Benthic: Designing Relational Traversal Structures to Enhance Diagram Accessibility

by

Catherine Mei

S.B. Electrical Engineering and Computer Science and Brain and Cognitive Sciences, MIT, 2023

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Catherine Mei
Department of Electrical Engineering and Computer Science
May 17, 2024

Certified by: Arvind Satyanarayan
Assistant Professor of Computer Science, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Benthic: Designing Relational Traversal Structures to Enhance Diagram Accessibility

by

Catherine Mei

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Diagrams are data structures for problem-solving and communication because they allow users to formalize and analyze complex concepts through spatial relations. However, their visual nature presents significant accessibility challenges for blind and low-vision users who rely on screen readers. Existing methods for making diagrams accessible often fall short, providing only superficial overviews and lacking detailed, navigable structures. This paper introduces Benthic, a system for generating intermediate representations and depicting relational information in diagrams. Benthic provides an interface that allows screen reader users to navigate the diagram data structure. Benthic uses a *hypergraph* traversal structure, where diagram nodes are grouped by *hyperedges* that represent diagram *relations*. These *relations* are presented in the screen reader interface according to their *priority* (or visual salience), allowing screen reader users to traverse the information similarly to how sighted users might view the diagram. Additionally, users can explore diagrams at various levels of detail by choosing to navigate high-level relations or more detailed relations based on their needs. We evaluate Benthic's effectiveness through three comparative case studies with existing diagram accessibility systems. Benthic aims to create a design space of traversal structures that will allow blind and low-vision users to leverage the same affordances available to sighted users, enabling intuitive interaction and comprehensive understanding of diagrams.

Thesis supervisor: Arvind Satyanarayan
Title: Assistant Professor of Computer Science

Acknowledgments

I am deeply grateful to the many individuals who have mentored, supported, and collaborated with me throughout my research journey.

First, I would like to express my heartfelt thanks to my supervisor, Arvind Satyanarayan, and my mentor, Josh Pollock. Your unwavering support, insightful feedback, and willingness to troubleshoot with me have been crucial to the success of this project. I have learned so much from working with you, and your genuine concern for my well-being means a lot to me. Your guidance has fostered both my personal and professional growth, making this research experience incredibly rewarding and enjoyable.

A special thank you to my MEng buddy, Grace Huang, for being a great friend. Your support has helped me power through the year, and I am really glad to have met you through this research experience.

I would also like to extend my gratitude to the entire MIT Visualization Group. Your continuous support, thoughtful feedback, and enthusiasm for my research have been incredibly motivating. I have cherished group social events, where I found a genuine sense of belonging. The friendly and supportive atmosphere has made my research journey even more fulfilling. Thank you all for making this experience so rewarding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Diagrams are powerful tools for computation and communication, enabling users to externalize and navigate complex conceptual structures through spatial relationships [1]. They are essential across various domains for problem-solving and reasoning. However, this visual-centric medium poses significant challenges for blind and low-vision users who rely on screen readers, or software programs that convert screen content into synthesized speech or Braille.

Screen readers typically handle data visualizations as static images, with accessibility provided through alternative text (alt text) and captions. This method, however, presents several challenges such as determining the appropriate content for alt text, establishing the order of information presentation, and enhancing utility for blind and low-vision users encountering unfamiliar visualizations [2], [3]. Morris et al. emphasized the need to consider properties like interactivity, stability, and personalization in presenting alt text [4]. However, existing alt text methods for visualizations often provide only a broad overview, preventing screen reader users from explore data comprehensively. This gap highlights the need for a more nuanced screen reader experience with well-structured navigation and detailed semantic content [5].

Additionally, the interaction with screen readers varies among users based on their familiarity with these tools. Hence, it's crucial to allow for customization in textual descriptions,

enabling self-guided data exploration at varying levels of detail. Key characteristics such as presence, verbosity, ordering, and duration of descriptions can significantly reduce cognitive load and enhance user experience [6]. More comprehensive and intuitive traversal schemes are necessary to provide blind and low-vision users with the same affordances that sighted users gain from visual diagrams.

Existing systems, such as Olli [7], Data Navigator [8], and accessibility initiatives in STEM fields like chemistry [9], have made notable efforts to enhance diagram accessibility. However, these approaches often come with limitations, such as unclear traversal patterns and insufficient support for visually overlapping diagram elements.

We present Benthic, an intermediate representation and screen reader interface designed to make diagrams more accessible and intuitive for screen reader users. Benthic uses hypergraph traversal structures, which are relaxed tree-like structures where a child node can have multiple parent nodes [10].

A key idea in Benthic is the concept of *relations*, or a grouping of diagram elements. Existing research has examined diagrams by understanding relations between diagram elements. For instance, Larkin and Simon explored the computational benefits of diagrams by constructing data structures based on analyzing relations and groupings within diagrams [1]. In Benthic, *relations* are represented as hyperedges that group multiple nodes within the traversal structure. Because of Benthic's hypergraph and hyperedge representation, Benthic is able to represent overlapping relations in graphic representations.

Benthic's traversal structures prioritize relations based on visual salience, allowing screen reader users to access the most visually prominent elements first. This ordering mimics the natural viewing sequence of sighted users, creating a more parallel and intuitive experience for screen reader users as they navigate and analyze diagrams. Additionally, Benthic's traversal structures provide abstraction boundaries, enabling users to explore diagrams at various levels of detail. Depending on their task and the amount of information needed, users can choose to traverse diagrams using either higher-level relations to gain a broad overview or

through more detailed, primitive relations.

We evaluate Benthic traversal structures through three case studies: a comparison between the Benthic and Data Navigator [8] traversal structure for a stacked bar chart graphic created by Highcharts [11], a comparison between the Benthic and Data Navigator traversal structure for a set diagram created by Penrose [12], and a walkthrough of the Benthic traversal structure for a physics pulley diagram.

Future work on Benthic can advance its utility through several promising directions. One possibility involves developing methods to prune and tailor the traversal structures to specific tasks. For example, users engaged in complex problem-solving may require detailed diagram structures, whereas casual users might benefit from broader overviews. This can be achieved by leveraging relational information to merge related graph nodes. Another direction to explore involves developing adapters for various diagramming frameworks to enable automatic extraction of relational information. This would allow outputs from popular visualization tools to be compiled to Benthic traversal structures, enabling automatically accessible graphic representations. By pursuing these directions, Benthic can significantly broaden its impact and adoption across diverse applications.

Chapter 2

Related Work

2.1 Accessibility for Graphic Representations

Numerous efforts have been made to improve the accessibility of diagrams and data visualizations, with a special emphasis on creating traversal structures that are intuitive and user-friendly for visually impaired individuals. Within this field, three distinct systems - Olli, Data Navigator, and a system for chemical molecule accessibility - have each explored unique approaches to enhance the traversal experience.

2.1.1 Olli

Olli is an open-source JavaScript library that enhances web visualization accessibility for screen reader users. Olli works by converting various visualization specifications into a unified OlliVisSpec. This intermediate representation is then rendered as an HTML tree view, allowing screen reader users to navigate the rendered structure using arrow keys. The intermediate OlliVisSpec representation contains information for the structural and hierarchical relationships in the visualization, as well as information on the visual elements in the original visualization [7]. A limitation of Olli's traversal structure is its tree-based design, which requires strict parent-child relationships. Since every child must only have

one parent, Olli performs poorly when representing complex data visualizations that do not conform to a hierarchical model.

2.1.2 Data Navigator

Data Navigator employs a dynamic graph traversal structure to construct a range of navigable data structures including lists, trees, graphs, and spatial relationships, thus enabling more effective navigation in data visualizations. This flexibility extends to a variety of input modalities, accommodating screen readers, keyboards, speech, gesture detection, and even custom assistive devices. Data Navigator generates a graph traversal structure from a visualization by utilizing a node-edge graph framework. Nodes in Data Navigator are designed as objects containing a set of edges. Each edge comprises a minimum of four pieces of information: a unique identifier, a source, a target, and navigation rules [8]. Navigation rules provide the logic for how users navigate through data visualizations; for example, pressing "Backspace" could navigate the user up towards the x-axis on the visualization.

