Petra Neumann

# Focus+Context Visualization of Relations in Hierarchical Data

Otto-von-Guericke-University Magdeburg

Faculty of Computer Science
Department of Simulation and Graphics

Diplom Thesis

# Focus+Context Visualization of Relations in Hierarchical Data

Author:

Petra Neumann

September 6, 2004

Supervisor:

Dr. Stefan Schlechtweg

Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D–39016 Magdeburg
Germany

**Neumann, Petra:**
Student Number: 157849
*Focus+Context Visualization of Relations in Hierarchical Data*
Diplom Thesis, Otto-von-Guericke University Magdeburg, 2004.

1st Reviewer: Dr. Stefan Schlechtweg
2nd Reviewer: Dr. Sheelagh Carpendale

# Abstract / Zusammenfassung

Over the recent years, visualization has become increasingly important in electronic and interactive media. This thesis introduces a new interactive visualization technique tailored to emphasize relations between data elements in a hierarchical data structure. The display of relations between data items in tree structures is commonly solved by grouping items according to color, shape, proximity, size, etc. The approach presented in this thesis, however, represents relations using direct connections. These have been shown to be a stronger grouping principle than proximity, color, size, or shape. The presented visualization technique involves filter and zoom mechanisms to explore the data structures. Automation mechanisms for these operations are introduced through focus+context navigation in the tree. A particular quality of the presented techniques is the consideration of relations in focus+context interaction and presentation which is a novel aspect for tree visualizations.

Die visuelle Darstellung von Datensätzen und darin enthaltenen Informationen in interaktiven digitalen Medien erhält seit einigen Jahren eine immer größere Bedeutung. Diese Diplomarbeit stellt eine neue interaktive Visualisierungstechnik vor, die in besonderer Weise die Darstellung von Relationen zwischen Datenelementen in hierarchischen Datensätzen ermöglicht. Typischerweise werden Relationen zwischen Datenelementen in solchen Datensätzen durch Farbcodierung, Verwendung von ähnlichen Formen oder Texturen realisiert. Die Visualisierung, die hier vorgestellt wird, benutzt im Gegensatz dazu eine direkte visuelle Verbindung als Codierungstechnik. Dies basiert auf Untersuchungen, die ergeben haben, dass solche direkten visuellen Verbindungen Relationen besser darstellen können als Farbe oder Form. Die vorgestellte Visualisierungstechnik beinhaltet außerdem Zoom- und Filtermechanismen um die Daten interaktiv untersuchen zu können. Als ein Automatisierungsmechanismus für diese Interaktionsmöglichkeiten wird Focus+Context Navigation in den Daten vorgestellt. Besonders hervorzuheben ist in diesem Zusammenhang die Beachtung von Relationen in der Focus+Context Navigation und Präsentation, die neuartig für die Visualisierung von Baumstrukturen ist.

# Acknowledgement

My studies and the work on this thesis would not have been possible and as enjoyable without the support and help of a great many people.

First of all I would like to thank my parents and sister for their generous support of my school and university studies in and out of the country. Thanks for all the long trips between Magdeburg and Steinenberg, for helping me move in and out, and for just being a wonderful family.

In Magdeburg I would like to thank Stefan Schlechtweg for being an excellent and enthusiastic supervisor. His helpful suggestions gave the work developed here much of its present character.

Thanks go also to the professors who made the "Diplomandenlabor" happen that provided me with a great working environment for the completion of this thesis. The interaction and friendship with other students in the lab has contributed greatly to the work presented here. In particular, I am indebted to Angela "Strokewoman" Brennecke who insisted many times that taking a break is an essential part of the work day and who always shared her great knowledge of LaTeX with me.

Many thanks also go to my friends Anja, Ingmar, Julia, Laura, Sandra, Uta, René, and all others who made my stay in Magdeburg so much more valuable and enjoyable.

My special thanks go to Tobias "Floh" Isenberg who encouraged and supported me since my third semester at university. Thanks for enduring bad-mood times when I had to study for exams, for proof-reading many of my works, in particular this thesis, for introducing me to LaTeX, and thanks for being a wonderful friend.

I would also like to thank the German Academic Exchange Service (DAAD) for funding my research stays in Canada and Taiwan.

# Contents

CHAPTER 1

# Introduction

Over the recent years, the combination of visualizations and digital media has become increasingly important. Much of today's communication involves visualizations in digital form. Aiding users with different tasks by making the underlying data visual is one important research area in computer science. Such research in the area of visualization is largely based on a characteristic of the human visual system: visual information can be processed in parallel and with a high bandwidth into the human cognitive centers (WARE, 2000). This is also the reason why an effective information display can often lead to insight quicker and more memorably than a few pages of written text. This thesis is concerned with the creation of such an effective information display. As one of the most common data types, hierarchical data structures will be visualized with an emphasis on the display of relations between data items. Motivation for the research presented here will be discussed in this chapter. The discussion will give insight into the inherent problems in the development of the visualization and briefly summarize the further development process discussed in subsequent chapters.

## 1.1 Motivation

Knowledge obtained from visualizations often comes from insight gained by recognizing that data items are related in some way. In graphs, for example, edges can represent any conceivable kind of relation including those of temporal, causal, or functional nature. The display of relations is, therefore, one of the most essential tasks in information visualization. Understanding relations between items in a visualization helps the viewer to build a mental model of the underlying data. This mental model is needed to understand the scheme or situation to which the presented data refers. For making decisions based on the visualization of data the interpretation of this internal model is essential. This thesis will introduce a new visualization technique specifically tailored to recognize relations between data elements in a hierarchical data structure.

## 1.2 Problem Statement

In traditional displays of tree structures, edges represent parent-child relations. If more than the inherent parent-child relations needs to be visualized for nodes in a tree, different encoding techniques are necessary. Figure 1.1 affirms this proposition. The figure displays a tree structure in a traditional node-link diagram. Additional links are introduced to represent binary relations between nodes in the tree. From the display parent-child relations and the additional relations cannot be distinguished. In addition, edge-edge intersections are introduced that lead to a cluttered display.

