CHAPTER 5

DESCRIBING DATA EXPLORATION SESSIONS

The purpose of the GDE model is to store historical information about data exploration sessions, while preserving derivational relationships. It essentially defines a temporal database capable of tracking how an analyst explores some other database. In this chapter we use both forward derivation component sequence and graph representations to further characterize the data exploration process.

While we strive to describe data exploration in the general sense, i.e., independent of any data exploration domain, some descriptive measures developed in this chapter reflect the data exploration scenario posed in Section 3.1, which limits both the type of data and derivations allowed. This is only an artifact of the data exploration scenario being used. As other scenarios and task repertoires are considered, the types of data and the derivations allowed will increase.

After revisiting the concept of data exploration *state* in the context of the data exploration session, we present a small taxonomy of visual representation schemes for the forward derivation component graph, to aid in its comprehension. Then, the concept of data exploration *scope* is introduced as a framework for deriving measurable quantities (i.e., metrics) from the forward derivation component graph. Some metrics are static measurements of the process because they are historical summaries of vertices and forward derivation paths. Other metrics are dynamic measurements because they measure

changing data interaction values over the course of the session. Dynamic metrics are termed the *data exploration calculus* for this reason. As part of the data exploration calculus, differentiability, continuity and kinematic properties of data exploration are defined.

5.1 Data Exploration State Revisited

The concept of data exploration *state* corresponds to the data under investigation by an analyst *at a single instant in time*. This single instant is typically the moment after a derivation is applied, when the derivation image data entities are created. Note that the preimage vertices do not necessarily have the same timestamp value because they are chosen as required by the data analyst. In contrast, the image vertices are required to have the same timestamp value because they are all created at the same time.

The data analyst, however, must place any observation into the proper context. From the data analyst's perspective, the data exploration state can be viewed in two ways. One view is immediate, as mentioned above: the set of data entities currently being examined, and possibly their neighbors. The other view is historical: the sequence of derivations and state changes that preceded the current state, or the relationship between the current state and the session. Thus, the data exploration state is used to answer two important questions for the data analyst: "what am I looking at?" and "where am I?". In addition to being used in a temporally restricted, localized context, it is also used in a more expansive context, encompassing a sequence of states tracing back to the database, and thus can also answer the question "where am I going?" and "at what rate am I going?".

Answering the question "what am I looking at?" requires a knowledge of the current data derivation, and that the current state is made visible to the analyst. Answering the question "where am I?" places a larger context on the analyst's knowledge, and

requires a relationship to be expressed between the current state and the data source, namely the data lineage (see Section 4.5.3).

A data exploration state exists in some multidimensional data space defined by the data sets contained within its data entities. In terms of the data exploration scenario of Section 3.1, this *data exploration space,* $\mathbf{S_{dexp}}$, is defined as the set of all possible data exploration states, or the Cartesian product of all data attributes *and* the configurable visualization display parameters. What this means is that not only is there a meaningful data subset, but also a meaningful data presentation. The set of values defined by each distinct data attribute and visualization parameter must belong to a *metric space* to be included in $\mathbf{S_{dexp}}$. Metric spaces have distance functions (i.e., metrics) associated with them.

The size of this space, even for small databases can be huge, since in our model it represents all possible database views *and* visualization views. For example, if a database has ten distinct selectable attributes, and ten data-independent visualization display parameters that belong to metric spaces, then the data exploration space has twenty dimensions. Overcoming the shear size of the data exploration space is the central challenge of data exploration.

In this section we have shown that data exploration state has several interpretations. One is the data-centric view of a derivation preimage or image. Another is the user-centric (i.e., analyst) view, which includes neighboring data entities and the paths back to the data store, to provide context and answer questions of location within the all-encompassing data exploration space.

5.2 Representing Data Exploration Sessions

In this section we point out some of the design issues in conveying the information embedded in the forward derivation component graph, and introduce an alternate representation for our illustration purposes. The forward derivation component graph is an information-rich data structure, and its visualization requires careful planning. We'd like the visualization to convey as much information as possible, so the number of displayed vertices and edges must be maximized. The visualization must also be comprehensible, so there are limits to the number of displayed vertices and edges. Two activities that address these competing issues are:

1. choosing the visual representation scheme

2. reducing the graph complexity

Choosing the visual representation scheme - There are many possible visual representation schemes, each having design and perceptual tradeoffs. Here we wish to relate some of the critical issues in choosing an effective scheme. Our visual representation scheme repositions the graph vertices along two major axes, *exploration depth* and *exploration breadth*. Exploration depth is the number of edges from the data source to a target vertex, and exploration breadth is the number of vertices at a particular depth. Vertices are positioned based on their *level number* with respect to these two axes. This places the process into a metric space that is defined by the level number, allowing the determination of derivational distances between vertices. Exploration breadth and depth as metrics are discussed in greater detail in Section 5.4.2. The visualization of the forward derivation component graph of Figure 4-7 is shown in Figure 5-1.

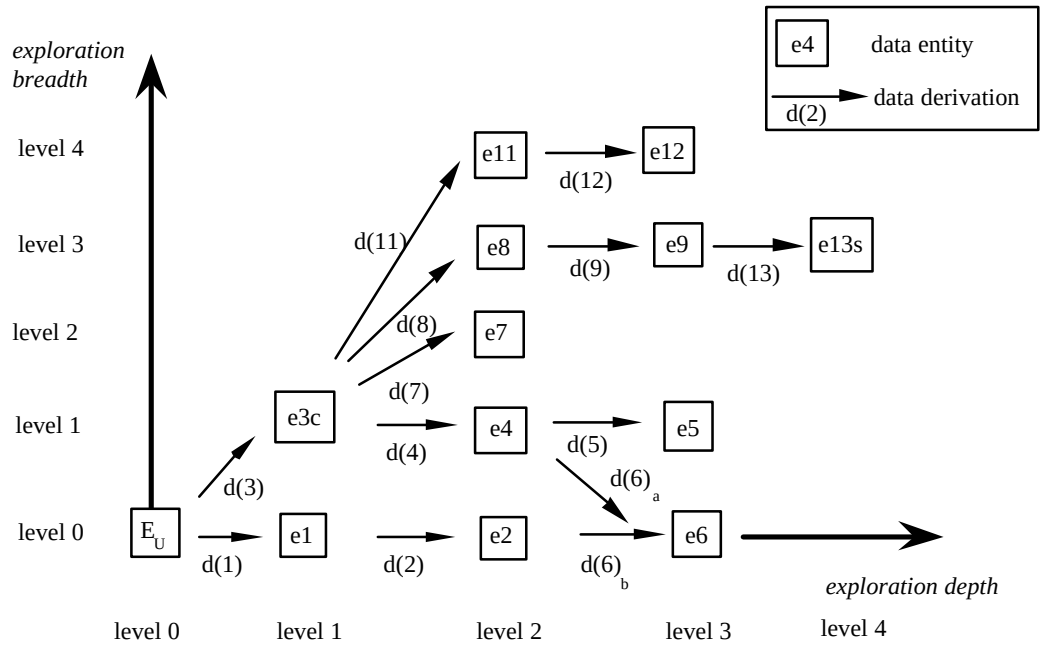


Figure 5-1. Visualization of the forward derivation component graph of Figure 4-4.

This visualization technique allows two attributes of data exploration (depth and breadth) to be easily determined and compared. Time progresses in only two directions (up and to the right), instead of an arbitrary direction, making the session more comprehensible. It does suffer, however, from occlusion in cases where vertices and edges overlap as the degree of branching increases with longer sessions. Additionally, the branching structures are not portrayed with any kind of symmetry, making the visualization look unbalanced (e.g., some derivation edges are longer than others).

This effect can be ameliorated somewhat by increasing the dimensionality of the breadth axis from linear to planar. The result would be a visualization similar to Cone Trees (Robertson, Card and Mackinlay, 1991), where branches form a conical shape, and the exploration depth axis converts into a collection of circles. The problem with this approach is that the ordering to the breadth axis is lost, making it difficult to deduce

precise breadth values directly from the graph, without additional assistance or cues from the visualization.

We can perform a limited amount of vertex repositioning and introduce symmetry into the branching to make the graph look more balanced and increase the display density. In this modification, an optimization strategy is applied to pack all the vertices that are created via branches into the vertical space between the bounding horizontal paths, such as in Figure 5-2. As a local solution degenerates (i.e., we run out of room to place a vertex), the strategy backtracks through the graph to expand the vertical positioning of the bounding horizontal paths. An approach such as *simulated annealing* is effective here. The caveat is that the breadth axis has irregular spacing between levels, making visual comparisons difficult. As a visualization aid, a histogram of the breadth at each depth value can be provided for summary information.

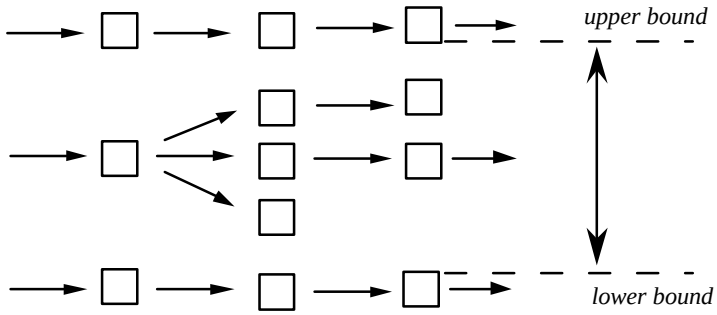


Figure 5-2. Bounds on repositioning vertices with increasing branching.