While the flexibility to define custom navigation rules could enhance user experience, it also introduces an element of ambiguity. Specifically, the interpretation of navigation commands in the graph traversal structure, such as what constitutes a "backward" or "forward" movement, is not inherently defined. This is particularly confusing for complex visualizations where directional navigation isn't intuitively apparent. Therefore, while Data Navigator allows the user to traverse many diagram relations, it's unclear how to formalize the navigation rules to ensure they align with the intuitive understanding of the visualization's structure.

2.1.3 Chemical diagram

Accessible visualizations have also been explored in other STEM areas, like chemistry. Sorge et al. introduced a system to make chemical diagrams accessible for visually impaired learners by converting chemical diagrams into a Chemical Markup Language (CML) format and then into annotated Scalable Vector Graphics (SVG). The SVG is then rendered on a browser to

allow for interactive exploration supported by various platforms and screen readers [9].

The accessible chemical diagram system uses an abstraction tree with a four-level hierarchy. The root symbolizes the molecule, and the first level is a graph with vertices representing atomic or sub-ring graphs (like functional groups or benzene rings). Sub-ring graphs can have atomic graphs as children. Users navigate this abstraction tree structure using vertical and horizontal keys. The "Down" and "Up" keys move the user to a child graph's first node or back to the parent graph, respectively. The "Right" and "Left" keys enable horizontal movement within a graph, following the elements' ascending or descending order [9]. This design facilitates structured navigation through the diagrams. However, a limitation occurs when a subgraph belongs to multiple chemical relations, causing ambiguity in vertical navigation due to unclear parent node definition.

2.2 Bluefish: Relation Based Diagramming Frameworks

Benthic's approach to creating screen reader traversal structures is inspired by the Bluefish diagramming framework, a component-based library that allows users to create diagrams using flexible and declarative building blocks called *relations* that allow users to create complex diagrams without micromanaging every detail of the layout.

A key innovation of Bluefish is its use of a compound graph scenegraph structure. Unlike traditional tree-based scenegraphs, which represent diagrams in a strictly hierarchical manner, Bluefish's compound graph captures both hierarchical and adjacent relationships between nodes [13]. This allows for more natural and accurate representation of complex diagrammatic relationships where elements can participate in multiple relationships simultaneously. Benthic traversal structures build on Bluefish's novel approach to diagrammatic representation.

Chapter 3

Usage Scenario

In this section, we will examine how the Benthic traversal structure can be used to represent the stacked bar chart depicted in Figure 3.1 from the Highcharts Charting Library [11]. This chart illustrates the trophies won by four English football teams across three competitions. The x-axis groups the data by football team, with each bar representing the cumulative trophies won by each team. Each bar is divided into three segments, each corresponding to the trophies won in a different competition. These segments are color-coded by competition, as indicated by the legend.

3.1 Benthic Traversal Structure Walk-through

A sighted user may interpret the information in the graphic in various ways. They might begin by examining the bars on the x-axis to determine the number of trophies won by each football team. Alternatively, they could focus on the trophies won in each competition, analyzing the data through the color groupings indicated in the legend. Each segment in the bar chart is part of two groupings: the football team bar and the competition color grouping.

Figure 3.2 illustrates the Benthic hypergraph traversal structure for the stacked bar chart. Benthic's hypergraph is a relaxed tree structure that removes the constraint that each child node must have only one parent. This allows child nodes to be contained by multiple parents,

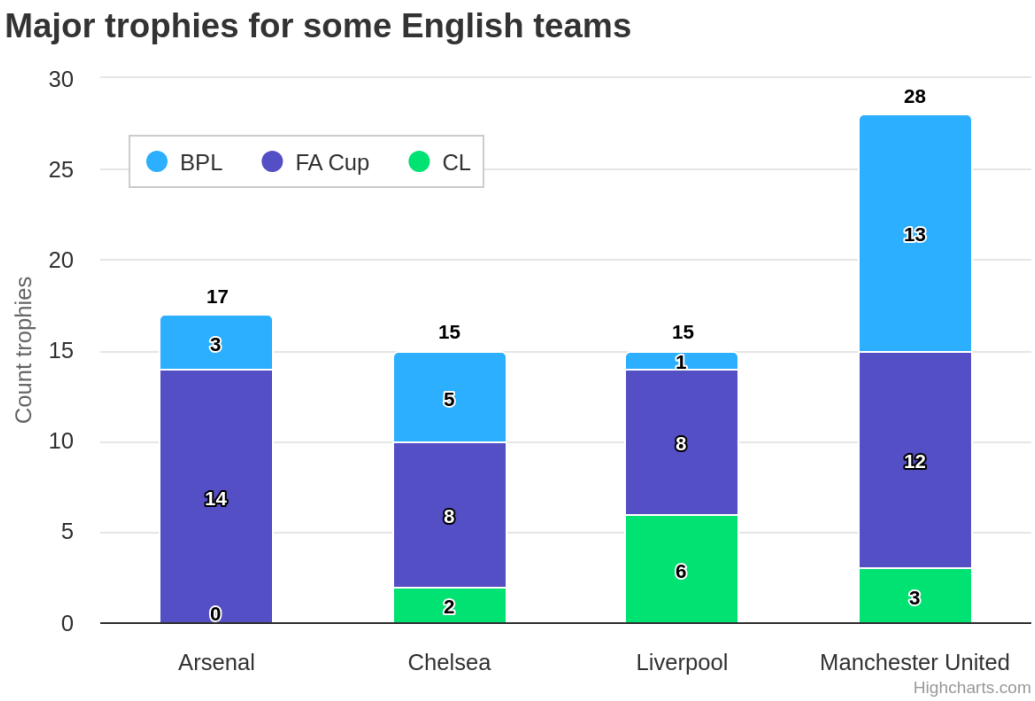

Figure 3.1: Stacked bar chart demo from Highcharts Charting Library illustrating the trophies won by four English football teams across three competitions. The x-axis groups the data by football team, with each bar representing the cumulative trophies won by each team. Each bar is divided into three segments, each corresponding to the trophies won in a different competition. These segments are color-coded by competition, as indicated by the legend.

effectively representing diagram elements that are part of multiple overlapping groupings.

To understand the Benthic traversal structure in detail, we walk through how a screen reader user might navigate the stacked bar chart in Figure 3.1. The steps are also depicted in Figure 3.3.

(1) At the top level, the Benthic traversal structure provides a description of the overall graphic. This top-level node contains two child nodes: the *X-axis* and the *Legend*, both of which group information within the graphic. A screen reader can choose to navigate to either the x-axis grouping or the legend grouping. In our walk-through, we choose to navigate to the x-axis grouping.

(2) The x-axis contains four bars, represented as four nodes grouped under the *X-axis element* category. Additionally, the screen reader always has access to its parent nodes, in this case, the *Root Graphic* node from which the screen reader user just navigated. The

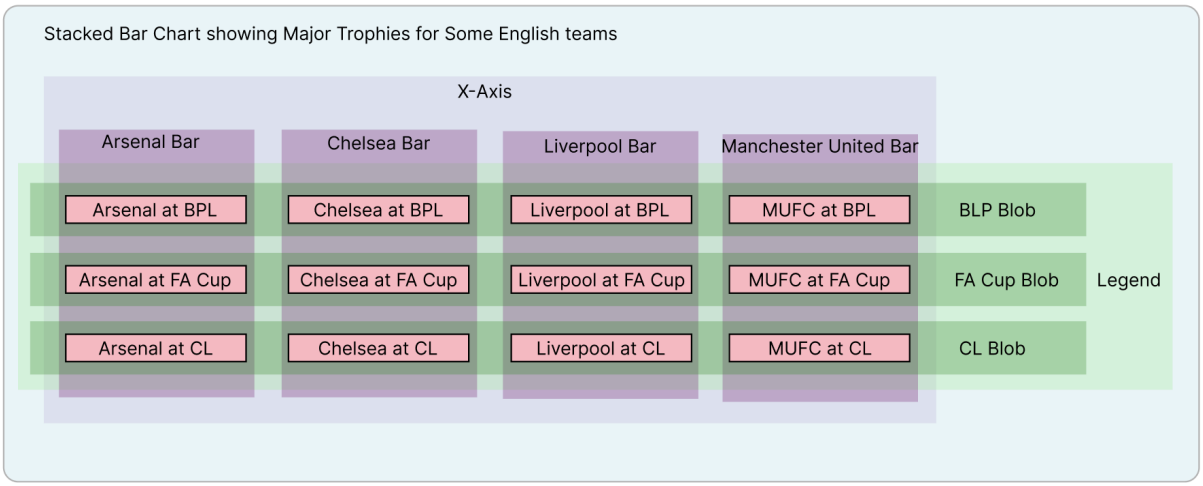

Figure 3.2: Benthic hypergraph traversal structure for the Highcharts stacked bar chart demo.

screen reader user can also choose to navigate to sibling nodes (other nodes that the parent node owns, such as the *Legend*). In our walk-through, we choose to navigate to the *Arsenal* bar.