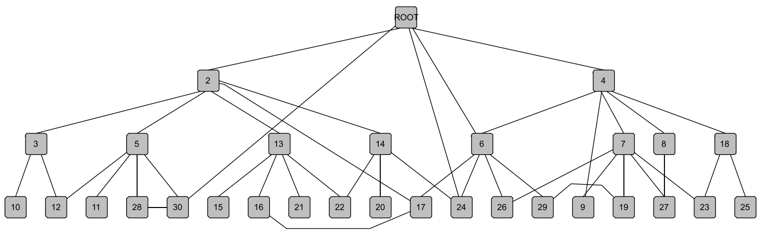


**Figure 1.1:** Additional relations introduced in a traditional tree layout. Different types of relations cannot be distinguished and edge intersections are introduced.

Typical encoding techniques to circumvent misinterpretations include using color, texture, size, or different node shapes to emphasize additional relations instead of direct links. However, psychological research has found that connectedness can be a more powerful grouping principle than proximity, color, size, or shape (PALMER and ROCK, 1994). The development of a visualization for relations in hierarchical data draws from these findings. This thesis introduces an attempt to solve the mentioned problems of a typical display of direct links on tree structures.

One of the most severe problems in visualizing tree structures is the limited screen space offered by common desktop displays. Trees easily require large aspect ratios which force parts of the tree to be cut from the display. The field of focus+context presentation deals with the display of large information spaces by taking the interest of viewers in parts of the data into account. Research from this field will be integrated into the visualization presented in this thesis to alleviate the display of large hierarchical data structures.

## 1.3 Results

This thesis introduces a novel visualization for the display of relations in hierarchical data structures. The chosen tree and relation layout solve the mentioned problems of displaying direct links between two data items in a tree structure. A particularly novel aspect is the consideration of relations in the development of focus+context navigation and presentation.

A nested tree layout was chosen for the visualization of tree structures to be able to restrict the visualization in space. An effective use of screen real estate is also achieved through the visualization of relations as arc shaped glyphs that can be adapted in height and shape. Zoom and filter mechanisms help to interactively explore tree content. Focus+context navigation offers a user-centered automation for interaction on the tree. Encodings for attributes of the data and affordances of displayed data items were carefully chosen. An example of the developed *arc tree visualization* can be seen in Figure 1.2.

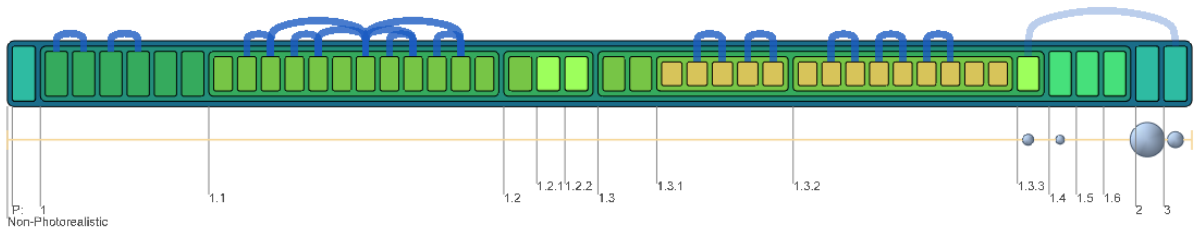


**Figure 1.2:** An overview of the developed *arc tree visualization*.

The work presented in this thesis was implemented as a JAVA3D application. JAVA3D is a high level, scene graph based API which offers sophisticated graphics rendering utilities. The underlying tree structure was implemented using the composite design pattern as introduced by COOPER (1998). The composite pattern allows the definition of a class hierarchy of simple objects and more complex composite objects so that they appear to be the same to the client program. All screenshots which serve as examples for the developed visualization were taken from the implemented application.

## 1.4 Thesis Organization

The following sections briefly introduce the contents of subsequent chapters.

### Chapter 2

An overview of the research field that forms the framework for this thesis is presented in Chapter 2. It introduces the field of visualization with particular emphasis on information visualization. Further research presented in this thesis is influenced by several research areas in the information visualization community. These areas are concerned with the presentation of different data sets: linearly ordered data, hierarchical data, and graphs. The discussion of visualization techniques from these areas particularly emphasizes the presentation of relations within the data.

### Chapter 3

Chapter 3 begins with the introduction of the specific challenges for the development of a visualization. In subsequent sections a layout for linearly ordered hierarchical data structures is developed. Emphasis is placed on the display of additional relational information between nodes in the tree. As a prerequisite to the development of the visualization a set of definitions for the underlying data structure including the relational information as well as the drawing and visualization of the data are discussed.

### Chapter 4

Interaction and navigation are two important aspects of an information visualization. This chapter is concerned with the development of interaction techniques for the visualization developed in the previous chapter. First, general operations on the data set are described. Focus+context techniques extend these operations to aid navigation in the visualization. The presented techniques act as a filter mechanism to eliminate unwanted information and avoid information overload.

### Chapter 5

Case Studies introduce possible application domains for the developed visualization. It is shown how the requirements specified for the visualization task were implemented in context to a particular application. The remainder of the chapter establishes how an evaluation of the visualization can be carried out without being restricted to a specific application domain. In addition, aspects for which an evaluation would be most appropriate at this stage of the development of the visualization are highlighted.

### Chapter 6

This chapter concludes the thesis and summarizes the key contributions of the presented work. Areas for future work are highlighted in the remainder of the chapter.

# CHAPTER 2

# Related Work

The following chapter gives an introduction to the research field that forms the framework for this thesis. It begins with an introduction to the field of visualization, in particular information visualization and its visualization techniques in Section 2.1. Subsequent sections give an overview of previous research in information visualization with respect to the visualization technique developed in this thesis. Emphasis is placed on assessing the visualization of relations in presented techniques. Section 2.2 addresses visualizations of linearly structured data. Representations for hierarchical data are introduced in Section 2.3 followed by a short overview of graph drawing techniques in Section 2.4. Finally, Section 2.5 attends to visualization techniques that were specifically developed to emphasize relations in data.

## 2.1 Visualization Field

Visualization, in general, refers to the graphical representation of data or concepts. It involves the building of a mental model of data typically with the goal of supporting decision making. Ware defines five advantages of visualization (Ware, 2000):

1. **Comprehension:** Visualization supports the comprehension of huge amounts of data.
2. **Pattern Perception:** Previously unnoticed properties of data may emerge in visualizations.
3. **Problem Analysis:** Problems within the data may become immediately apparent.

4. **Adaptability:** Visualization facilitates understanding of large- and small-scale features of data.
5. **Interpretation:** Hypothesis formulation is facilitated by visualization.

Of course, these advantages only apply to *good* visualization design. The design criteria for such visualizations will be the subject of Section 3.2.