Reducing graph complexity - The visualization of Figure 5-1 suffers from the central problem of a very dense integration of text and graphics. There is so much textual information at vertices and edges that the labeling scheme can obfuscate any emerging structure in the graph. One can easily “lose sight of the forest for the trees”, and become

mired in the details of timing and labels. Two ways of reducing graph complexity are eliminating the graph notation and reducing the number of derivations for visualization.

Eliminating the notation of the graph immediately allows increasing the density of the visualization, as valuable display space once reserved for textual labels is used to hold more vertices and edges of the graph. Such a graph is called a *summary* graph. The summary graph shows more derivations applied by the analyst at the expense of identifying the derivations. Also, there is no representation of explicit state changes anywhere, where previously the timestamp vector within each vertex contained this data. The visualization of Figure 5-1 is shown in Figure 5-3 as a summary graph.

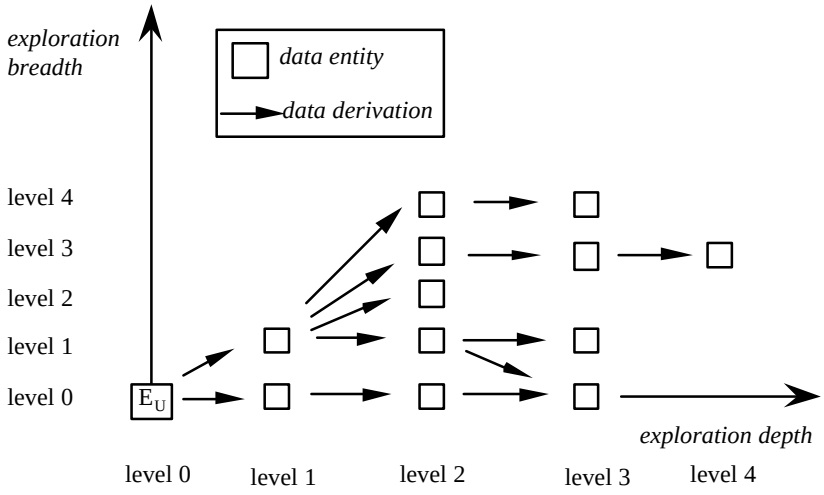


Figure 5-3. The graph of Figure 5-1 visualized as a summary graph.

The summary graph representation gives the *gestalt* of the data exploration session. One can easily see clusters of derivations, landmark vertices, and interesting structural patterns. It is purely a derivational view, irrespective of time and explicit state changes. At this level, it does not matter what the derivations actually are; all that matters is that they derive data.

Reducing the number of derivations of the graph requires an intimate knowledge of the derivations applied to the data. Specifically, if the set of derivations to be applied exhibit *closure* (such as the relational operations used in relational database systems), then sequences of related derivations can be replaced by a single derivation, thereby compressing the forward derivation component graph. Such a graph is called a *compressed* graph. One possible compressed summary graph of Figure 5-3 is shown in Figure 5-4.

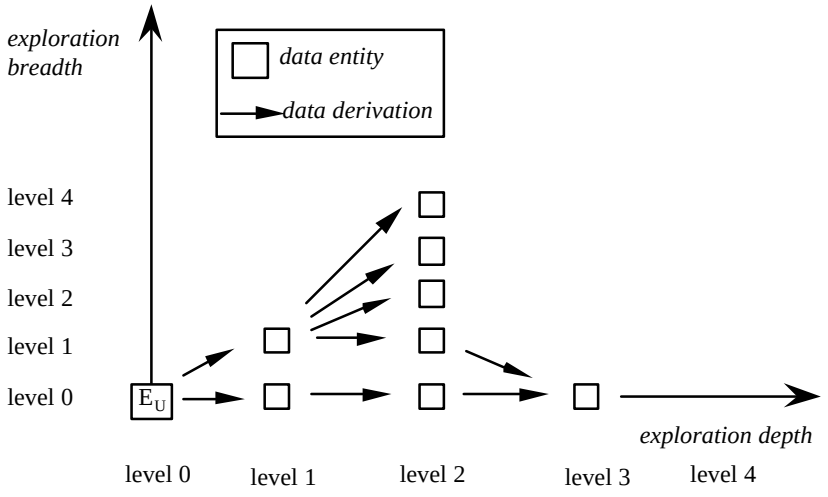


Figure 5-4. A possible compressed summary graph of Figure 5-3.

The compressed graph might be useful for determining a minimal path to the discovered knowledge, a lower bound on the exploration session. This runs the risk, however, of "putting derivations in the analyst's head" that never existed in the first place. Compression might show a "lower bound" on an exploration session, or how an "expert" analyst might have explored the database.

Table 5-1 shows a taxonomy of the visual representations of the forward derivation component graph discussed in this section.

Number of vertices and edges	Notation Present	No Notation Present
maximal	detailed	summary
less-than maximal	compressed	compressed summary

Table 5-1. A taxonomy of visual representations of the forward derivation component graph.

It is beneficial to use both the detailed and summary graph representations in conjunction with each other. The summary graph lends itself towards visual inspection, and would be useful for an overview of the session, to show clusters and patterns more clearly than the detailed graph. The detailed graph lends itself to algorithmic analysis, and would be useful for in-depth probing of the component derivations and data entity contents.

5.3 Data Exploration Scope

Data exploration scope is like an environment, or a region of influence, that places the analyst in a specific context within the forward derivation component graph, and therefore the exploration session. This is similar to the definition of scope in the programming language and compiler domains (a region of program text). Data exploration scope is simply a temporally-independent neighborhood of the forward derivation component graph, in terms of a particular reference structure: a data entity vertex, a forward derivation path, or the entire forward derivation component graph. It is a structural description that determines what data is relevant and accessible for operations such as graph construction, analysis and presentation. We identify three kinds of data exploration scope: *local, derivation path* and *exploration session scope*.

Local scope focuses on a single data entity vertex, its incident and exiting edges, and the other vertices they connect to, as shown in Figure 5-5. This provides the minimum amount of information to describe the data entity vertex as a derivation result or source. Figure 5-5 shows the local scope of a *target* vertex, that includes its two immediate ancestor vertices and three immediate successor vertices, labeled *fringe* vertices.

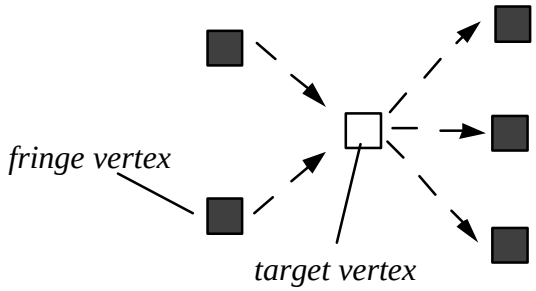


Figure 5-5. The local scope of a single target vertex.

Local scope is based on the derivation ordering, so all derivations are included, regardless of their timestamp values. This gives a one-dimensional history of a data entity vertex during the entire exploration session. Note that if a set of target vertices are specified, each would have its own local scope, and the local scope of the entire set would be the union of the local scopes of each target vertex.

Derivation path scope is associated with the forward derivation path, and therefore contains information about how a data entity was derived, or how it was subsequently transformed during a portion of the exploration session. In much the same way that local scope is defined, derivation path scope encompasses a forward derivation path plus the fringe vertices the path connects to, as shown in Figure 5-6.

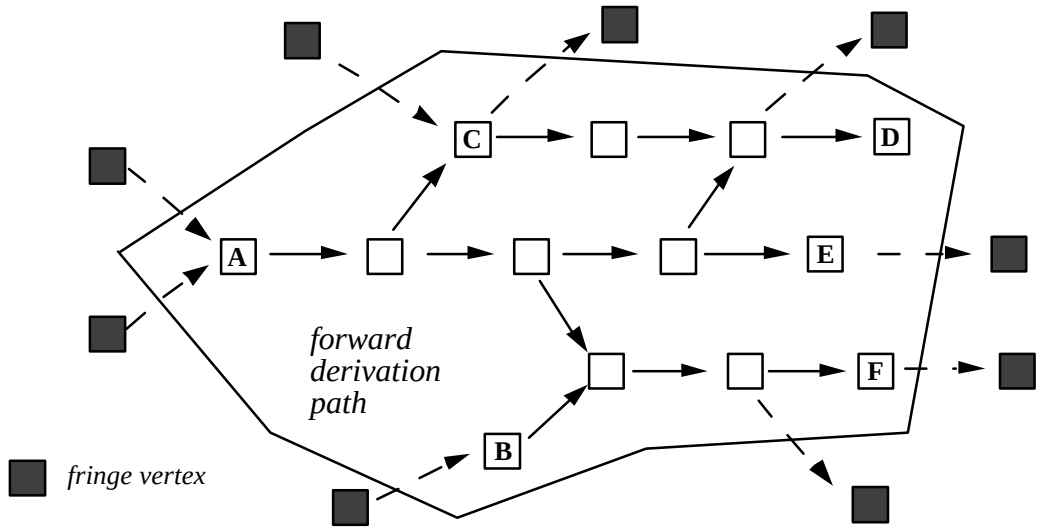


Figure 5-6. The derivation path scope of a target forward derivation path.

In Figure 5-6, a forward derivation path is shown between two sets of data entity vertices {A, B} and {D, E, F}, with additional incident and exiting edges connecting to fringe vertices. Vertex C is not classified as an input vertex because only fringe vertices connect to input vertices. Likewise, vertex C is not classified as an output vertex because output vertices only connect to fringe vertices. Though the fringe vertices are not part of the forward derivation path, they are considered part of the derivation path scope.

Global, or *session scope*, pertains to the entire forward derivation component graph, and is used for describing entire exploration sessions, or comparing sessions. The distinguishing characteristic is that it must include the database, whereas derivation scope might not. Global scope also contains all tagged vertices which may represent units of knowledge.

In summary, data exploration scope defines a frame of reference for determining descriptive information about the data exploration, relative to a location within the data derivation graph: a data entity vertex, a derivation path or the entire session graph itself.

This frame of reference allows us to define metrics (in Section 5.4) on appropriate portions of the forward derivation component graph.

5.4 Data Exploration Metrics

Describing the data exploration session requires measuring the process. Measuring the process requires making comparisons among the artifacts of the process (vertices, derivations, derivation paths, etc.). A *metric* is a standard of measurement, where values are associated with elements of a set. This section defines a number of *domain-independent* data exploration metrics that describe the interaction structures that make up the forward derivation component graph. Though the actual domain data stored in vertices and edges is unknown at this point, there is still enough information available to derive useful metrics about how the analyst interacts with the data during the session. The following vertex attributes are used to derive or affect data exploration metrics:

- *vertex indegree and outdegree values*
- *timestamp values*
- *tag and comment fields*

Vertex indegree and outdegree values are the most useful attributes for computing data exploration metrics. They show the number of derivations a vertex participates in, either as an input or an output.

Timestamp values help identify and temporally order vertices and edges. Since derivations can contain numerous edges, it is important to know how many distinct derivations compose the derivation edges present at a vertex. Timestamps can also show temporal clusters of activities within the exploration session.1

Tag and comment fields are part of data entity metadata, and identify vertices that are important to the analyst. Not only do they account for the analyst's insights and observations useful for subsequent analyses, they provide the impetus and reference points for the analyses. A tag, being a binary element, is relatively easy to use. Its value signifies something should be done with the vertex, such as determine the data lineage. A comment, being an unstructured text element, is harder to use. Its purpose as a recording feature of the analyst's thoughts would serve as a contextual assistant when revisiting or analyzing the session graph.

From this vertex-resident information, we can progressively build complex metrics that describe elemental aspects of exploration sessions. We identify two types of metrics, organized by the portion of the forward derivation component graph they describe: *vertex-based metrics* and *derivation path-based metrics.*

5.4.1 Vertex-Based Metrics

Vertex-based metrics use the local scope of a vertex, and fall into two categories: fundamental metrics and derived metrics. Fundamental metrics are determined directly from the vertex degree values. Derived metrics are computed from the fundamental metrics. Since derivations in general can be represented by multiple edges, the indegree and outdegree values do not necessarily correspond to the number of individual derivations associated with a target vertex, as shown in Figure 5-7. In Figure 5-7, the indegree (2) does not equal the number of incident derivations (1), though the outdegree (3) equals the number of exiting derivations. As in Chapter 4, we are still using the data-entity view as opposed to the set-of-data-entity view.

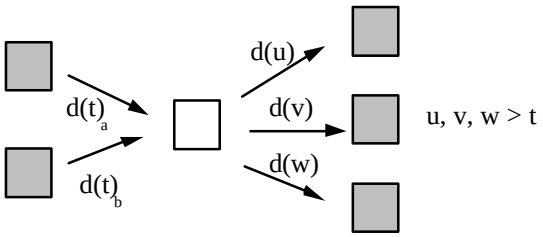


Figure 5-7. A difference between indegree and number of incident derivations.

We now define three fundamental metrics, based on the indegree and outdegree values, but phrased in terms of distinct data derivations:

- the derivation result metric $\mathbf{M}_{\text{result}}$
- the derivation source metric $\mathbf{M}_{\text{source}}$
- the derivation visitation metric $\mathbf{M}_{\text{visit}}$

$\boldsymbol{M}_{result}$: Derivation Result Metric

The derivation result metric measures the number of *distinct* incident derivations to the target vertex, as is shown in Figure 5-7. This indicates how often the vertex was the *result* of a forward derivation. Thus, $\mathbf{M}_{\text{result}}$ measures the input capacity of a vertex in terms of distinct data derivations. $\mathbf{M}_{\text{result}}$ also corresponds to the number of implicit state changes to the vertex, since implicit state changes result from forward derivations. Since the existence of a vertex implies at least one incident derivation exists, $\mathbf{M}_{\text{result}} \geq 1$.

$\boldsymbol{M}_{visit:}$Visitation Metric

The visitation metric gives the total number of times the vertex was visited, or $\mathbf{M}_{\text{result}} + \mathbf{C}_{\text{result}}$, where $\mathbf{C}_{\text{result}}$ equals all explicit state changes into the target vertex. This

metric can be compared against a user-defined threshold value, to determine whether or not it can be designated a *landmark* vertex. A landmark vertex is a prominent vertex in the data exploration session. Its data entity serves as a known reference point. A landmark vertex would be expected to have many explicit incident derivations, as the analyst continually changes the exploration state to the now-familiar vertex.

***M**$_{source}$: Derivation Source Metric*

The derivation source metric gives the number of *distinct* exiting derivations from the target vertex, or how many times the target vertex was used as the *source* of a forward derivation. Thus, $\mathbf{M}_{source}$ measures the derivation output capacity of a vertex. Unlike $\mathbf{M}_{result}$, $\mathbf{M}_{source} \geq 0$. If $\mathbf{M}_{source} = 0$, the vertex is a *terminal* vertex. A terminal vertex has no exiting edges; all interactions with the vertex are explicit state changes out of the vertex to some other existing vertex.

These three metrics provide a minimal set of measures that summarize how a data entity vertex was used during a data exploration session. We can now express a simple, but fundamental relationship among these metrics, along with explicit state changes in the *Conservation of State Change Property* (CSCP):

$$\mathbf{M}_{visit} = \mathbf{M}_{result} + \mathbf{C}_{result} = \mathbf{M}_{source} + \mathbf{C}_{source} \qquad \text{(Equ. 5-1)}$$

An additional explicit state change value, $\mathbf{C}_{source}$, is required to account for the number of state changes out of the vertex. The CSCP expresses the fact that the sum of all state changes, both implicit (i.e., derivational) and explicit, into a vertex equals the sum of all state changes out of the same vertex. This applies to every vertex except the database and the last vertex created in the session. There are only two ways to arrive at a vertex: by deriving it through one or more forward derivations from the database (or some

intermediate vertex), or by changing to it from some later state in the session. Similarly, there are only two ways to leave a vertex: by deriving a new state from it or by changing to some other existing state. Figure 5-8 shows this property graphically.

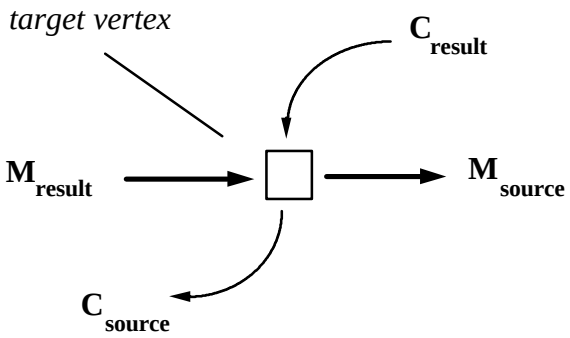


Figure 5-8. The fundamental vertex-based metrics and their relationship.

Note that both $\mathbf{C}_{source}$ and $\mathbf{C}_{result}$ are not included as fundamental metrics, because they can be determined directly from the fundamental metrics, as evidenced in Figure 5-8. Furthermore, they are not implicit derivations and do not directly contribute to the advancement of the data exploration session. Thus, they are not stored in the forward derivation component graph, though they are stored in the more general data derivation graph.

Considering each metric individually does not yield enough information to describe the vertex accurately, since there can be four kinds of interactions with the vertex during the data exploration session, as shown in Figure 5-8. The exception, as mentioned above, is if $\mathbf{M}_{source} = 0$, where the vertex can be immediately classified as a terminal vertex.

Table 5-2 summarizes the meaning of five important configurations of the fundamental vertex-based metrics. Cases 1, 2 and 3 are simple configurations. In case 1, all metrics are unity, so the vertex can be considered a mere waypoint in a simple

derivation path. Such a vertex is called a *sequential* vertex. In case 2, all metrics equal the same positive integer *k*, which indicates cycles exist within the session. Such a vertex is called a *k-sequential* vertex. In case 3, $\mathbf{M}_{source} = 0$, so the vertex is a *terminal* vertex, regardless of $\mathbf{M}_{result}$ and $\mathbf{M}_{visit}$.

Case	$\mathbf{M}_{result}$	$\mathbf{M}_{source}$	$\mathbf{M}_{visit}$	**Comments**
1	1	1	1	vertex is a waypoint in a simple derivation path - a *sequential* vertex
2	k: k > 1	k	k	*k-sequential* vertex, probable cycle
3	r: r > 1	s: s = 0	v: v > 1	*terminal* vertex
4	r: r > 1	s: s > r	v: v > r	v - r explicit state changes into vertex, possible *landmark* vertex if v >> r possible *quasi-source* vertex if s >> r
5	r: r > 1	s: r > s > 1	v: v = r	r - s explicit state changes out of vertex, possible *quasi-terminal* vertex if r >> s

Table 5-2. Vertex-based metrics summary.

Cases 4 and 5 are more complex configurations. In case 4, there are more exiting derivations than incident derivations, and there are more visits to the vertex than incident derivations. This indicates there are additional explicit state changes into the vertex, suggesting it might be a *landmark* vertex. This is more plausible if the number of visits are much greater than the number of incident derivations (v >> r). Also, if there are many more exiting than incident derivations (s >> r), then the vertex may be a *quasi-source* vertex. In case 5, there are more incident derivations than exiting derivations and there are the same number of visitations as incident derivations. This indicates the opposite of case

4: there must be additional state changes out of the vertex, suggesting it is some type of *terminal* vertex. If r >> s, then the vertex is designated a *quasi-terminal* vertex.

The next set of vertex-based metrics build upon the fundamental metrics, by combining them in various ways. As with the fundamental metrics, these derived vertex-based metrics are historical summaries, and thus do not consider temporal aspects of the data exploration session. Unlike the fundamental metrics, these metrics deal with $\mathbf{M}_{\text{result}}$ and $\mathbf{M}_{\text{source}}$ exclusively. These derived metrics are precursors to the calculus of interactions that is developed in Section 5.5.

ΔM : Derivation Increment

The derivation increment is the difference between a vertex's derivation source and result metrics, $\Delta M = \mathbf{M}_{\text{source}} - \mathbf{M}_{\text{result}}$. If $\Delta M > 0$, there are more exiting than incident derivations, making the vertex a *divergent* vertex. If $\Delta M < 0$, there are more incident than exiting derivations, making the vertex a *convergent* vertex. If $\Delta M = 0$, more information is required to classify the vertex as sequential or k-sequential, as indicated in Table 5-2. By Equation 5-1, $\Delta M = \mathbf{C}_{\text{result}} - \mathbf{C}_{\text{source}}$. If $\Delta M > 0$, $\mathbf{C}_{\text{result}} > \mathbf{C}_{\text{source}}$, so there are more explicit state changes into the vertex. If $\Delta M < 0$, $\mathbf{C}_{\text{result}} < \mathbf{C}_{\text{source}}$, so there are more explicit state changes out of the vertex.

R_d: Derivation Ratio

The derivation ratio is the ratio of derivation output to input, $R_d = \mathbf{M}_{\text{source}} / \mathbf{M}_{\text{result}}$. If $R_d = 0$, the vertex is a terminal vertex. If R_d is very large, meaning $\mathbf{M}_{\text{source}} >> \mathbf{M}_{\text{result}}$, the vertex is a quasi-source vertex. This ratio also further refines the divergence of a vertex, and gives a distinguishing criteria when the ΔM of two vertices are similar. For example, for vertex v_a in Figure 5-9(a), $\mathbf{M}_{\text{result}} = 4$, $\mathbf{M}_{\text{source}} = 200$, and for vertex v_b in

Figure 5-9(b), $\mathbf{M}_{result} = 204$, $\mathbf{M}_{source} = 400$. The derivation increments are equal, but the derivation ratios are not, for v_a, $R_d = 50$ and for v_b, $R_d = 1.96$. When comparing *divergent* vertices, the *higher* the R_d, the more divergent the vertex. Thus, vertex v_a is more divergent than vertex v_b.

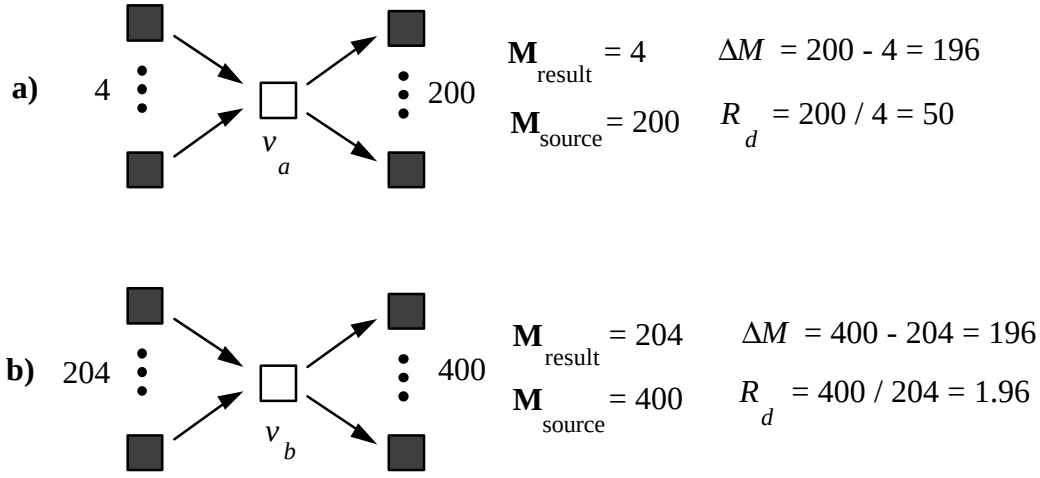


Figure 5-9. Comparing divergent vertices with the Derivation Ratio.

When comparing convergent vertices, the *lower* the R_d, the more convergent the vertex. Consider Figure 5-10, where the fundamental metrics are the reverse of Figure 5-9. Vertex v_a is now more convergent than vertex v_b.

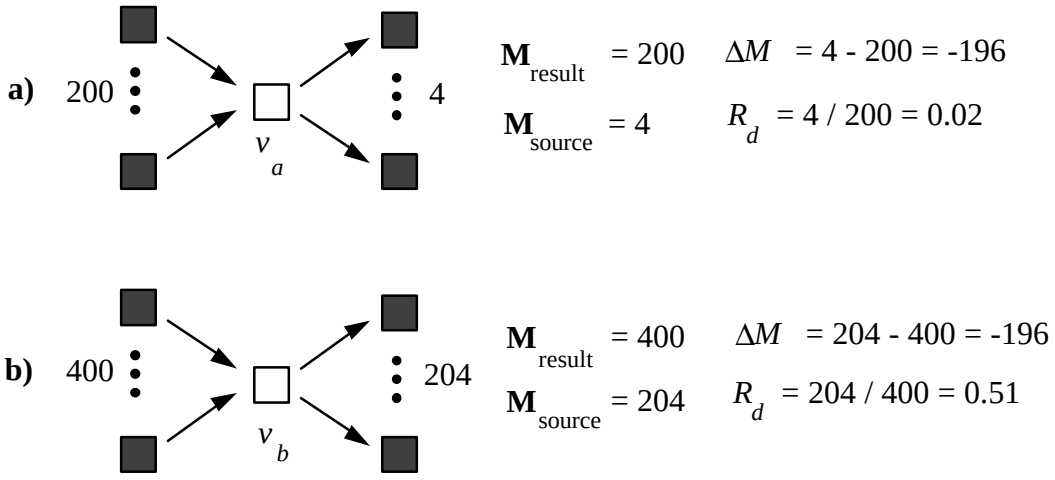


Figure 5-10. Comparing convergent vertices with the Derivation Ratio.

R_{conv}: Convergence Ratio

The convergence ratio is used for comparing the convergence between *any* pair of vertices, not necessarily convergent vertices. It is the ratio of the derivation result metric to the derivation increment, $R_{conv} = \mathbf{M}_{result} / \Delta M$. This value represents the percentage of all derivations of the vertex that are incident to the vertex. For the example given above, for vertex v_a in Figure 5-9(a), R_{conv} = 0.0196, and for vertex v_b in Figure 5-9(b), R_{conv} = 0.3377. In Figure 5-10(a), R_{conv} = 0.98, and in Figure 5-10(b), R_{conv} = 0.6622. Using R_{conv}, it is apparent that the vertices in Figure 5-10 are more convergent than their counterparts in Figure 5-9.

R_{div}: Divergence Ratio

Like the convergence ratio, the vertex divergence ratio is used for comparing the divergence between *any* pair of vertices, not necessarily divergent vertices. It is the ratio of the derivation source and the derivation increment, $R_{div} = \mathbf{M}_{source} / \Delta M$. This value is the percentage of all derivations of the vertex that exit from the vertex. For the example

given above, for vertex v_a in Figure 5-9(a), R_{div} = (200 / 204) = 0.98, and for vertex v_b in Figure 5-9(b), R_{div} = (400 / 604) = 0.6622. In Figure 5-10(a), R_{div}= (4 / 204) = 0.0196, and in Figure 5-10(b), R_{div} = (204 / 604) = 0.3377. Using the R_{div} value, it is apparent that the vertices in Figure 5-9 are more divergent than those in Figure 5-10.

From these definitions, we can express the following relationship between R_{div} and R_{conv}:

$$\Delta M = \frac{M_{result}}{R_{conv}} = \frac{M_{source}}{R_{div}}$$

$$R_{div} = R_{conv} \frac{M_{source}}{M_{result}}$$

$$\frac{R_{div}}{R_{conv}} = R_d$$

In this section, several vertex-based metrics are developed to categorize forward derivation component graph vertices based on their historical usage within the data exploration session. The fundamental vertex-based metrics categories include source, result, terminal, quasi-terminal, landmark, sequential and k-sequential. The derived vertex-based metrics provide additional descriptions of convergent and divergent vertices, and means of ranking these types of vertices.

5.4.2 Derivation Path-Based Metrics

Derivation path-based metrics build on the vertex-based metrics by considering a larger and more complex environment. Instead of analyzing individual vertices and their associated derivations using local scope, a connected subgraph of the forward derivation component graph is analyzed using derivation path scope. We identify three broad classes

of analyses as *aggregate vertex-based measures*, *path-as-a-vertex measures* and *structural relationships*.

Aggregate vertex-based measures are summarizations based on the vertex-based metrics of the previous section. Once the vertex-based metrics for each vertex in a forward derivation path are determined, statistical measurements such as frequency distributions and ratios can be computed. For example, the number and percentage of landmark vertices or terminal vertices in a forward derivation path is critical information for the analyst in characterizing the importance of a portion of the data exploration session.

In *path-as-a-vertex measures*, the forward derivation path is viewed in the same way that a vertex is viewed in the previous section: the path is a "supervertex" that has fringe vertices connected by incident and exiting derivation edges, as shown in Figure 5-6. Thus, the vertex-based metrics developed in the previous section can now be applied to entire forward derivation paths. This requires a set-of data entity view of the session, which is not a focus of this research. Augmented definitions for the categories developed in the previous section would be needed.

Both aggregate vertex-based measures and path-as-a-vertex measures are limited, however, because they ignore the internal structure of the path. The *structural relationships* existing among derivation path vertices hold the patterns of interaction between the analyst and data, and are of paramount importance. We identify the following path-based metrics: derivation path *depth, breadth, ratio and cyclomatic complexity*.

***D**$_{path}$: Derivation Path Depth*

Historically, the standard metric on graphs is the distance metric *d(u, v)*: the number of edges between vertices *u* and *v*. The length of the shortest path between *u* and *v* is called the *path metric* (Buckley and Harary 1990). The derivation path depth $\mathbf{D}_{\text{path}}$ is the primary measurable quantity of a *simple* forward derivation path: the number of derivations from the start vertex to the terminal vertex. The level number indicates the depth of any simple path from the root vertex to any target vertex in the graph, a minimal number of derivations the data analyst performed to arrive at the target data entity vertex.

In general, however there may be multiple simple paths of different lengths between any two vertices, since distinct paths in the forward derivation component graph represent unique derivation sequences that traverse different regions of the data exploration space. For our metrics, this means (1) the forward derivation path contains a *vector* of depth values, one for each linearly independent simple path, and (2) $\mathbf{D}_{\text{path}}$ can be described in terms of the *maximum*, *minimum* and *average* simple path depth. Figure 5-11 shows three simple forward derivation paths between two vertices. The maximum path depth is 8, the minimum path depth is 1 and the average path depth is (1+3+8)/3 = 4.

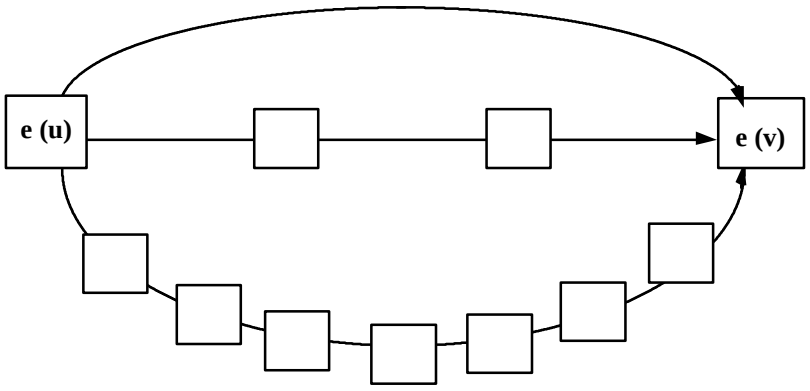


Figure 5-11. Determining the derivation path depth.

By calculating the distance between two data exploration states as the number of derivations between them, we have a relative measure that is only as good as the path taken. At any time, it is conceivable that another path of different length can be taken between the same two positions. Great care must be taken when choosing forward derivation paths for analysis.

$\mathbf{D}_{\text{path}}$ is thus a relative measure to the database or some intermediate vertex in the graph. It is affected by many factors, including analyst ability, data complexity, available tools and their interfaces. With this said, we offer the following Postulate to define a reasonable data exploration semantic based on $\mathbf{D}_{\text{path}}$:

Postulate 5.1 - Data Refinement Semantic:

D_{path} *relates the number of sequential derivations along a simple path to the degree of data refinement that a target vertex exhibits with respect to the goals of the data exploration session, and to its closeness to discovered knowledge. The greater the* ***D_{path}*** *value a target vertex has from some reference vertex, the more refined and closer to a unit of knowledge it is.*

The Data Refinement Semantic is equivalent to spatial zooming, but in the context of a data exploration session. This Postulate equates depth with data refinement. Forward derivation graphs with large average depth values would have more refined data sets that are closer to knowledge than other graphs over the same database that have smaller average depth values. In simple linear paths, the output of one derivation serves as the input to some follow-up derivation, so more work is being performed on a data set, which could indicate that knowledge is being gained from it.

***B**$_{path}$: Derivation Path Breadth*

The derivation path breadth is the complement to the derivation path depth. It is the number of vertices at a particular depth *d*, and for general forward derivation paths, is represented by a vector, just like $\mathbf{D}_{path}$. As with $\mathbf{D}_{path}$, $\mathbf{B}_{path}$ is bounded by the maximum breadth at any one depth. The derivation path breadth can vary widely throughout the graph, because derivations are added to the graph as the analyst sees fit; while there may be a general plan, this may vary as the database is transformed. The nature of data exploration requires derivations and explicit state changes to be applied in a non-deterministic fashion, as the data analyst interacts with the data to uncover its structure and meaning.

Explicit state changes give rise to exploration breadth, as the analyst revisits certain interesting vertices to apply new derivations, and thereby supplement the evolving mental model of the data. This activity transforms the linear derivation sequence into the derivation graph, and increases the $\mathbf{M}_{source}$ and $\mathbf{M}_{visit}$ vertex metrics for those vertices. Some of those vertices go on to become landmark vertices if their vertex-based metrics surpass a certain level.

As with $\mathbf{D}_{path}$, we can postulate a semantic based $\mathbf{B}_{path}$ such that we can further describe data exploration sessions:

Postulate 5.2 - Data Coverage Semantic:

***B**$_{path}$ relates the number of branching derivations that exist at a particular depth to the degree of data coverage of some subspace of the data exploration space. The greater the **B**$_{path}$ value, the larger the subspace of the data exploration space that is covered by the analyst.*

The Data Coverage Semantic is equivalent to spatial panning, but in the context of a data exploration session. This Postulate equates breadth with data coverage, or exploration in its purest sense. Exploration implies the coverage of new territory, and here it is a departure from the current *direction* in the data exploration space. Forward derivation paths with large values for derivation breadth can be considered more exploratory than those that have smaller values for the same database.

Proving Postulates 5.1 and 5.2 will require a great deal of user testing of explorations of various types of databases, under numerous conditions and with different kinds of exploration tools. This is clearly beyond the scope of this research, and probably a far-off goal to be reached in the distant future. The two Postulates make good sense, though, based on the principles we have outlined and our common sense.

***R_{path}**: Derivation Path Ratio:*

The derivation path ratio is a high-level indicator of the structure of the forward derivation path, given by $\mathbf{R}_{path} = \mathbf{B}_{path} / \mathbf{D}_{path}$. $\mathbf{R}_{path} > 0$ because a forward derivation path must exist to use this metric. If $0 < \mathbf{R}_{path} < 1$, then $\mathbf{D}_{path} > \mathbf{B}_{path}$, and there are more coverage than refinement derivations (i.e., more exploration is being performed). If $\mathbf{R}_{path} > 1$, then $\mathbf{D}_{path} > \mathbf{B}_{path}$, and there are more coverage than refinement derivations. If $\mathbf{R}_{path} >> 1$, there would be a great deal of refinement occurring, and if $\mathbf{R}_{path} = \mathbf{D}_{path}$, then $\mathbf{B}_{path} = 1$ and the forward derivation component graph would be a linear sequence of derivations.

A key aspect of the data exploration process is to traverse as little of the data exploration space as possible, while still being highly confident about the validity of the discovered knowledge. Different exploration techniques undoubtedly have different $\mathbf{R}_{path}$ values associated with them. Different stages of the data exploration session also have

different $\mathbf{R}_{path}$ values. We would expect early stages of the session to have derivation paths with $\mathbf{R}_{path} < 1$, as there would be more coverage, or *exploratory* interactions being performed. As the session progresses, we would expect to see more derivation paths with $\mathbf{R}_{path} > 1$, as more refinement or *focusing* interactions are being performed.

***C**$_{path}$: Derivation Path Cyclomatic Complexity:*

Though the forward derivation component graph does not show cycles directly, they do exist in a data exploration session, predominantly in the form of explicit state changes. Cycles understandably increase the complexity of the graph, as they lead to more branching structures. The derivation path cyclomatic complexity metric is a measure of the overall complexity of the path. It is an adaptation of Berge's cyclomatic number (Berge 1973) in the spirit of McCabe (McCabe 1976) who applied this measure towards validating the structured programming methodology. This measure is given by the following formula: $\mathbf{C}_{path} = |D| - |E| + 2p$, where in our model $|D|$ is the number of derivation edges, $|E|$ is the number of entity vertices and p is the number of connected components in the forward derivation path.

Though the forward derivation component graph has only a single connected component, each branch in the graph is considered analogous to a subroutine call in McCabe's work. Thus, the degree of branching in the forward derivation component graph directly affects the cyclomatic complexity. As with McCabe's measure, $\mathbf{C}_{path}$ yields the number of linearly independent paths in the forward derivation component graph.

When comparing forward derivation paths, the path with the higher $\mathbf{C}_{path}$ value can be considered more complex because it involves more explicit state changes and more branching.

5.4.3 Metrics Summary

The metrics described in this section compute historical measures on individual vertices and entire forward derivation paths. We have attempted to define a minimal set of metrics that are meaningful, meaning they are based on structure and not on statistics.

Vertex-based metrics are one-dimensional, since they pertain to the immediate neighborhood of individual vertices. Adhering to the principles of data exploration state and local scope, these metrics answer the question “what am I looking at?”, with respect to the session. They reply with a categorization of the vertex (with classes such as landmark, terminal, quasi-terminal, quasi-source, sequential, etc.), as long as there is some statistical information available (such as the total number of vertices and the distribution of metric values across all vertices). Vertex-based metrics are instrumental in defining the Conservation of State Change Property, which is in many respects similar to the conservation of mass, momentum and energy in physical systems.

Path-based metrics are two-dimensional, because they pertain to the forward derivation path, which is a metric structure for the data exploration process, ordered by depth and breadth levels. These metrics speak more towards answering the question “where am I?”, as they describe the derivational structure of the path. Path-based metrics are instrumental in defining the two Postulates that infer meaning from certain important graph patterns.

As mentioned earlier, the metrics defined in this section are static - they are computed from snapshots of the data exploration session without regard to the sequencing of forward derivations. In the following section, additional dynamic measures are defined that are based on derivation sequencing and changing data values.

5.5 A Data Exploration Calculus

Calculus is a method of calculation used to analyze rates of change. Here we consider the calculus defined on the Real numbers, but applied to data derivations. Other, more familiar applications of calculus include determining tangents to curves (derivatives), areas of regions and volumes of solids (integrals), and computing minimum and maximum values. In this section, we define a data exploration calculus that is based partly on the forward derivation component sequence, and partly on the forward derivation component graph. The data exploration calculus is used to answers the questions “where am I going?” and “at what rate am I going?”.

5.5.1 Relevant Assumptions

In the data exploration process, the changing quantities are the regions of the data exploration space traversed by the analyst. There is naturally an underlying temporal basis to these changing quantities: each data derivation marks a distinct point in time, so data changes can be related to time. There is also a derivational basis: the change in data is due to some forward derivation; backward derivations are explicit state changes and identity derivations that do not alter data.

In defining the data exploration calculus, the following conditions must apply:

1. *Data Exploration Time*:
    a) starts at zero when the session begins
    b) is an increasing Real number
    c) stops if analyst is distracted (e.g., eating, sleeping, performing other tasks)
    d) resumes when analyst resumes session

2. *Data Exploration Time* consists of:

a) time spent by the analyst in analyzing a visualization (t_a)

b) time spent by the analyst in formulating a new data derivation (t_f)

c) time spent by the analyst in applying the derivation (t_p)

d) time spent by the computer in computing and presenting the result (t_c)

Thus far, we have been restricting timestamps on data entities and derivations to be positive integers, though they are defined as positive Natural numbers (cf. Definition 4.2 and 4.3). The data exploration calculus puts the Natural numbers to use, as they carry more precision. Condition 1(c) is an idealized assumption, unique to data exploration time.

Conditions 2(a), (b) and (c) account for the cognitive activity the analyst must undertake in order to perform a data derivation. Using Norman's model (cf. Section 2.2.2), this activity includes *perceiving* the visualization, *interpreting* its meaning, *evaluating* the meaning with respect to some expectation, forming the *intention* of the follow-up derivation, *specifying* the derivation and finally *executing* the derivation at the user interface. We call the time required to perform these mental tasks the *analysis time*, given by $t_A = t_a + t_f + t_p$.

5.5.2 Calculus in One Dimension

Taking a simplistic view of the data exploration process as an ordered sequence of data exploration states, having unknown internal structure, we can approximate data exploration by a function *f* on the open interval (t_0, t_z), that maps data exploration time with data exploration state. We can safely assume that there is some minimal temporal interval between successive states, due to t_A such that the function is *strictly increasing*, meaning that for $f(t)$, if $t_1 < t_2$, then $f(t_1) < f(t_2)$, as shown in Figure 5-12.

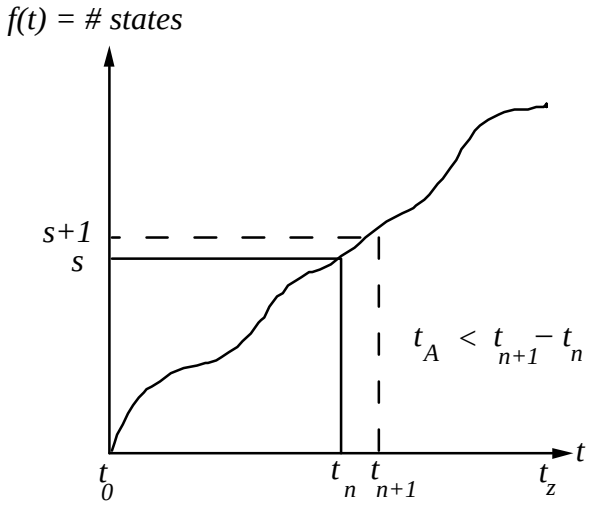


Figure 5-12. A one-dimensional view of data exploration.

The progression from one state to the next state takes the same number of mental activity steps (that make up t_A), but the steps can take different lengths of time. Similarly, the time component of each interaction due to t_c varies with the task and data to be transformed. The greater the time between steps and the longer the time the analyst spends analyzing a particular visualization will flatten out the graph in Figure 5-12. Intuitively, we expect data exploration to be a temporally continuous process, when viewed as an alternating sequence of visualizations and database operations, and given the conditions set forth above. We can strengthen this claim.

We can define *f*(t) to be composed of 3rd order polynomials that are piecewise smooth in their second derivatives (such as cubic splines), making the function differentiable. So, *f*(t) is defined for all values on the open interval (t_0, t_z), the limits at any point *p* on the curve also exist because of the minimal temporal distance t_A between adjacent time points, and there is connectivity between all data points that are defined to

be on the curve. Because f(t) is differentiable, it is also continuous on the open interval (t_0, t_z).

Having a differentiable and continuous function allows the methods of calculus to be applied to determine the rates of change of data exploration states with respect to time. Another reason for specifying a 3rd order polynomial is to allow differentiation to occur twice, yielding acceleration information. Average *speed of interactions* is calculated as $\bar{s} = \frac{\Delta s}{\Delta t}$, and instantaneous speed of interactions is calculated as $s' = \lim_{\Delta t \to 0} \frac{\Delta s}{\Delta t} = \frac{ds}{dt}$ (derivations / second). Similarly, the magnitude of the average *interaction acceleration* is calculated as $\bar{a} = \frac{\Delta \bar{s}}{\Delta t}$ and the magnitude of the instantaneous interaction acceleration is calculated as $a' = \lim_{\Delta t \to 0} \frac{\Delta s'}{\Delta t} = \frac{d^2 s}{dt^2}$ (derivations / second^2).

Using the calculus, we now have a way of determining, at a very high level, how easily an analyst is exploring a database. The one-dimensional calculus allows forward derivation component sequences to be analyzed, where the time changes between successive derivations relates to how fast the exploration is proceeding. For example, in Figure 5-13 several forward derivation component sequences are shown graphically. A constant derivation speed is shown in Figure 5-13(a), and a faster constant derivation speed is shown in Figure 5-13(b). Figure 5-13(c) shows a forward derivation component sequence that initially accelerates, then decelerates.

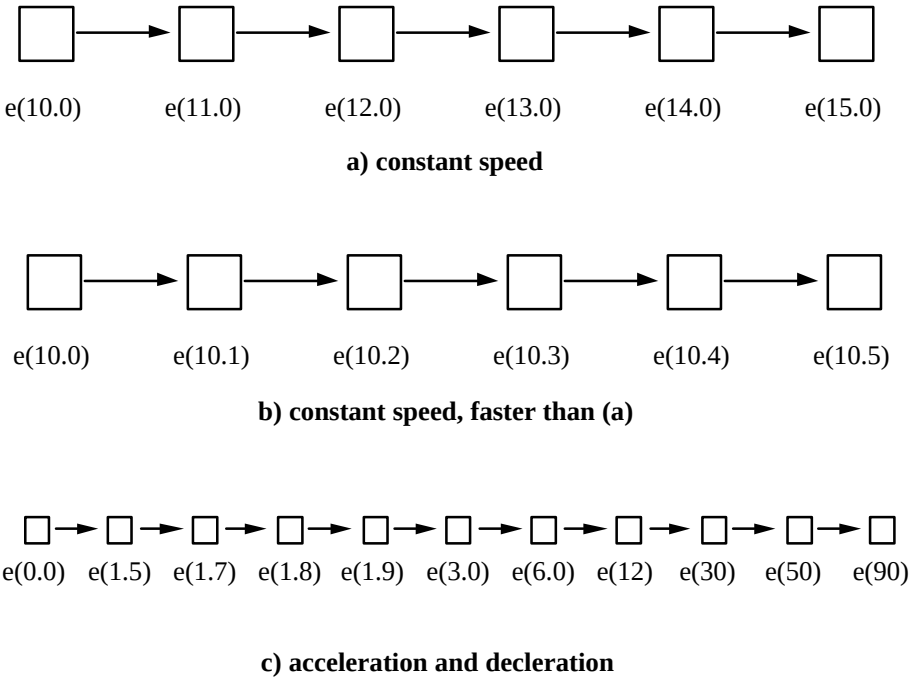


Figure 5-13. One-dimensional derivation speed and acceleration examples.

The calculus does not yield the underlying reasons for the speed of interactions, however, as Figure 5-13 shows effects, not causes. Possible reasons include analyst abilities, the presentation method, the available exploration tools and user interface considerations. Controlled experimentation is required to determine the reasons for various exploration scenarios.

5.5.3 Calculus in Multiple Dimensions

Though we have a basic notion of the dynamics of the exploration session as a one-dimensional signal, we have not considered the contribution of the data being explored. If we now take the contents of the data exploration state into consideration, we can determine the *direction* of the session at any vertex or along any derivation path, with respect to the data exploration space. If we know what attributes of $\mathbf{S}_{dexp}$ are manipulated

by a forward derivation, we can easily determine direction. If we also know *how* those attributes are manipulated, we can compute distance, velocity and acceleration with respect to the data.

Here, we focus on the former condition, when we only know what attributes of $\mathbf{S}_{dexp}$ are being manipulated by a derivation. While this can be viewed as a limitation (we cannot yet compute data distances, etc.), it more importantly fits into the type of model we are defining; we are describing the data exploration process in a general fashion, devoid of any reference to the data being explored. By only using the names of the attributes, we still honor this requirement, while gaining additional descriptive information. Concrete instantiations of the GDE model will have to completely define their unique $\mathbf{S}_{dexp}$, and therefore allow distances and derivatives with respect to the data to be performed. Chapter 6 contains relevant examples.

The attributes manipulated by each derivation express a derivation *unit vector* $\hat{\mathbf{s}}$ that signifies the direction within $\mathbf{S}_{dexp}$ the derivation takes the analyst. For example, Figure 5-14 shows a hypothetical forward derivation component graph, that highlights the derivation unit vectors associated with derivation paths. Timestamps are omitted for clarity.

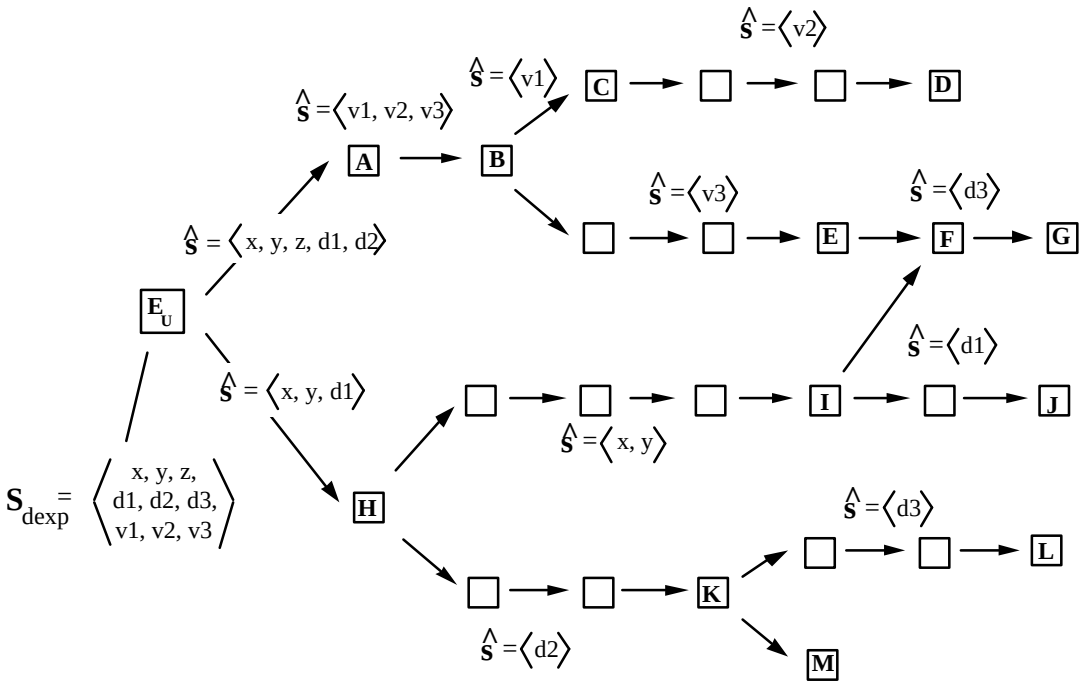


Figure 5-14. n-dimensional derivation directional unit vector example.

In Figure 5-14, the database (E_U) has a 9-dimensional data exploration space consisting of three spatial attributes {x, y, z}, three non-spatial data attributes {d1, d2, d3} and three visualization attributes {v1, v2, v3}. In the forward derivation component graph, relevant vertex timestamps are replaced with a letter {A, ... , M}, though their temporal ordering is not important for this example. Each edge represents a distinct derivation, except for the two edges that create vertex F. Associated with each derivation path (that may consist of one or more edges) is a derivation unit vector that identifies the attributes of $\mathbf{S}_{dexp}$ that are used in the derivation specification. The derivation unit vectors are summarized in Table 5-3, where each derivation path is labeled, along with the directional unit vector components, and a comment describing the data manipulated.

Path	Attributes	Comments
<E_U, A>	x, y, z, d1, d2	3 spatial, 2 non-spatial data attributes
<A, B>	v1, v2, v3	3 visualization attributes
<B, C>	v1	single visualization attribute
<C, ..., D>	v2	single visualization attribute
<B, E>	v3	single visualization attribute
<(E,I), F>	d1, d3	2 non-spatial data attributes
<F, G>	d3	single non-spatial data attribute
<E_U, H>	x, y, d1	2 spatial, 1 non-spatial data attribute
<H, ..., I>	x, y	2 spatial data attributes
<I, ..., J>	d1	single non-spatial data attribute
<H, ..., M>	d2	single, non-spatial data attribute
<K, ..., L>	d3	single, non-spatial data attribute

Table 5-3. The derivation directional unit vectors of Figure 5-14.

Some derivation paths manipulate several attributes at once, while others manipulate only a single (the lower bounds on the length of the direction vector). Note that since the forward derivation component graph is not a metric space for the data exploration space, descendant derivations are not restricted to be a sub-vector of some ancestor's directional unit vector; there is no containment relationship among vertices along any simple path, with respect to the derivation direction vector. Also, when incorporating actual timestamp values, derivational velocity and acceleration can be determined in a similar manner as Figure 5-14 shows.

5.5.4 Data Exploration Calculus Summary

As its name implies, the data exploration calculus captures the dynamic nature of data exploration sessions. The one-dimensional data exploration calculus allows scalar speed and acceleration to be established from the forward derivation component sequence. The n-dimensional calculus allows vector velocity and acceleration to be computed. This enables correlations to be drawn between speed and directional unit vectors to yield further insights into the data exploration process. For instance, in locations or regions of the graph where the derivation speed has abruptly changed, additional information about the derivation and its history is available in the derivation unit vectors. Perhaps it is a particularly cumbersome visualization technique that is slowing the progress down, or that a too-large and complex data subset is being explored. The data exploration calculus helps address such questions about the process.

Not only does the data exploration calculus capture the time-varying behavior of the data exploration session, it provides indicators into the changing attributes. Additional meaning to the static data exploration metrics of the previous section can then be imparted. Landmark vertices may contain certain data attributes. Certain oft-traversed paths through the data exploration space become “interaction corridors” or “interaction highways”, depending on the speed at which derivation paths traverse the space. By focusing on the changing data over time, the data exploration calculus complements the static metrics, and provides a solid foundation for describing *data continuity*, the topic of the next section.

5.6 Data Continuity

The two key results of the previous section are that (1) data exploration is a *continuous* process in time, when considering the derivation sequence and given certain

restrictions on the nature of data exploration time, and (2) data exploration is *piecewise continuous* in time when considering the data under exploration. As the data exploration session progresses, the forward derivation component graph is constructed incrementally, as a result of the data analyst selecting existing data entities to derive further, deriving new data entities from the data store, or a combination of both. *Data continuity* is a description of what happens between successive data derivations, in the sense of what data entities of the current data exploration state are "preserved" by being derived further. Specifically, it is the relationship between the image of a data derivation at time *t*, and the preimage of a data derivation at time *t+1*.

After each derivation is a decision point. Based on the analyst's perception of the current data exploration state (which includes all knowledge about the data attained thus far), the data analyst must decide how to proceed with the exploration. Among the possible alternatives are to continue along and derive exclusively and entirely from the current data exploration state, to derive exclusively from a subset of the current data exploration state, to incorporate additional data by deriving from previous data exploration states, to derive exclusively from previous data exploration states, or derive from the data store.

Figure 5-15 shows the types of data continuity possible between temporally adjacent data exploration states. In each example, a set of data entities is shown that belongs to one or both data derivations, d(t) and d(t+1). Shaded data entities are those that are preserved across both derivations. The data continuity between derivations can be *maximally continuous* as shown in Figure 5-15(a), *discontinuous* as shown in Figure 5-15(b), or *partially continuous* as shown in Figures 5-15(c) through 5-15(e).

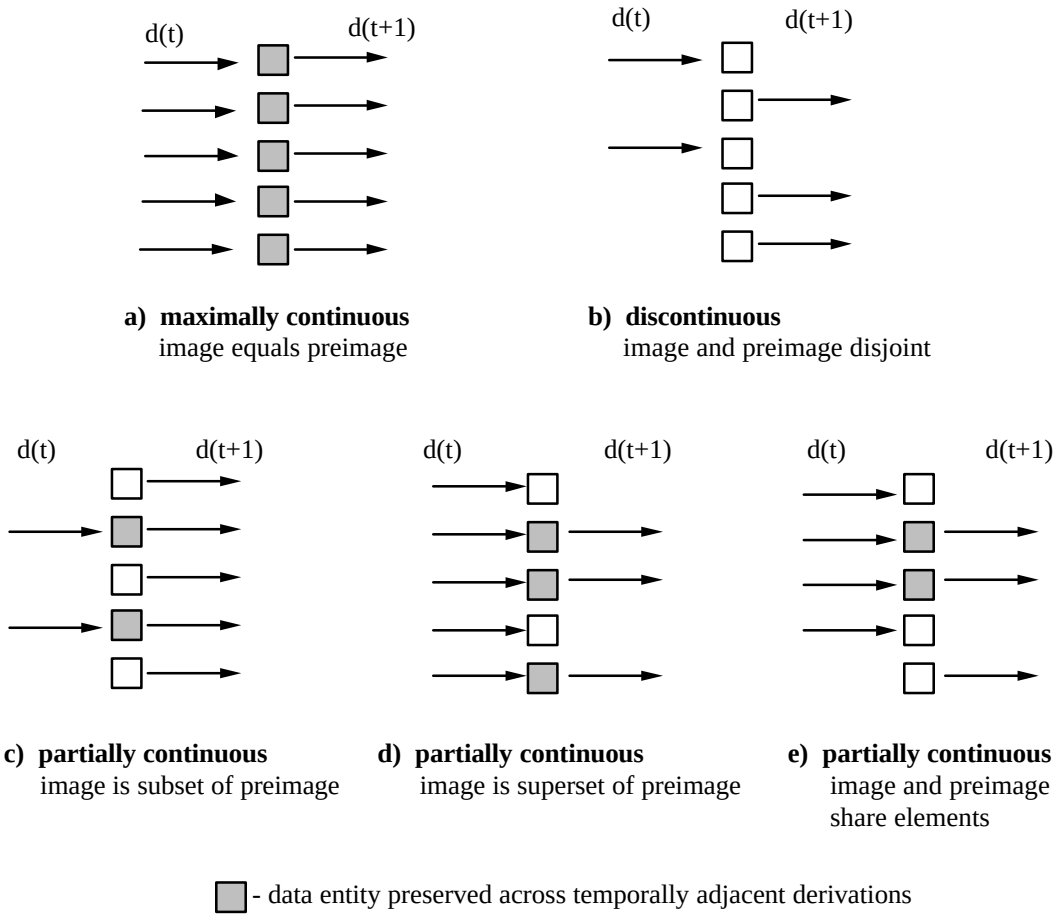


Figure 5-15. Examples of data continuity.

At one extreme, in the maximally continuous transition of Figure 5-15(a), the image of d(t) equals the preimage of d(t+1). This means that the analyst decided to continue exploring the current set of data entities. One can infer that there might be something useful or important the data being explored, because the analyst chose to continue exploring it. Using the notation introduced in Section 4.3, $E^o(t) = E^i(t+1)$.

At the other extreme, in the discontinuous transition of Figure 5-15(b), the image and preimage are disjoint - they share no elements: $E^o(t) \cap E^i(t+1) = \{\varnothing\}$. This means that the analyst decided to explore elsewhere, and abandon the current data exploration state because nothing useful or interesting was found. Note, however, that the analyst

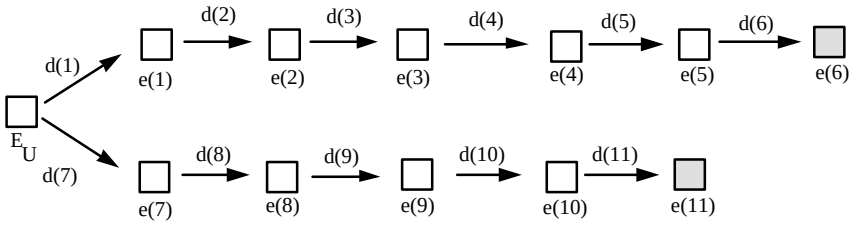
could revisit this state at a later time, and derive from it further. Such a situation will be elaborated upon later in this section.

In partially continuous transitions, there are several alternatives. The image of d(t) can be a subset of the preimage of d(t+1) as in 5-15(c): $E^o(t) \subseteq E^i(t+1)$. The image can be a superset of the preimage as in 5-15(d): $E^i(t+1) \subseteq E^o(t)$. The image can be neither subset nor superset of the preimage, but still have elements in common, as in 5-15(e): $E^o(t) \not\subset E^i(t+1)$, $E^i(t+1) \not\subset E^o(t)$ and $E^o(t) \cap E^i(t+1) \neq \{\varnothing\}$.

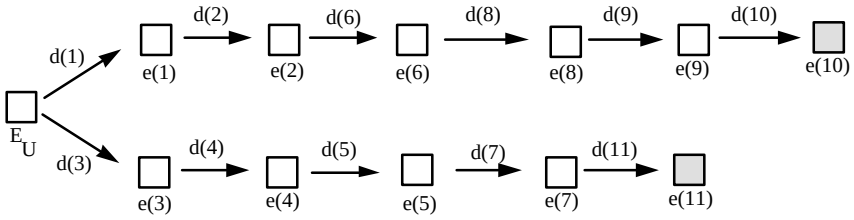
The concept of data continuity can be broadened from considering the transition between two data derivations to considering the transitions along a data derivation path. Some paths lead to "dead ends", terminal vertices deemed useless by the data analyst, such as Figure 5-15(b), where the current data exploration state is not used as the preimage of any subsequent derivation. Other paths may prove more fruitful, eventually leading to tagged terminal vertices. The tag and comment fields of terminal vertices are the true arbiters between useful and useless directions. Granted, useless here is a relative term, because all paths traveled increase the knowledge gained by the analyst, and what was useless might become useful in the future, as more knowledge about the data is attained.

Data derivation paths exhibiting *temporal* continuity are said to be *temporally continuous*. In general, however, there is no such continuity requirement among the states of a derivation path. The dynamic nature of data exploration allows changes in direction and the revisitation of vertices and paths, as more knowledge is gained about the data. Such paths that are revisited are considered to be *piecewise temporally continuous*. Figure 5-15 shows two simple derivation path examples that are structurally identical, but one being temporally continuous, and the other piecewise temporally continuous.

The temporally continuous derivation path of Figure 5-16(a) contains two distinct derivation sub-paths, one creating terminal vertex e6 and the other e11. In both sub-paths, the image of each derivation is the preimage of the subsequent derivation, with the exception of derivation d6. The analyst first derives data entities to eventually create e6, which is then abandoned in favor of starting a new path that eventually creates e11. Drawing an analogy from graph theory, temporally continuous derivation paths are similar to a depth-first traversal of a portion of the data exploration space, constructing a forward derivation component graph in the process.



a) temporally continuous derivation paths



b) piecewise temporally continuous derivation paths

Figure 5-16. Temporal data continuity examples.

The piecewise continuous path of Figure 5-16(b) also shows two distinct derivation sub-paths, one creating terminal vertex e(10) and the other e(11). In this example, the analyst constructs the two sub-paths simultaneously by occasionally alternating derivations between them. This type of construction is a variant of a breadth-

first graph traversal, with some depth-first (i.e., temporally continuous) substructures interspersed among the alternating derivations.

In summary, data continuity is a temporal-based description of the data derivation process. It focuses on the overlapping of data *between* derivations, and therefore yields some insights to what data is important to the data analyst.

5.7 Chapter Summary

This chapter presents a number of important concepts and measurements that can be used to characterize data exploration sessions. By applying the data exploration metrics and calculus to various data exploration scenarios, generalizations about the data exploration process can then be made. Before the metrics and calculus could be developed, though, several foundational concepts had to be developed.

The *data exploration space* is the data metric space that provides the data context in which the data analyst works. *Data exploration state*s are distances between the data sets that make up each state. Measuring data distances cannot be performed in the GDE model, however, as there is no knowledge of the actual data.

Just as the data exploration space can be complex, so can the data exploration process space. Even simple forward derivation component graphs (which are already reduced representations of the session because they do not show identity and backward derivations) can be complex. In presenting forward derivation component graphs, some heuristics can be applied to configure the graph in a comprehensible manner. These heuristics organize the graph based on *derivation depth* and *breadth*, and reduce visual clutter by additional summarizations and compressions.

The fundamental forward derivation component graph structures (vertices, edges and paths) can be "packaged" such that they describe a context for performing analyses. *Data exploration scope* echoes the parallels data exploration has with programming languages, as the fundamental graph structures are similar to programming statements and procedures. Additionally, another parallel between data exploration and programming can be seen in the *cyclomatic complexity* metric.

Data exploration metrics use data exploration scope to provide context, and are based on graph structures, not statistics. Static metrics (the vertex-based and path-based metrics) focus on graph topologies, while dynamic metrics (the data exploration calculus) focus on the temporal aspects, differentiability and continuity of the process. As a by-product of the metric development, the *Conservation of State Change Property*, the *Data Refinement Semantic* and the *Data Coverage Semantic* were defined. These definitions speak towards deriving meaning from structures and patterns within the forward derivation component graph.