(3) The *Arsenal* bar has three segments representing the number of trophies won by the Arsenal team in the three competitions. Here, we choose to navigate to the segment that represents the number of trophies Arsenal won in the CL competition.

(4) The *Arsenal at CL* segment has two parent nodes: the *Arsenal* bar and the *CL competition* group. Since the *Arsenal at CL* segment is part of multiple groupings, the screen reader user can explore the diagram from different angles. At this point, we choose to navigate to the node representing the CL competition legend grouping rather than continuing with x-axis navigation.

(5) Finally, the screen reader user reaches the node representing the trophies earned by different teams in the CL competition.

3.2 Key Characteristics of Benthic Traversal

This walk-through demonstrates key characteristics of the Benthic traversal structure:

First, it is evident that diagram elements fulfill dual roles, functioning both as visual marks (or objects) and as grouping elements. For instance, the x-axis in a diagram is not merely a line with labels; it also serves to categorize the trophy data for each football team. Nodes within the Benthic traversal structure mirror to this pattern by representing both diagram marks and relational groupings. This dual representation allows users to interact with individual diagram elements and understand the relations between them.

Second, the traversal pattern for all nodes in the graph is highly consistent. Each node in the hypergraph shares a uniform structure comprised of a description and a list of relevant groupings to the node. Different types of groupings within the graphic, such as the legend (using color associations) and the x-axis (using spatial containment), are traversed the same way. Consistency in Benthic's traversal pattern provides a predictable and learnable experience for screen reader users.

Third, Benthic traversal structures reflect how sighted users perceive diagrams by prioritizing visually salient information. The screen reader interface presents the most visually salient elements first, following the natural viewing order of the diagram. This method ensures that screen reader users receive information in an intuitive and effective sequence.

Finally, the Benthic traversal structure enables screen reader users to navigate through different levels of the diagram's visual hierarchy. For example, in the stacked bar chart, a screen reader user can navigate from a bar (the parent node) to any of the bar segments (the children nodes). The screen reader user can also easily return to the parent bar from any child bar segment. This bidirectional navigation ensures a clear and intuitive traversal experience, allowing users to reverse their traversal action by navigating to a previously visited level in the traversal hierarchy.

By incorporating these features, Benthic traversal structures provide an intuitive and informative method for blind and low vision users to explore complex diagrams.

Figure 3.3: Usage scenario walk-through. (1) At the top level, the Benthic traversal structure provides a description of the overall graphic. Navigation to the x-axis is chosen. (2) The x-axis contains four bars, represented as four nodes grouped under the "X-axis element" category. Navigation to the "Arsenal" bar is chosen. (3) The Arsenal bar has three segments representing the number of trophies won by the Arsenal team in the three competitions. Navigation to the bar segment that represents the number of trophies Arsenal won in the CL competition is chosen. (4) The "Arsenal at CL" bar segment has two parent nodes: the "Arsenal" bar and the "CL Competition" group. Navigation to the "CL Competition" group is chosen. (5) Final view for "CL Competition" group node.

Chapter 4

The Benthic System

Benthic is an intermediate representation and screen reader interface for graphic representations. Benthic's main contribution is the use of a *hypergraph* traversal structure, which utilizes *hyperedges* representing *relations* that connect multiple graph elements. Benthic's screen reader interface outputs relations in order of visual salience; more visually prominent relations are read to the screen reader user first. Benthic aims to create a screen reader experience that mirrors a sighted user's experience of visually interacting with a diagram.

4.1 Design Goals

A diagram is a data structure for computation, and the relationships between diagram elements help viewers reason and draw conclusions about the information conveyed in the diagram [1], [14]. The Benthic system formalizes a design space of traversal structures that can represent the underlying data structure for diagrams and reify the structure for screen reader usage. Benthic traversal structures follow three design principles: correspondence, consistency, and reversibility.

4.1.1 Correspondence

Diagrams are full of overlapping components, where one diagram element can appear to participate in multiple relations. For example, a diagram element can be simultaneously aligned with, spaced from, or contained by other diagram elements. Sighted users will often pay more attention to visually prominent relations; for example, a background bounding box may serve to visually associate diagram elements better than spacing or alignment. The viewing order of the graphic representation influences the way that a user interacts with and understands the content of the diagram. Therefore, it's important to create traversal structures that are able to provide the same interaction experience for screen reader users.

4.1.2 Consistency

Many existing methods for creating diagram traversal structures often involve custom specifications by diagram authors, and traversal behavior may be highly variable for different diagrams. As a result, it is very easy to create ad hoc and unintuitive traversal schemes.

Benthic offers a traversal scheme that is consistent across diagram types. The same relation in different diagrams are represented in the same way in the Benthic traversal structure. This consistency offers a learnable baseline traversal structure that screen reader users can navigate.

4.1.3 Reversibility

Reversibility (or the ability to undo a traversal action) is important in direct manipulation interfaces [15]. In some current systems, traversal structures for diagrams implement navigational paths that are not always bidirectional. Because of this, users are sometimes unable to undo their traversal action, which may be both confusing and frustrating. Benthic traversal structures are reversible

Benthic traversal structures are implemented as hypergraphs [10], where an edge is able

to join multiple elements in the graph. This structure is important because in relations with multiple elements, all related elements have a connecting bidirectional edge. Every action a user takes to traverse from one element to another in a relation is reversible because all elements in that relation are maximally connected.

4.2 Relations

Benthic defines a *relation* as a grouping of diagram elements. As a baseline, Benthic supports Gestalt relations [16], which are primitive groupings like alignment, spacing, and connection, common across diagrams from all disciplines [17], [18]. However, Benthic traversal structures can be easily extended to include domain specific relations for diagrams from different disciplines. Additionally, diagram elements can serve the purpose of grouping other diagram elements. Additionally, diagram elements can group other elements; for instance, a background bounding box relates elements by containing them within the same space. Therefore, in Benthic, diagram elements can also function as relations.

Benthic traversal structures are hypergraphs [10] with relations implemented as hyperedges. In Benthic, the hypergraph traversal structure is similar to a relaxed tree where the same child node can belong to multiple parent nodes. Hyperedges function like subtrees by connecting all child nodes of a given node. Overlaps in the diagram data structure are represented as overlapping hyperedges in the Benthic traversal structure. Additionally, diagram elements that do not relate other diagram elements are represented as hyperedges without children.

The hyperedge representation of relations is useful because relations typically group multiple diagram elements [13]. Intuitively, we expect a bidirectional edge between every pair of diagram elements within a relation. In other words, we expect to be able to traverse from one element in a relation to any other element in that relation, and we would expect to be able to backtrack in our traversal and directly return to an element we've already visited.

The hyperedge representation of relations makes Benthic traversals reversible.

Relations are a key concept in Benthic traversal structures because visual groupings are crucial in diagrams across various domains and can convey as much information as the diagram elements themselves [1]. For instance, physics diagrams might use spacing, alignment, or connections to model real-life scenarios. Sighted users implicitly process Gestalt and domain-specific relations, which aids in cognitive reasoning [19]. For example, solving a physics problem with a diagram that is inaccurately scaled, either due to incorrect spacing or alignment, can lead to incorrect reasoning about the situation, even if all necessary diagram objects are present. By providing relational information through Benthic, we aim to offer cognitive affordances to screen reader users, helping blind and low vision users reason about diagrams in the same way as sighted users.