Two main areas of visualization have evolved in the computer science community: *scientific visualization* and *information visualization*. Scientific visualization is primarily concerned with displaying real or simulated scientific data. Basic visualization techniques for this area include surface rendering, volume rendering, and animation (SCHUMANN and MÜLLER, 2000). Typical examples include processing of satellite photographs or medical data.

The work in this thesis is not concerned with scientific visualization since the data basis for the presented visualization is not primarily scientific. The following section is, therefore, devoted to the latter area of information visualization.

### 2.1.1 Information Visualization

The field of information visualization is influenced by many different research domains including computer science, psychology, semiotics, graphic design, and art. In the field of computer science, computer graphics and human-computer interaction (HCI) are the most relevant areas. Computer graphics constitutes the basis for the practical implementation of visualization ideas and concepts while HCI is involved through its study of how people work with computers and how interfaces and programs can be designed to help people to effectively use them. Different areas of psychology offer guidance for visualization tool designers on how humans perceive and process visual information. Semiotics is a related subject in the humanities. It involves the study of symbols and how they convey meaning. Design is integrated through its knowledge about the process of creating visual artifacts and arrangements to satisfy a specific purpose. Finally, fine art has developed many styles and aesthetics for conveying visual meaning in sub-disciplines ranging from drawing to cinematography.

Information visualization deals with creating visual or graphical aids to access, distribute or explain data. CARD et al. give a general definition for information visualization as

> *"the use of computer-supported, interactive, visual representations of abstract data to amplify cognition."* (CARD *et al., 1999)*

Taking this definition, information visualization is a means of creating visual aids that lead to insight in the underlying data sets. In this sense it is not about producing nice pictures but about making data understandable and explorable so that the visualization helps us gain knowledge about the data. It is the process of forming a mental model for the acquired data and so helping the viewer to understand underlying concepts, patterns, and connections within the data (SPENCE, 2000). Data sets in information visualization typically come from large information spaces or information systems that need to be made accessible and understandable to the user. For the production of good visual representations of data several visualization techniques have been developed which are discussed in the following.

### 2.1.2 Visualization Techniques

The evolution of computers has lead to many new means of producing visualizations for collected information. Much of the work in this field focuses on creating graphical interfaces for complex datasets stored in databases. Often, such visualizations are combined with means for interactive exploration as mentioned in the definition given for information visualization. Visualization techniques include concealment of data, using a three-dimensional space, layering data, scaling techniques to provide more space for certain information (e. g., focus+context), overview+detail techniques, and taking advantage of psychological principles of layout such as proximity, alignment, and shared visual properties (e. g., color). These techniques all take advantage of the properties of the graphics system. BERTIN introduces these as visual or graphical variables (BERTIN, 1983). These will be the topic of the following discussion.

#### Graphical Variables

The notion of graphical variables was introduced as early as 1918 in the work "Semiology of Graphics" (BERTIN, 1983, Translation of: Sémiologie graphique, 1918). Graphical or retinal variables[1] form the scope of each graphics system. The 2D plane receives a central position as the canvas on which all graphical representations are assigned a *position* (cf. Figure 2.1). Graphical variables are factors for depth perception and can help designers to add a third dimension in a visualization on a two-dimensional plane. BERTIN identifies the following variables:

| | |
|---|---|
| **Size:** | Height, width, or length of objects on the plane, |
| **Value:** | Various shades of gray of objects on the plane, |
| **Texture:** | Variations of fineness and coarseness of a texture on an object, |
| **Color:** | The hue of an object, |

1 The variables are called *retinal* since the eye is sensitive to them.

**Orientation:** Orientation of objects or patterns, and

**Shape:** Categories of shapes for objects.

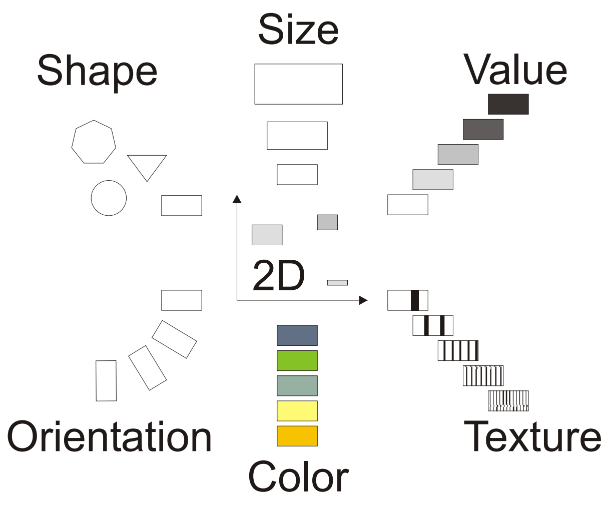


**Figure 2.1:** Graphical variables according to BERTIN (1983).

Any of these graphical variables can be used in building representations for objects. However, not every variable is necessarily well suited for encoding non-visual properties of data. According to BERTIN, next to the described retinal properties, information is encoded using positional (1D, 2D, and 3D) or temporal (animation) variables. The tools for encoding information are graphical marks such as points, lines, and area.

An area of human cognitive psychology attends to *preattentive processing* of visual properties. A limited set of visual properties or features has been identified to be processed automatically by the human vision without the need for focused attention. It has been shown, for example, that numerical information encoded using hue or orientation can be rapidly processed and estimated by humans (HEALEY et al., 1996). Table 2.1 presents an overview of features which are known to be processed automatically by the human visual system. These features are also candidates for encoding information visually (CARD et al., 1999).

## Gestalt Laws

*Gestalt laws* have been introduced by the Gestalt School of Psychology founded by Max Wertheimer in 1912. The basis of Gestalt psychology is the insight that we often perceive things that are not part of our simple sensations. Although much of the research in Gestalt psychology has been refuted, the Gestalt laws offer an enduring value for visualization. These laws are concerned with pattern perception and offer

| | | |
|---|---|---|
| Line orientation | Length | Width |
| Size | Curvature | Number |
| Terminators | Intersection | Closure |
| Color | Intensity | Flicker |
| Direction of Motion | Binocular Luster | Stereoscopic Depth |
| 3D Depth Cues | Lighting Direction | |

**Table 2.1:** Visual features that can be preattentively processed. Table adapted from (HEALEY et al., 1996). For further reference to authors who describe preattentive tasks using the given feature please refer to the cited article.

valuable advice for encoding information. Eight Gestalt laws are introduced as design principles:[2]

**Proximity:** Objects near one another are grouped together into a perceptual unit. In a visualization, relations between items can be emphasized by placing them in close proximity.

**Similarity:** If more than one object type is presented there is a tendency to group similar items together. In a visualization, related visual elements should look similar. Similarity can be created on the basis of shape, value, color, orientation, texture, size, etc.

**Continuity:** Objects are grouped together if they form some type of continuous pattern. This implies that neighboring elements are perceived as a group if they are connected by straight or smoothly curved lines. In a visualization, relations can be encoded using connection.

**Symmetry:** Shapes are perceived as figures made up of combined symmetrical forms rather than individual asymmetric parts. In a visualization, symmetry can be used to relate visual entities.

**Closure:** Objects tend to be recognized by closed contours. Contours are also united when they are very close to each other and a single object is formed. In a visualization, closed frames can be used to segment display space and group elements.

**Relative Size:** Smaller components of objects tend to be perceived as objects. The biggest part is often perceived as background.

**Common Fate:** Elements moving in the same direction form a visual group. In a visualization, relations can be encoded using a uniform animation of objects.

2 For further information on Gestalt laws as design principles refer to (WARE, 2000) and (CARD et al., 1999).

**Familiarity:** Elements are more likely to form groups if the groups have a familiar shape or appear meaningful. In a visualization, elements can be related by placing them in a familiar or meaningful shape.

Once the basic visualization techniques and the graphical variables for building a visualization are known, a certain sequence of actions is required to build a final presentation. This sequence is known as the *information visualization pipeline*.

### Information Visualization Pipeline

The goal of the visualization process is to present abstract data in one of many possible ways. In the creation of visualizations several steps are traversed which are arranged in a visualization pipeline. CARD et al. (1999) introduce such a pipeline or reference model for visualization which is shown in Figure 2.2. The visualization process begins with the acquisition of *raw data* that is to be accessed, distributed, or explained. This data can be collected in any given format. A *data transformation* process then turns the raw data into a *data table* format and augments it with relational information and other metadata. Raw data might, for example, be turned into a hierarchical data structure with additional relational information or a network graph structure. At this step, a preselection or filtering action for unwanted data may also be performed. This is the most important step for the visualization tool designer since here his or her knowledge about visualization techniques and graphical properties is required. Visual mappings can also be performed on raw data, however, it often lacks a direct spatial component. Visual structures utilize such a spatial component for layout and add visible marks and graphical properties to encode information. The final *view transformation* creates a *view* on the visual structures by defining parameters of the graphics system like viewpoint position, clipping parameters, or distortion effects. Parameters for all steps in the visualization process can be influenced by humans, being either the visualization tool designer or the user of a visualization system. Users can, for example, directly manipulate data trough the interaction with visual structures or perform dynamic queries on the data and so further filter and select relevant subcategories of the data. The following section will now introduce SHNEIDERMAN's visual information seeking mantra as a basic design guideline for the creation of an interactive visualization. *Visual mappings* turn data tables into *visual structures*.

### Visual Information Seeking Mantra

The *visual information seeking mantra* proposed by SHNEIDERMAN (1996) summarizes many design guidelines for creating an interactive visualization. A three-step design is recommended: Overview first, zoom & filter, and detail-on-demand. These three steps

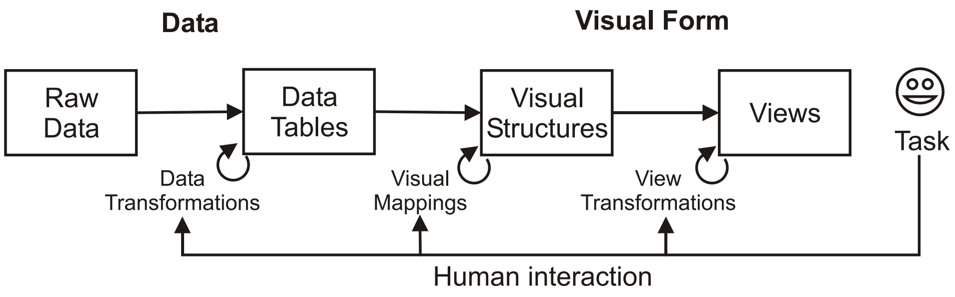


**Figure 2.2:** Visualization pipeline adapted from (CARD et al., 1999).

are the basic principles for browsing and searching in a visualization. An effective information visualization, therefore, works on a macro and micro level.

A visualization needs to provide an *overview* of the entire data set for the user to gain a good understanding of the data. The two main visualization techniques for providing such an overview are: *overview+detail* and *focus+context*, according to a definition by SCHUMANN (2003).[3] In an overview+detail display two images are provided, one detailed view and an overview image that shows the location of the detailed view in the relation to the entire data set. Both images are usually shown in parallel or sequentially. In the overview image the user can usually specify which region to show in the detailed view. Focus+context display of data, on the other hand, is a common characteristic of many visualizations. In a focus+context display the most important data lies in a focal region at large size and detail. Objects in the context outside the focal region help the viewer to relate the focus to the entire data structure. Both focus and context regions are integrated into one single view. Regions far from the focal region are usually displayed smaller or selectively omitted. Distortion of data representations is a common characteristic of focus+context displays. For full functionality of focus+context displays the focal region and magnification factors can usually be specified by the viewer. In the second step of the *visual information-seeking mantra*, *zoom and filter* operations are provided. The user has to be able to specify points of interest and selectively zoom in to explore the data. On the other hand, filter operations are necessary to leave out uninteresting data items and by this simplify the display of the data set. When the user has specified points of interest, *detail* should be provided *on demand*, e. g., by displaying tooltips or labels after the user has clicked on an item of interest.

The following sections will now be concerned with visualization processes. They will introduce visualization problems that relate to the task addressed in this thesis.

3 Overview+detail is sometimes also called detail+context, while focus+context might also be called fisheye views, distortion-oriented presentation techniques, elastic presentation space, etc.

## 2.2 Visualization of Linearly Structured Data

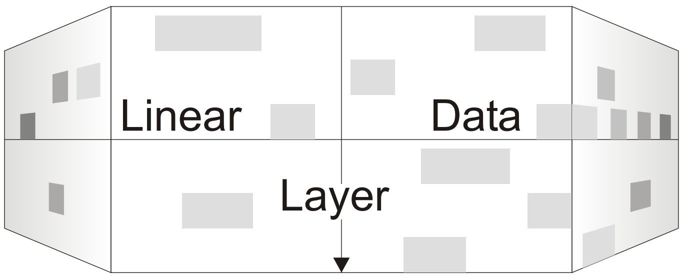
Data in today's work environments is often organized as linear structures according to some metric, e.g., chronologically (timetables, project records, etc.) or alphabetically (directories). Much of the time during a work day is used for managing data, locating information in data, or analyzing collections of data. In information retrieval, the position of an object along such a metric is used as a retrieval cue. It is set either directly by specifying the exact position of an item or by referring to a known landmark along the metric and continuing the search from this point. The visualization techniques presented in this section all aim at helping the user with the basic tasks involved in dealing with linear data: searching in the data and analyzing it. According to MACKLINAY et al. (1991), the principal obstacles for visualizing linear data lie in the presentation of large information spaces and the display of the extreme aspect ratio of linear data on the screen. This is the reason why focus+context presentations are often involved in these visualizations. This section presents landmarks in visualizations of linearly structured data and analyzes the work with respect to the research presented in this thesis. In particular, the presented relational techniques require attention in this context. Linear data already presents underlying relational attributes through the sorting order given by the applied metric. This type of relation is almost exclusively visualized through position. Other relational attributes in the data often need to be identified by the user through recognizing and mentally connecting color patterns, size ratios, textural attributes, etc. Table 2.2 gives an overview of techniques described in the following. The techniques are grouped into three categories. The first two describe general visualization techniques that are applicable to any linearly organized data. Three techniques for visualizing chronological data are followed by three particularly interesting techniques for the visualization of line-oriented or text-based data.

| Technique / Attributes | Dim | Linear Data Category | Primary Visual Attributes | Focus + Context |
|---|---|---|---|---|
| Perspective Wall | 3D | general | perspective,ordered layout | x |
| Table Lens | 2D | general | graphical & symbolic representations, cell size | x |
| Gantt Chart | 2D | chronological | ordered layout | - |
| LifeLines | 2D | chronological | ordered layout, color | - |
| Lifestreams | 3D | chronological | perspective | - |
| TileBars | 2D | line-oriented | color | - |
| SeeSoft | 2D | line-oriented | color, (size) | - |
| Value Bars | 2D | line-oriented | size | - |

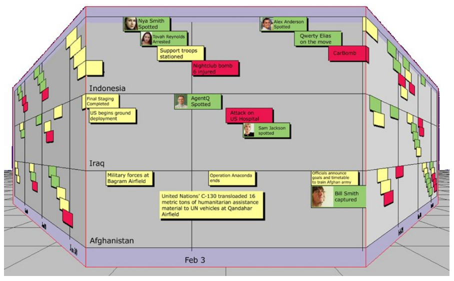
**Table 2.2:** Comparison of techniques for visualizing linear data.

### 2.2.1 The Perspective Wall

The *Perspective Wall* for visualizing linear information by integrating detailed and contextual views was presented by MACKLINAY et al. (1991). Two-dimensional layouts with wide aspect ratios are turned into three-dimensional visualizations by applying a folding metaphor. The visualization contains three panels a center panel for viewing detail and two side panels for viewing contextual information. The side panels are attached to the center panel and folded away from the viewer to form the three-dimensional visualization as shown in Figure 2.3(a). By applying a perspective projection, the information on the far ends of the side panels is automatically distorted and reduced in size. For typical values of the parameters for the wall, detail on the center panel is at least three times larger than detail on a flat wall that would fit inside the current field of view. Therefore, the *Perspective Wall* displays three times as much information since it provides contextual information which would otherwise be hidden outside the field of view in a flat wall display. However, some screen space is wasted above and below the side panels of the display. Linear information on the wall is displayed from left to right while the vertical dimension can be used for layering information. The visualization does not originally provide means to identify relations and references between data items except for the underlying relations given through the sequential ordering and layering. Other relations can be encoded through coloring or texturing, for example.



(a) Schematic overview of the Perspective Wall.

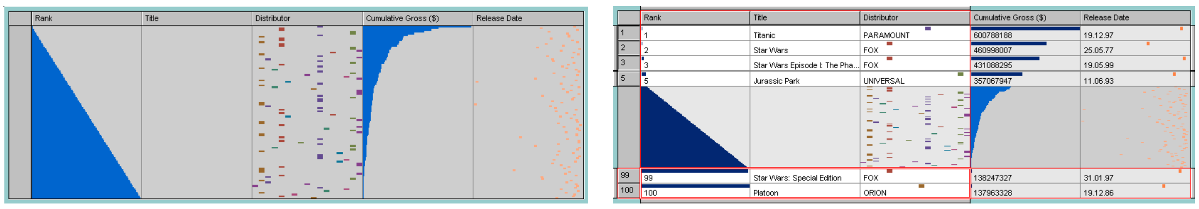


(b) Implementation of the TimeWall by Inxight® Software, 2004.

**Figure 2.3:** Schematic overview and implementation of the *Perspective Wall*.

### 2.2.2 Table Lens

Tables can help to present linear data by highlighting the underlying *logical structure* through the use of a special *visual structure*. The visual structure of a table usually comprises many equally-sized and consecutive blocks. These blocks contain information of various kinds. An ordering scheme controls the arrangement of consecutive blocks according to their information content. Typical one-dimensional tables are lists, outlines, or one-dimensional arrays or vectors such as a to-do list or table of contents. Multi-dimensional tables, on the other hand, store information in more than one row or column of blocks and, therefore, consist of several interrelated one-dimensional tables. Multi-dimensional tables are the underlying data structure of the *Table Lens* visualization technique presented by RAO and CARD (1994). *Table Lens* uses a focus+context technique to support the user in fluidly exploring detail in table data without losing its framing context. A *degree of interest function* controls mapping of interest levels in different cells to the respective cell addresses while a transfer function maps these cell addresses to physical locations. Data in table cells is visually encoded depending on several factors: the value to be displayed (e. g., using bar charts to encode quantity), the value type to be displayed (e. g., using different types of charts), the region type (e. g., different presentation styles for data in or outside the focal area), user choices (e. g., number of colors), and finally spotlighting (e. g., for highlighting cells by values). A set of interactive operations is defined on the table to enable focus changes, sorting, and zoom operations. An implementation of the *Table Lens* is shown in Figure 2.4. References between data items are encoded in the underlying tabular layout. However, relations between data elements in unconnected cells have to be identified by the user interactively. Cells can be compared by placing focus regions on the cells of interest and analyzing the cells' contents according to the graphical representation provided for these cells.



(a) Table of the top 100 movies sorted by cumulative gross.

(b) The same table as in 2.4(a) with focus on the top five and last two movies.

**Figure 2.4:** *Table Lens* implementation by Inxight® Software.

Tables are also a tool frequently used for the display of data in chronological order. The following section, therefore, describes different techniques used in the visualization of chronological data.

## 2.2.3 Visualizations for Chronological Data

Timelines are the most common and simplest technique for visualizing chronological data. Common chronological data sets include project management records or personal history records such as medical records. *Gantt charts* have emerged as a graphical representation of the duration of tasks against the progression of time. They are used mainly for planning and scheduling projects and are included in many commercial project management packages like MS PROJECT.

The *LifeLines* approach presents a visualization tool for personal history records such as medical or legal data (PLAISANT et al., 1996). This visualization tool is capable of displaying several different aspects of a personal record in one overview while providing more detailed information through operations like rescaling, regrouping, highlighting, etc. The visualization of relations between aspects of a person's record is given through an ordered layout, highlighting as a response to user interaction, coloring of related record entries, and the design of graphical items representing time periods or events.

*LifeStreams* presented by FREEMAN and FERTIG (1995) are an organizational metaphor for time ordered streams of documents. This technique is aimed primarily at visualizing a diary of a user's electronic life, e. g., his or her personal data files, electronic mail, or schedules. The *LifeStream* visualization consists of a three-dimensional representation of an ordered list of data. The tail represents the oldest data files and is virtually placed the furthest away from the viewer, the head of the list includes future data files, like reminders, schedules, to-do lists, etc. (cf. Figure 2.5). Relational attributes in lifestreams are encoded through the border color of the represented items and the thickness of the lines.

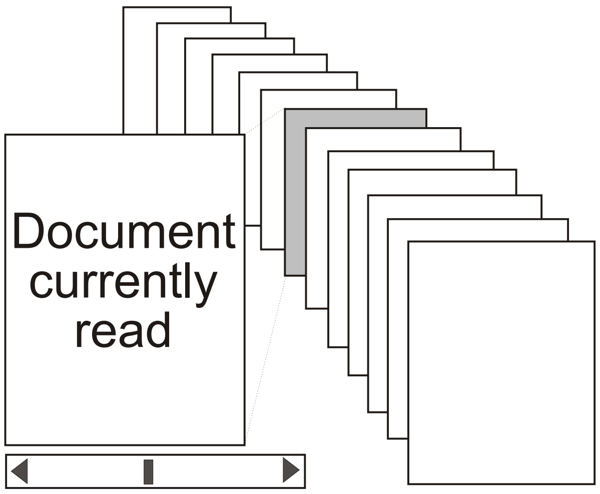


**Figure 2.5:** Schematic overview of *LifeStreams*.

A sophisticated use of graphical representations for qualitative and quantitative attributes of time-oriented data is presented by BADE et al. (2004). Different levels of abstraction are introduced to present timelines or segments of timelines at different levels of detail. Encoding techniques such as color and/or height as well as icons are used to represent qualitative and quantitative attributes. For the display of high frequency data an adapted *Information Mural* and a variation of the TUKEY box plot are introduced. With additional interaction techniques this visualization is capable to support exploration of data for users with different research tasks.

Chronological relations are commonly visualized through sequential ordering. Items placed in time are typically arranged by displaying the oldest items on the left and the newest items on the right. Through this ordering, "before" and "after" relations are easily identified. Further indications of relations in time have to be derived from additional information such as labels indicating exact time or distance between items indicating how far apart these items have been placed in time. In the visualization of line oriented data such conventions are less obvious leading to seemingly different visual representations as can be seen in the following.

### 2.2.4 Document Visualization

The visualizations presented in this section are primarily concerned with displaying attributes of textual documents. If these attributes are presented in an intuitive way the user may be supported in document retrieval through gaining a general understanding of the document's overall contents. Visualization of textual attributes may, therefore, be important for several reasons: to give the user an understanding of the general assembly of a document in terms of structure or the occurrence of search terms and by providing an overview aiding the user in finding suitable documents, helping him or her to edit documents, or by merely helping in reading the document. The following visualization techniques demonstrate how some of these goals can be accomplished.

#### TileBars

The *TileBars* approach aims at visualizing text structure to ease information retrieval from document collections (HEARST, 1995). Text structure is visualized according to term distribution. Term distribution is measured by determining the frequency of search terms from a boolean-type query in document segments. Relative document length, query term frequency, and query term distribution are considered. For each document in the collection a diagram similar to the one in Figure 2.6 is created. Each row in the diagram displays the distribution of one search term while each column represents a segment in the document such as pages, paragraphs, or chapters. The wider the diagram the more sections are contained in the text. In this visualization, relations between documents and keywords are displayed in a compact and coherent way. Bars for each set of query terms are lined up next to each other and diagrams for each document are usually visualized in a linear ordering to ease comparison. Therefore, relations are encoded by position and saturation.

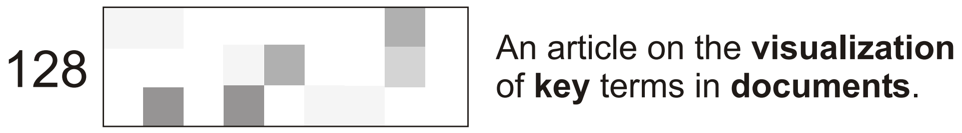

128
An article on the **visualization** of **key** terms in **documents**.

**Figure 2.6:** A typical *TileBars* diagram for three search terms.

## Seesoft

Providing statistical information on lines of code in a compact and interactive visualization is the goal of the *Seesoft* system (EICK et al., 1992). A schematic overview of the visualization is shown in Figure 2.7. Four key ideas are presented:

- A *reduced representation* is created by organizing the data in columns of one column per file to be analyzed. Longer columns represent larger files. Lines of code are represented by thin rows in the columns, either indented according to line length or non-indented (see File1 and File2, respectively, in Figure 2.7).
- A color scale encodes statistical information on the lines of code such as age, date of last modification, etc. The rows are, therefore, represented through *coloring by statistic*.
- *Direct manipulation* on the data in the visualization is performed by selecting or deselecting files and lines of code in the files or by selecting color regions on the color scale.
- The *capability to read actual code* is given through an overview+detail technique. A small lens can be moved across the columns to select areas of interest. A reading window then displays the actual code represented by these lines.

*Seesoft* is a visualization technique for linear or line-oriented data which has underlying statistical information to be displayed. Relations between lines are encoded by color. The viewer has to identify patterns to make visual comparisons between related line blocks.

An extension of the *SeeSoft* system was proposed by JERDING and STASKO (1998). The *Information Mural* technique is introduced for displaying large information spaces in a single-display window. The goal is accomplished by mapping data to the available pixels on the screen and through using color to encode the number of pixel, objects, or points mapped to a single pixel.

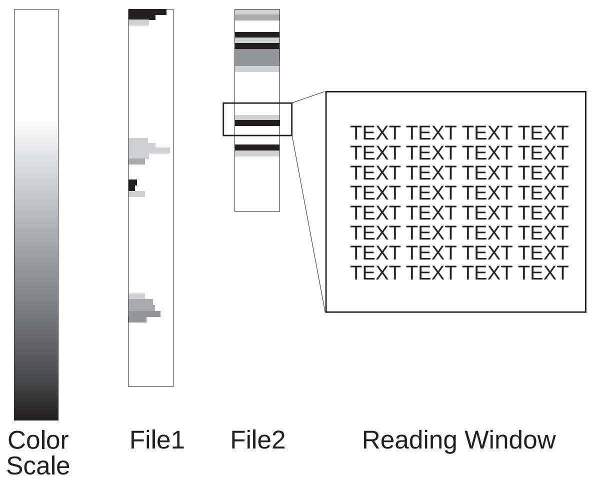


**Figure 2.7:** Schematic overview of the *SeeSoft* visualization.

### Value Bars

*Value Bars* were developed to visualize line-oriented data with quantifiable attributes (CHIMERA, 1992). The visualization concentrates on displaying “important” items discerned by item values or weights. *Value Bars* is a technique resembling the *Seesoft* visualization in its layout design. However, a different algorithm and a weighting function are introduced for the visualized items of text. In the layout, thin, vertical columns are attached to the window, usually a text display, containing the underlying data. The items in this data are represented as heights in the value bar in a linear ordering. The height of the item is deduced from the item’s weight or value relative to the total weight of all items. Once calculated for a document, the value bar remains unrevised. Therefore, the visualization presents an overview of items where relations are recognized from comparing heights in the value bar. Navigating the value bar requires low cognitive load.

Having introduced visualizations for linearly ordered data, the following section will be concerned with data that can be stored hierarchically in tree structures. The deepest level in the hierarchy may sometimes require or inherently present a given linear order of the data items. Therefore, the reader might be reminded of some of the techniques presented in this section when advancing to Section 2.3.

## 2.3 Visualization of Hierarchical Data

Hierarchical data, commonly saved in tree structures, has a long visual history. From very early paintings of family trees in Figure 2.8(a) to genealogical trees in early biological textbooks in Figure 2.8(b), new forms of family trees in Figure 2.8(c), or a fairly recent painting with different levels of emphasis in Figure 2.8(d), the general problem has remained the same: how to fit a tree and its accompanying information in the available display space. The following visualization techniques will introduce innovative ways in which this basic layout problem has been approached.

### 2.3.1 General tree layout

Tree layouts are commonly encoded using either connection or containment. Traditional tree diagrams, also called *node-link diagrams*, are created using connection in that nodes are directly linked with edges. Containment is used to create nested tree visualizations. In a traditional layout child nodes are positioned below a common ancestor. The best known algorithm for such layouts was presented by REINGOLD and TILFORD (1981). Disadvantages of the traditional tree representation lie in the poor use of the available display space due to the large amount of empty space included. Also, traditional tree layouts can easily grow very large and require extreme aspect ratios to be displayed. The traditional layout algorithm is acceptable for displaying small trees where nodes can still be shown in a size that allows the encoding of additional information such as labels, color, or size.

An in-depth analysis on connection and containment as encoding techniques for trees was conducted by CARD et al. (1999). Please refer to Section A.1 in the Appendix for examples of common tree representation.

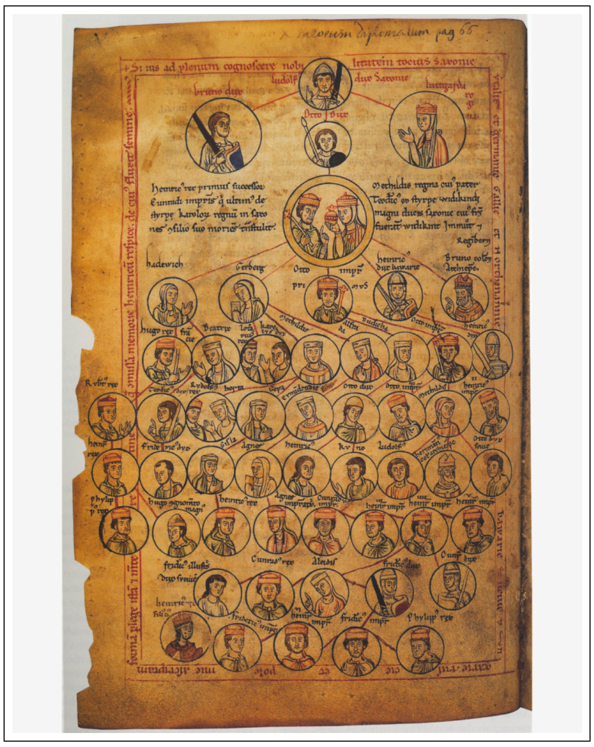
Typical data sets stored in tree structures include the already mentioned genealogical trees, organizational charts, mathematical formulas, tables of contents, library catalogues, file systems, and many more. The following section will introduce attempts to solve tree layout problems using computational assistance. Table 2.3 gives an overview of the presented techniques. Visualizations are categorized according to three different layout techniques:

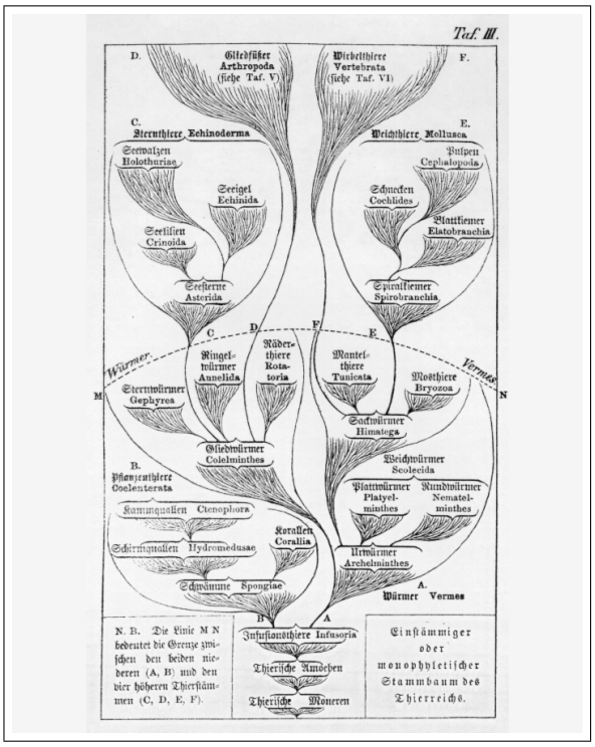
**Simple Tree Layout:** Aesthetic criteria are met such as minimal line crossings or placing nodes of the same tree depth at the same level. Space is not an issue.

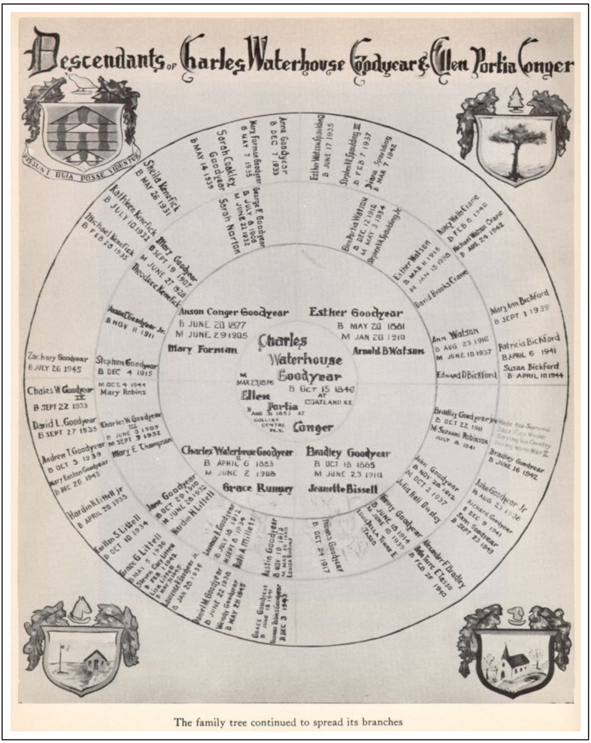
**Compressed Tree Layout:** This layout resembles the simple tree layout but spatial compactness is adhered to.

**Containment Tree Layout:** This layout encodes the tree structure using containment creating a nested tree layout.

These layout techniques can have additional attributes, in particular interaction and focus+context display. *Interaction* allows users to change the view of the tree, e. g., by expanding and collapsing subtrees. The tree does not necessarily have to fit into


(a) Genealogical tree, 13th century (PUHLE, 2001).


(b) Genealogical tree (HAECKEL, 1868).


(c) Goodyear family tree (GOODYEAR, 1950).

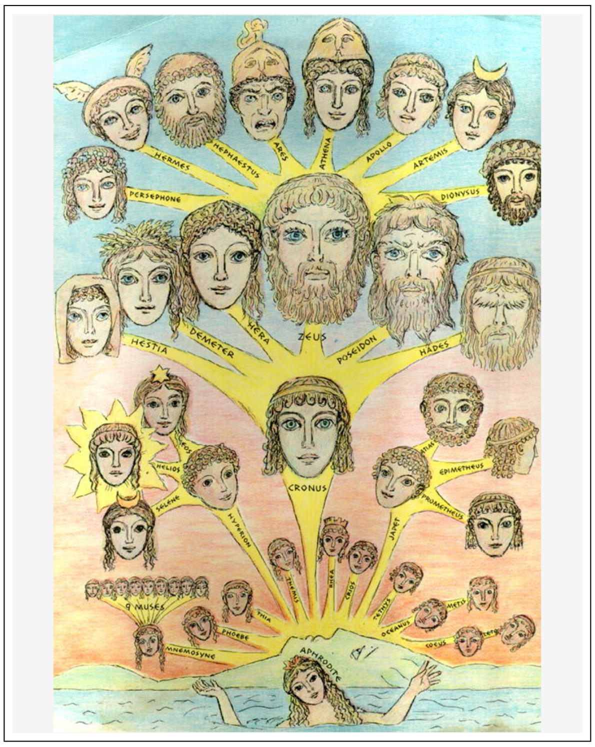

(d) Family tree of Greek Gods and Godesses (CARTHY, 1997).

**Figure 2.8:** Early visualizations of tree structures.

the view. In a *focus+context* display the view of the data can be adjusted to provide a large display of nodes in the focus area and scaled or clustered nodes or branches in the context area to make the tree fit the view area.

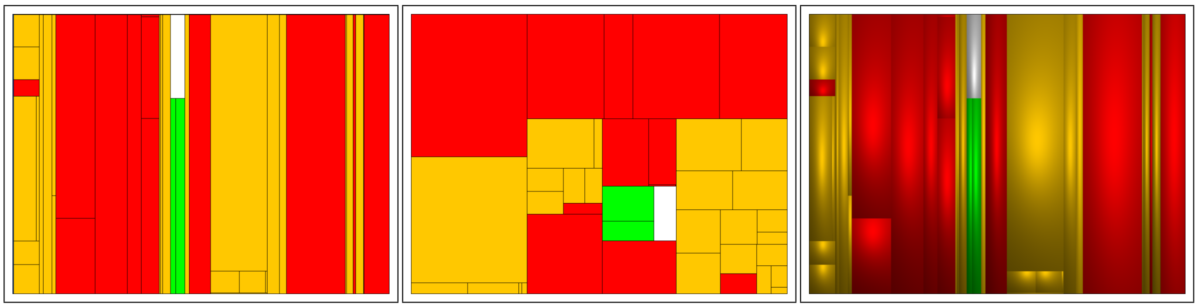
In the following, a number of examples for these tree layout techniques will be discussed. Visualization techniques with a containment tree layout are the topic of Section 2.3.2 and Section 2.3.3. These are followed by a description of approaches primarily using a compressed layout technique including focus+context display of tree structures.

| Layout / Attributes | Simple | Compressed | