains (e.g., healthcare and the life sciences) to better understand the benefits and limitations of our approach.

REFERENCES

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer, 1st edition, 2011.

[2] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl. PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation. In *2013 17th International Conference on Information Visualisation*, volume 0, pages 457–463, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[3] P. Almagro, B. Barreiro, A. Ochoa de Echagüen, S. Quintana, M. Rodríguez Carballeira, J. L. Heredia, and J. Garau. Risk factors for hospital readmission in patients with chronic obstructive pulmonary disease. *Respiration*, 73(3):311–317, 2006.

[4] P. Andr, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and M. Schraefel. Continuum: Designing timelines for hierarchies, relationships and scale. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, page 101110, New York, NY, USA, 2007. ACM.

[5] R. Bade, S. Schlechtweg, and S. Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, page 105112, New York, NY, USA, 2004. ACM.

[6] M. Burch, F. Beck, and S. Diehl. Timeline trees: Visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, page 7582, New York, NY, USA, 2008. ACM.

[7] S. B. Cousins and M. G. Kahn. The visual display of temporal information. *Artificial Intelligence in Medicine*, 3(3):341357, 1991.

[8] J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *IEEE Symposium on Visual Analytics Science And Technology*, page 167174, 2006.

[9] D. Gotz, F. Wang, and A. Perer. A methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data. *Journal of Biomedical Informatics*, Online Early Access, 2014.

[10] D. Gotz and K. Wongsuphasawat. Interactive intervention analysis. In *Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium*, volume 2012, pages 274–280, Washington, DC, USA, 2012.

[11] B. Harrison, R. Owen, and R. Baecker. Timelines: An interactive system for the collection and visualization of temporal data. In *Grahpics Interface*, 1994.

[12] S. Hibino and E. A. Rundensteiner. User interface evaluation of a direct manipulation temporal visual query language. In *Proceedings of the fifth ACM international conference on Multimedia*, MULTIMEDIA '97, page 99107, New York, NY, USA, 1997. ACM.

[13] J. Jin and P. Szekely. Interactive querying of temporal data using a comic strip metaphor. In *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 163–170, 2010.

[14] G. M. Karam. Visualization using timelines. In *Proceedings of the 1994 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA '94, page 125137, New York, NY, USA, 1994. ACM.

[15] M. Krstajic, E. Bertini, and D. Keim. CloudLines: compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[16] K. Minardo. Seeing sequences: How current temporal visualization techniques fail in complex domains. Technical Report 07-1225, The MITRE Corporation, 2007.

[17] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.

[18] M. Monroe, R. Lan, J. Morales del Olmo, B. Shneiderman, C. Plaisant, and J. Millstein. 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*, CHI '13, page 23492358, New York, NY, USA, 2013. ACM.

[19] P. Mui. Introducing Flow Visualization: visualizing visitor flow. Google Analytics Blog. `http://analytics.blogspot.com/2011/10/introducing-flow-visualization.html`, 2011. [Online; accessed 24-January-2014].

[20] A. Perer and D. Gotz. Data-driven exploration of care plans for patients. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 439444, New York, NY, USA, 2013. ACM.

[21] D. Phan, A. Paepcke, and T. Winograd. Progressive multiples for communication-minded visualization. In *Proceedings of Graphics Interface 2007*, GI '07, page 225232, New York, NY, USA, 2007. ACM.

[22] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 221–227, New York, NY, USA, 1996. ACM.

[23] A. Rind, T. Lammarsch, W. Aigner, B. Alsallakh, and S. Miksch. TimeBench: a data model and software library for visual analytics of time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2247–2256, Dec. 2013.

[24] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, page 17, New York, NY, USA, 2006. ACM.

[25] S. R. Steinhubl, B. A. Williams, J. Sun, R. J. Byrd, Z. S. Daar, S. Ebadollahi, and W. F. Stewart. Text and data mining of longitudinal electronic health Records (EHRs) in a primary care population can identify heart failure (HF) patients months to years prior to formal diagnosis using the framingham criteria. In *AMA Scientific Sessions*, Orlando, Florida, 2011.

[26] K. Vrotsou, K. Ellegard, and M. Cooper. Everyday life discoveries: Mining and visualizing activity patterns in social science diary data. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 130–138, 2007.

[27] K. Vrotsou, J. Johansson, and M. Cooper. ActiviTree: interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945952, Nov. 2009.

[28] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 457466, New York, NY, USA, 2008. ACM.

[29] P. C. Wong, W. Cowley, H. Foote, E. Jurrus, and J. Thomas. Visualizing sequential patterns for text mining. In *IEEE Symposium on Information Visualization, 2000. InfoVis 2000*, pages 105–111, 2000.

[30] K. Wongsuphasawat and D. Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare*, Providence, Rhode Island, USA, 2011.

[31] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):26592668, 2012.

[32] K. Wongsuphasawat, J. A. Guerra Gmez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 17471756, New York, NY, USA, 2011. ACM.

[33] K. Wongsuphasawat, C. Plaisant, M. Taieb-Maimon, and B. Shneiderman. Querying event sequences by exact match or similarity search: Design and empirical evaluation. *Interacting with Computers*, 24(2):55–68, Mar. 2012.

[34] K. Wongsuphasawat and B. Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology VAST*, Atlantic City, NJ, USA, 2009.