4.3 Abstraction Boundaries

Relations are a useful construct in Benthic traversal structures because they enable the implementation of a consistent hierarchical structure. Screen reader users typically seek an overview of the graphic representation first, followed by the ability to zoom in on specific details [20]. Additionally, diagrams often contain overlapping relations that can significantly increase the cognitive load for screen reader users during traversal. Therefore, abstraction boundaries are necessary to help traversal structures orient screen reader users and prevent cognitive overloading.

Each relation node in the Benthic traversal structure is augmented with a detailed description of the relation as well as descriptions of the children nodes grouped by the relation. These descriptions help orient screen reader users to possible traversal paths and provide an overview of the sub-graph structure. Screen reader users can choose whether or not to enter a relation and explore the diagram nodes contained by the relation. This allows for variable fidelity traversals where users can request details on demand. Abstraction boundaries

are beneficial because a traversal structure with extensive details may not always be useful, depending on the user's task [5], [21].

Abstraction boundaries are especially helpful for representing domain-specific relations, which are typically composed of numerous primitive Gestalt relations. Screen reader users generally prefer to interpret diagrams through semantically meaningful domain-specific relations rather than through a collection of primitive relations [5]. Domain-specific relations reduce computational effort because they correspond to semantically meaningful concepts within the diagram [1]. Since primitive relation nodes are children nodes of domain-specific relation nodes, Benthic traversal structures allow screen reader users to bypass traversal of primitive relations (e.g., by not traversing the children of the relation node). This approach reduces cognitive load and enhances the understanding of the diagram's structure.

4.4 Priority

In Benthic, *priority* refers to the ranking of a relation's visual salience. Benthic traversal structures enable screen reader users to navigate through higher-priority relations first. This mirrors the natural viewing order of diagrams, where a sighted user's attention is likely to be drawn to the most visually prominent elements in the diagram [14], [22], [23].

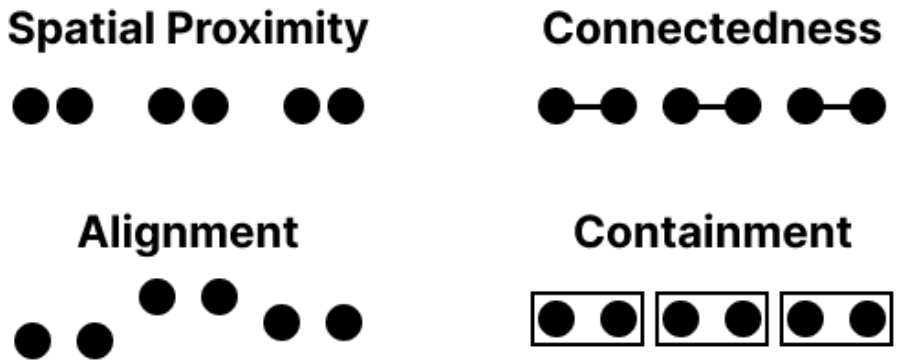

Figure 4.1: Visual examples of different primitive Gestalt relation categories, including spatial proximity, alignment, connectedness, and containment.

Benthic provides a default priority ordering for primitive Gestalt relation categories, which include spatial proximity, alignment, connectedness, and containment [16]–[18]. Examples of each relation category are illustrated in Figure 4.1. In Benthic's default priority ordering, containment relations have the highest priority, followed by connectedness. Alignment and spatial proximity have the lowest priority. This priority ordering can be easily customized; for example, users can adjust the priority of alignment relations to be higher than that of spatial proximity relations. Additionally, users can assign priority levels to any domain-specific relations incorporated into the Benthic traversal structure.

Benthic's priority approach offers two key advantages. First, priority helps serialize diagram information in a manner that aligns with the natural viewing patterns of sighted users. Relation ordering is also highly flexible to accommodate high variance in how diagrams are read across domains. Second, priority standardizes the interaction pattern for hierarchical and overlapping relations, ensuring a consistent and clear representation of complex diagram structures.

4.4.1 Serializing Relations

Diagrams contain numerous relations that sighted users can perceive and process almost simultaneously [22]. However, screen readers present information sequentially [24], making the order of presentation crucial to avoid overwhelming users with non-essential details. Benthic leverages the *priority* of relations as a heuristic to determine the most visually salient information in the diagram, which is likely the most important for screen reader users [14]. The highest priority relations are presented first in the traversal structure, ensuring that key information is conveyed effectively.

Additionally, different domains may prioritize various diagram relations, resulting in variations in their visual prominence. For instance, connectedness is crucial in graph theory, whereas spatial alignment may be more important in diagrams modeling physical situations. In these cases, Benthic's priority approach provides a simple method to adjust the serializa-

tion order of relations, making its traversal structures easily adaptable to a wide range of fields.

4.4.2 Standardized Interactions

In many existing diagram accessibility systems, hierarchical and overlapping relations are treated as distinct concepts. Visual hierarchy receives special attention because it guides focus within the diagram by categorizing elements and indicating their relative importance [25]. Benthic's priority model explains why visual hierarchy is often emphasized: it typically manifests as containment relations, which are the most visually salient form of perceptual grouping. Benthic's traversal structures consistently represent both hierarchical and overlapping relations, assigning higher priority levels to hierarchical ones. This approach ensures a unified traversal pattern for both types of relations in Benthic traversal structures.

4.5 Traversal Structure Specification

The data type specification for the Benthic traversal structure is detailed in Figure 4.2. The overall hypergraph structure maps `Id`s to corresponding `RelationNode` objects, with each `RelationNode` representing a node in the traversal structure. Each `RelationNode` contains six key pieces of information: the node's id, the name read by the screen reader, the description read by the screen reader, a list of ids for the node's parent nodes, a list of ids for any child nodes, and the priority of the relation represented by the node. Relations in the `parents` and `children` list are sorted by priority when displayed to the screen reader interface.

```typescript
type Id = string;

export type RelationNode = {
  id: Id;
  displayName: string;
  description?: string;
  parents: Id[];
  children: Id[];
  priority: number;
};

export type Hypergraph = {
  [id: Id]: RelationNode;
};
```

Figure 4.2: The TypeScript type specification for Benthic traversal structure hypergraph and nodes. Each `RelationNode` contains 6 pieces of information, including the node's id, name, description, parents, children, and priority.

4.6 Screen Reader Interaction

Benthic features a screen reader interface that allows users to navigate through the diagram traversal structure. Figure 4.3 provides an example of the screen reader output for the X-axis node within the stacked bar chart traversal structure discussed in the usage scenario. Screen reader users can access an interface displaying the node name, node description, and any relations relevant to the current node.

Figure 4.3: Screen reader output for X-axis node in the traversal structure for the Highcharts stacked bar chart example.

When navigating the traversal structure, the screen reader user first encounters the name

and description of the current node. They can choose to enter the node by using the screen reader modifier key and the down arrow key. Once inside the node's abstraction boundary, they can access a list of relations that the current node participates in. This list is presented in order of visual salience (or priority) of the relation. Users can then select a relation to traverse and navigate to the next node of their choice. The screen reader interface then displays the new information for the node the user navigates to.

Chapter 5

Evaluation

We evaluate Benthic through three comparative case studies. The first case study contrasts the Benthic and Data Navigator traversal structures for a stacked bar chart from Highcharts [11]. This example is discussed in the usage scenario and is illustrated in Figure 3.1. The second case study compares the Benthic and Data Navigator traversal structures for an intersecting sets diagram created using Penrose [12]. The third case study examines Larkin and Simon's proposed data structure for a physics pulley diagram [1], exploring how Benthic traversal structures can facilitate the reasoning task presented by Larkin and Simon.

5.1 Case Study 1: Stacked Bar Chart

The Benthic traversal structure for the Highcharts stacked bar chart [11] is illustrated in Figure 3.2. The structure features a top-level node representing the root graphic. This top-level node contains two child nodes: the x-axis and the legend. The x-axis node contains child nodes for each bar in the chart (representing the number of trophies won by each football team), while the legend node includes child nodes for each color grouping (representing the number of trophies won per competition). Within each of the bar and color groupings, bar segments represent the intersection of each football team and competition.

Data Navigator [8], an existing system for accessible visualizations that uses a graph

based approach, analyzed the Highcharts stacked bar chart example as one of their case studies. In this section, we will compare and contrast the traversal structures of Benthic and Data Navigator for the Highcharts stacked bar chart.

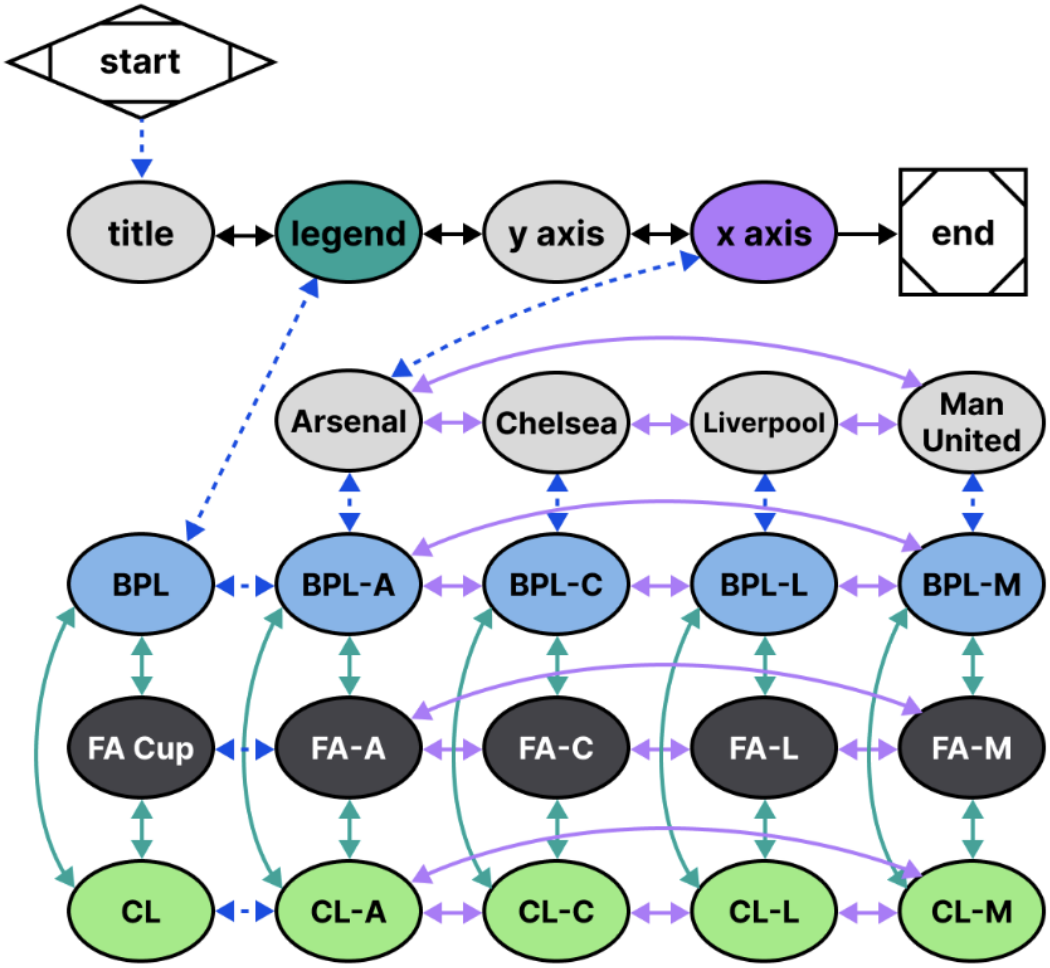

Figure 5.1: Data Navigation traversal structure for Highcharts stacked bar chart example. This traversal structure enables two-axis traversal. Specifically, users can use the left and right arrow keys to move across the x-axis items (trophies won, grouped by football team). The up and down arrow keys navigate through legend items (trophies won, grouped by competition). Enter and backspace keys allow users to enter and exit scopes or groupings.

Data Navigator proposed a traversal structure that enables users to navigate the stacked bar chart via two axes: horizontally (left and right) or vertically (up and down). Specifically, users can use the left and right arrow keys to move across the x-axis items (trophies won, grouped by football team). The up and down arrow keys navigate through legend items

(trophies won, grouped by competition). Additionally, the enter and backspace keys allow users to enter and exit scopes or groupings. For instance, pressing enter on a bar for a specific football team allows users to see the breakdown of trophies earned by that team across the three competitions. Data Navigator highlights that a key benefit of this navigation scheme is its alignment with the visual affordances of the chart because the two axes traversal aligns with the axes of the stacked bar chart. The proposed traversal structure by Data Navigator is illustrated in Figure 5.1.

However, there are some limitations to this approach. First, the two-axis navigation suggested works well only for diagrams arranged in a grid-like fashion, where left-right and up-down movements correspond intuitively to visual elements. In cases where the diagram structure is less consistent, it may be challenging to determine the appropriate actions for the left-right and up-down arrow keys. On the other hand, Benthic's hypergraph traversal structure creates a more flexible and intuitive traversal structure. By organizing nodes into hyperedges representing diagram relations, Benthic enables users to navigate diagrams based on the relational importance of elements rather than their spatial arrangement. This method ensures that navigation remains intuitive even in non-grid-like structures, providing a consistent experience for screen reader users.

Second, in Data Navigator's proposed traversal structure, only the first child is connected to the parent of the grouping. Pressing backspace while on any child node that is not the first child in a grouping will have no effect. In other words, some children cannot navigate to the parent node, resulting in a non-reversible traversal. This makes it difficult for screen reader users to move up the hierarchy. In comparison, Benthic's hypergraph traversal structure ensures that all nodes maintain a connection to their parent nodes. By preserving these hierarchical connections, Benthic ensures that screen reader users can move up and down the hierarchy seamlessly, making the traversal process fully reversible and more intuitive.

Finally, the stacked bar chart example includes diagram elements that participate in multiple relations. Each segment of the stacked bar chart is part of both a team grouping

(on the x-axis) and a competition grouping (specified by the legend). In the Data Navigator traversal, there are two key mappings used to navigate from the bar segments to either the parent bar or the legend color grouping: L and backspace. Backspace is the primary key used to navigate up the traversal hierarchy, but the L key is custom-assigned for this specific scenario. This inconsistency in representing hierarchy relations makes the Data Navigator system less learnable. In contrast, Benthic overcomes this inconsistency by standardizing the navigation for all types of relations within diagrams. This approach enhances learnability by reducing the cognitive load required to remember different navigation patterns for various relations.

5.2 Case Study 2: Intersecting Sets

In another case study, the authors of Data Navigator analyzed the traversal of an intersecting sets diagram created using Penrose [12]. This diagram, along with its corresponding Data Navigator traversal scheme, is illustrated in Figure 5.2. The diagram consists of two overlapping circles representing two intersecting sets. Each set has a common shared region with the other, as well as a region that is exclusively its own (similar to a Venn Diagram). Data Navigator's traversal scheme operates on three semantic levels: the overview level, the inclusion level, and the exclusion level. As shown in Figure 5.2, these levels are organized from the top to the bottom of the figure: the node labeled (1) represents the overview level, the nodes labeled (2), (3), and (4) represent the inclusion level, and the nodes labeled (5) and (6) represent the exclusion level.

In the Data Navigator traversal, pressing the enter key at node (1) brings the user to node (2), as it is the first child of node (1). Using the left and right arrow keys, the user can navigate to nodes (3) and (4) from node (2). The exclusion level, represented by nodes (5) and (6), can be accessed by pressing enter from nodes (2) and (4) respectively. The left and right arrow keys allow navigation between nodes (5) and (6). Finally, pressing backspace at

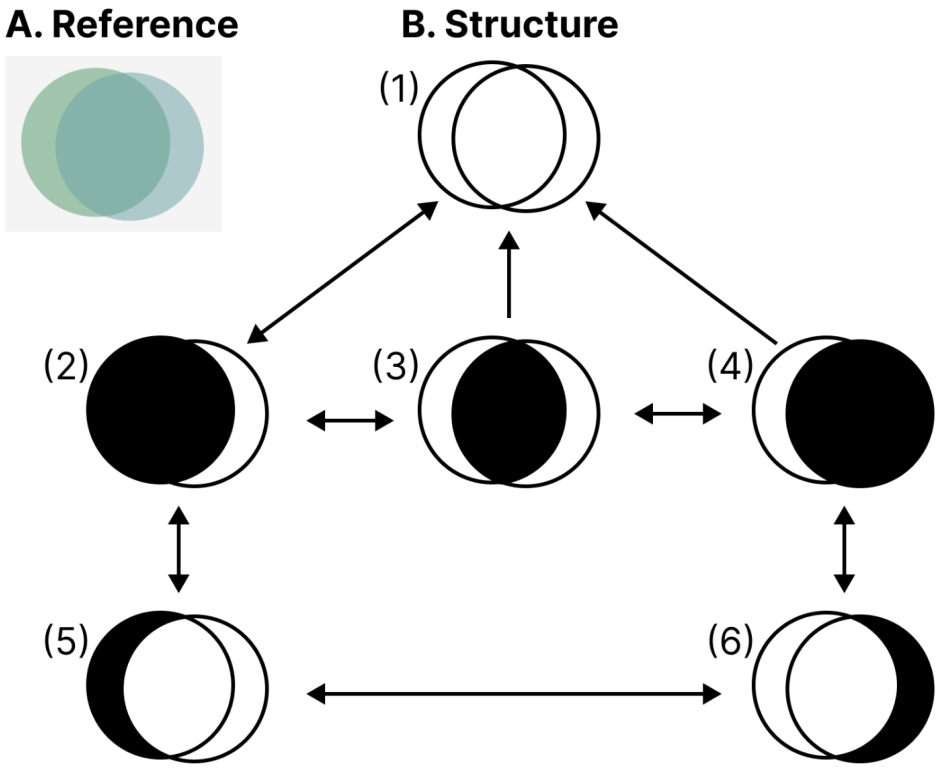

Figure 5.2: Original graphic and Data Navigator traversal structure for the Penrose intersecting sets diagram. The diagram consists of two overlapping circles representing two intersecting sets. Each set has a common shared region with the other, as well as a region that is exclusively its own (similar to a Venn Diagram). Data Navigator's traversal scheme operates on three semantic levels: the overview, the inclusion, and exclusion levels.

any node moves the user up a level to its parent. For example, pressing backspace at node (5) moves the user to node (2).

The Data Navigator traversal for the intersecting sets diagram has both strengths and weaknesses. One challenge arises from the lack of an inherent grid-like organization in the diagram, making a two-axis traversal method difficult to implement. As a result, the authors designed a custom hierarchy and traversal scheme for this diagram. For other intersecting sets diagrams, traversal schemes would have to be manually designed, which can result in inconsistent or ad-hoc traversal structures. The manually specified hierarchical structure and key mappings for navigation actions can also vary significantly from diagram to diagram, even if the diagrams represent similar subjects. However, custom-designed traversal structures have their advantages. When diagram authors specify the hierarchy, they can create more

semantically meaningful layers. For instance, the Data Navigator's overlapping sets diagram includes three meaningful layers: overview, inclusion, and exclusion layers.

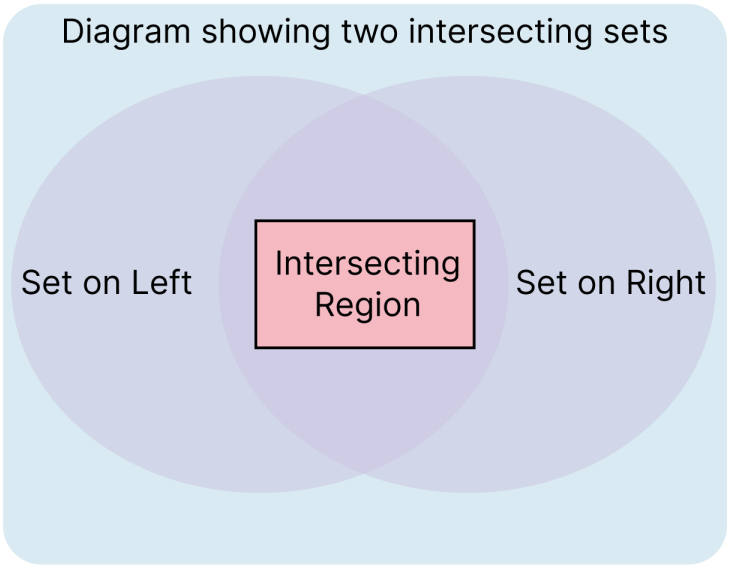

Figure 5.3: Benthic hypergraph for Penrose intersecting sets diagram. Traversal structure features a top-level node representing an overview of the graphic, which contains two nodes corresponding to the left and right overlapping sets in the original Penrose diagram. These sets share a single node that represents their intersection.

The Benthic hypergraph for the intersecting sets example, shown in Figure 5.3, provides a straightforward representation that visually resembles the original diagram. It features a top-level node representing an overview of the graphic, which contains two nodes corresponding to the left and right overlapping sets in the original Penrose diagram. These sets share a single node that represents their intersection. Despite the differences in diagram types, navigating the Benthic traversal structure for this intersecting sets diagram follows a similar pattern to navigating the traversal structure for the Highcharts stacked bar chart example from case study 1. This consistency in navigation rules across different types of diagrams makes the traversal structure easy to learn. Additionally, the traversal hypergraph is organized in a way that closely mirrors the structure of the actual intersecting sets diagram. Therefore, the intersecting sets diagram demonstrates that the Benthic traversal structure maintains both consistent navigation patterns and correspondence to the original diagram.

The Benthic hypergraph for the intersecting sets example is shown in Figure 5.3. The hypergraph representation is rather simple (and also looks very similar visually to the original diagram); there is one top level node representing an overview of the graphic. This top level node contains two sets (representing the left and right overlapping sets in the Penrose diagram). These two sets both contain a node representing the set intersections. Navigating the Benthic traversal structure for the intersecting sets diagram would be similar to navigating the traversal structure for the Highcharts stacked bar chart example from Case Study 1. Additionally, the traversal hypergraph itself is organized in a way that is highly similar to the actual intersecting set diagram. Therefore, the Benthic traversal structure is consistent and correspondent (traversal structure mirrors structure of original diagram).

However, the current Benthic traversal structure falls short in encoding the same level of semantic information as the manually designed Data Navigator traversal structure. The Benthic structure lacks the "inclusion" and "exclusion" layers, which could be particularly relevant in a topological or mathematical context. Future work should explore how to use the existing relational structure in Benthic to infer domain-specific information like the "inclusion" and "exclusion" hierarchy layers of set diagrams.

5.3 Case Study 3: Pulley Diagram

In our third case study, we assess the effectiveness of Benthic's traversal structure in implementing the traversal for a physics pulley diagram, as proposed by Simon and Larkin in their 1987 work "Why a Diagram is (Sometimes) Worth Ten Thousand Words" [1].

Figure 5.4 this pulley diagram, which involves three pulleys labeled `A`, `B`, and `C`, and two weights labeled `W1` and `W2`. Pulley `B` is fixed to a support at the top, while pulleys `A` and `C` are free-hanging. A string runs over Pulley `B` and is labeled rope `t`, then extends downward to Pulley `A`, labeled rope `x`. From Pulley `A`, the string descends to connect with weight `W1`, labeled rope `p`. Another segment of the string from Pulley `A` goes upward, labeled rope `y`,

and passes over Pulley `C`, labeled rope `z`. Finally, the string descends from Pulley `C` to attach to weight `W2`, labeled `s`.

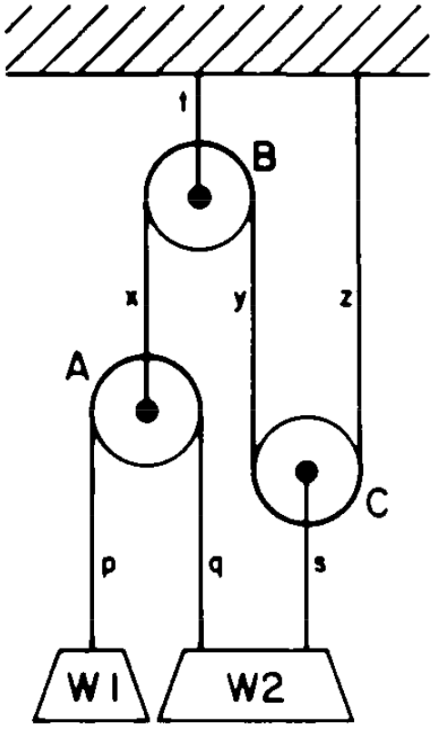

Figure 5.4: Pulley system from Larkin and Simon's 1987 work "Why a Diagram is (Sometimes) Worth Ten Thousand Words" involving three pulleys labeled `A`, `B`, and `C`, and two weights labeled `W1` and `W2`.

Larkin and Simon described a process by which a sighted user could determine the weight of `W2` given the weight of `W1` using this pulley diagram. They introduced two domain-specific relations, `hangs from` and `pulley system` to associate the ropes and pulleys in the diagram. For the `hangs from` relation, when an object with weight hangs from a rope, the rope's tension equals the weight of the object. For the `pulley system` relation, the weight supported by the pulley is determined by the combined tension of the ropes within the system. By applying the `hangs from` and `pulley system` relations, Larkin and Simon demonstrated how these groupings in the diagram could help someone traverse the diagram to reason and problem solve. Larkin and Simon showed that a diagram functions as a data structure that facilitates computation.

The Benthic traversal structure for the pulley diagram, as illustrated in Figure 5.5, offers similar cognitive affordances to screen reader users as the original pulley diagram does to sighted users. In the Benthic traversal hypergraph, overlapping hyperedges representing the `hangs from` and `pulley system` relations interconnect the graph nodes. Using the Benthic traversal structure, a screen reader user can navigate from the weight node `W1` to the rope node `P` via the `hangs from` relation. Subsequently, through the `pulley system` relation, the user can traverse to rope `Q`, and so on. These domain-specific relations, intrinsic to physics pulley diagrams, carry semantic meaning and align with the concepts depicted in the original diagram. Users can navigate into the `hangs from` and `pulley system` relations to explore the primitive Gestalt relations that constitute them. However, these domain-specific relations serve as abstraction boundaries, ensuring that screen reader users do not need to concern themselves with the underlying implementation details of the diagram. The pulley example demonstrates that the Benthic traversal structure is able to represent the underlying data structure of the pulley diagram.

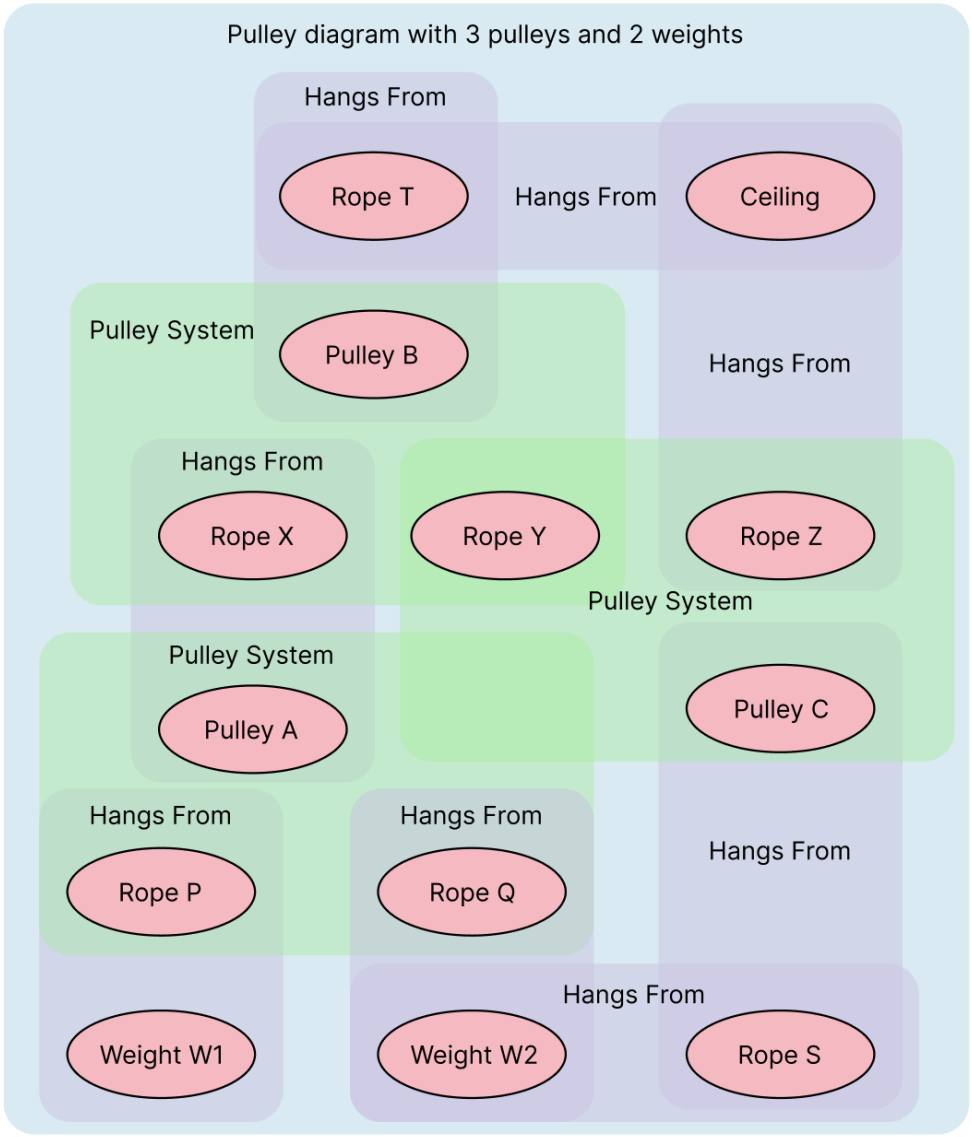

Figure 5.5: Benthic hypergraph traversal structure for the pulley system from Larkin and Simon's 1987 work "Why a Diagram is (Sometimes) Worth Ten Thousand Words" with domain specific relations `pulley system` and `hangs from`.

Chapter 6

Discussion and Future Work

In this paper, we presented Benthic, an intermediate representation and screen reader interface for graphic representations. We have demonstrated how Benthic's hypergraph traversal structure allows for corresponding, consistent, and reversible screen reader traversal. The Benthic system offers several promising directions for future research and tool development.

6.1 Graph Rewrites

We focused on designing an expressive traversal structure for graphic representations that enables screen reader users to navigate all the relations present in a diagram. However, depending on the task, users might not need access to all the provided information. For instance, blind and low vision users who encounter diagrams in a problem-solving context, such as students solving math or physics problems, will likely need more detailed information about the diagram structure. On the other hand, a user casually browsing a diagram on the internet might only require a top-level traversal.

Because Benthic traversal structures contain information about relations, we can simplify these structures effectively. For example, an annotation typically appears as a text node connected to another diagram node through an arrow relation. Knowing that arrow relations associate different diagram nodes allows us to infer and simplify the structure. Instead of

treating the annotation as a separate node, we can integrate it into the description of the node it annotates, providing a more streamlined and less cognitively demanding experience for the user.

Future projects could explore various methods of using relational information to infer and simplify the graph structure. This could include developing algorithms to automatically identify and merge related elements or creating user-defined rules for simplifying specific types of relations in diagrams. Additionally, customization options could be implemented to allow users to adjust the level of detail based on their preferences and the context of their tasks, ensuring a more tailored and efficient navigation experience.

6.2 Adapters for Diagramming Frameworks

Benthic traversal structures serve as intermediate representations that can be outputted to screen readers. However, extracting relation information from diagrams to create Benthic traversal structures can be challenging. Future work could focus on developing adapters to compile diagram outputs from various visualization libraries into Benthic traversal structures, making graphic representations automatically screen reader-friendly.

Creating these adapters involves extracting relation information from the outputs of visualization tools. Certain tools, like the Bluefish diagramming framework, make this process straightforward due to their relational scenegraph. However, integrating with popular visualization libraries such as D3, Vega-lite, and others presents more complex challenges. These tools often place diagram elements at explicit positions, so relational information is not retained.

In cases where relational information is not automatically generated, future researchers will need to develop methods to extract this information using calculations and heuristics. For example, they might analyze proximity and alignment to infer relationships between diagram elements.

Developing robust adapters for a variety of diagramming frameworks will be crucial for broadening the adoption and utility of Benthic. By integrating with the tools that diagram authors are already using, we can ensure that graphic representations become more accessible and navigable for blind and low vision users across diverse contexts and applications.

References

[1] J. H. Larkin and H. A. Simon, "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science*, vol. 11, no. 1, pp. 65–100, 1987. DOI: https://doi.org/10.1111/j.1551-6708.1987.tb00863.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1551-6708.1987.tb00863.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1551-6708.1987.tb00863.x.

[2] N. W. Kim, S. C. Joyner, A. Riegelhuth, and Y. Kim, "Accessible visualization: Design space, opportunities, and challenges," *Computer Graphics Forum*, vol. 40, no. 3, pp. 173–188, 2021. DOI: https://doi.org/10.1111/cgf.14298. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14298. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14298.

[3] A. Sharif, S. S. Chintalapati, J. O. Wobbrock, and K. Reinecke, "Understanding screen-reader users' experiences with online data visualizations," in *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '21, <conf-loc>, <city>Virtual Event</city>, <country>USA</country>, </conf-loc>: Association for Computing Machinery, 2021, ISBN: 9781450383066. DOI: 10.1145/3441852.3471202. URL: https://doi.org/10.1145/3441852.3471202.

[4] M. R. Morris, J. Johnson, C. L. Bennett, and E. Cutrell, "Rich representations of visual content for screen reader users," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, <conf-loc>, <city>Montreal QC</city>, <country>Canada</country>, </conf-loc>: Association for Computing

Machinery, 2018, pp. 1–11, ISBN: 9781450356206. DOI: 10.1145/3173574.3173633. URL: https://doi.org/10.1145/3173574.3173633.

[5] J. Zong, C. Lee, A. Lundgard, J. Jang, D. Hajas, and A. Satyanarayan, "Rich Screen Reader Experiences for Accessible Data Visualization," *Computer Graphics Forum (Proc. EuroVis)*, 2022. DOI: 10.1111/cgf.14519. URL: http://vis.csail.mit.edu/pubs/rich-screen-reader-vis-experiences.

[6] S. Jones, I. Pedraza Pineros, D. Hajas, J. Zong, and A. Satyanarayan, ""customization is key": Reconfigurable textual tokens for accessible data visualizations," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '24, ACM, May 2024. DOI: 10.1145/3613904.3641970. URL: http://dx.doi.org/10.1145/3613904.3641970.

[7] M. Blanco, J. Zong, and A. Satyanarayan, "Olli: An Extensible Visualization Library for Screen Reader Accessibility," in *IEEE VIS Posters*, 2022. URL: http://vis.csail.mit.edu/pubs/olli.

[8] F. Elavsky, L. Nadolskis, and D. Moritz, "Data Navigator: An accessibility-centered data navigation toolkit," *IEEE Transactions on Visualization and Computer Graphics*, 2023. URL: http://dig.cmu.edu/data-navigator/.

[9] V. Sorge, M. Lee, and S. Wilkinson, "End-to-end solution for accessible chemical diagrams," in *Proceedings of the 12th International Web for All Conference*, ser. W4A '15, Florence, Italy: Association for Computing Machinery, 2015, ISBN: 9781450333429. DOI: 10.1145/2745555.2746667. URL: https://doi.org/10.1145/2745555.2746667.

[10] Z. Wenping, L. Meilin, and L. Jiye, "Hypergraphs: Concepts, applications and analysis," in *2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2022, pp. 1–6. DOI: 10.1109/PAAP56126.2022.10010428.

[11] Highsoft, *Stacked column demo*, https://www.highcharts.com/demo/column-stacked [Accessed: May 2024], 2018.

[12] K. Ye, W. Ni, M. Krieger, D. Ma'ayan, J. Wise, J. Aldrich, J. Sunshine, and K. Crane, "Penrose: From mathematical notation to beautiful diagrams," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 144–1, 2020.

[13] J. Pollock, C. Mei, G. Huang, D. Jackson, and A. Satyanarayan, *Bluefish: A relational framework for graphic representations*, 2023. arXiv: 2307.00146 [cs.GR].

[14] M. S. Fine and B. S. Minnery, "Visual salience affects performance in a working memory task," *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, vol. 29, no. 25, pp. 8016–8021, 2009. DOI: 10.1523/JNEUROSCI.5503-08.2009. URL: https://doi.org/10.1523/JNEUROSCI.5503-08.2009.

[15] S. Margono and B. Shneiderman, "A study of file manipulation by novices using commands vs. direct manipulation," in *Sparks of Innovation in HumanComputer Interaction*. 1987, pp. 39–50. URL: http://citeseer.ist.psu.edu/margono87study.html.

[16] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt, "A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization," *Psychol Bull*, vol. 138, no. 6, pp. 1172–1217, Nov. 2012.

[17] C. Richards, "Diagrammatics," Available from: diagrammatics.com, Ph.D. dissertation, Royal College of Art, 1984.

[18] J. Bertin, *Semiology of graphics*. University of Wisconsin press, 1983.

[19] S. L. Franconeri, L. M. Padilla, P. Shah, J. M. Zacks, and J. Hullman, "The science of visual data communication: What works," *Psychological Science in the Public Interest*, vol. 22, no. 3, pp. 110–161, 2021. DOI: 10.1177/15291006211051956. URL: https://doi.org/10.1177/15291006211051956.

[20] A. Sharif, S. S. Chintalapati, J. O. Wobbrock, and K. Reinecke, "Understanding screen-reader users' experiences with online data visualizations," in *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, ser. AS-

SETS '21, <conf-loc>, <city>Virtual Event</city>, <country>USA</country>, </conf-loc>: Association for Computing Machinery, 2021, ISBN: 9781450383066. DOI: 10.1145/3441852.3471202. URL: https://doi.org/10.1145/3441852.3471202.

[21] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Visual Languages, IEEE Symposium on*, Los Alamitos, CA, USA: IEEE Computer Society, Sep. 1996, p. 336. DOI: 10.1109/VL.1996.545307. URL: https://doi.ieeecomputersociety.org/10.1109/VL.1996.545307.

[22] R. W. Jones, J. W. Warner, and C. L. Cross, "How are graphs read? an indication of sequence," en, *Behav. Res. Methods Instrum. Comput.*, vol. 30, no. 2, pp. 238–245, Jun. 1998.

[23] K. B. Bennett and J. M. Flach, "Graphical displays: Implications for divided attention, focused attention, and problem solving," *Human Factors*, vol. 34, no. 5, pp. 513–533, 1992. DOI: 10.1177/001872089203400502. URL: https://doi.org/10.1177/001872089203400502.

[24] M. Hegde, "User Experience Study of Screen Readers for Visually Challenged Users," Aug. 2023. DOI: 10.25394/PGS.23750877.v1. URL: https://hammer.purdue.edu/articles/thesis/User_Experience_Study_of_Screen_Readers_for_Visually_Challenged_Users/23750877.

[25] Interaction Design Foundation - IxDF, *What is visual hierarchy?* Accessed: 2024-05-17, Aug. 2016. URL: https://www.interaction-design.org/literature/topics/visual-hierarchy.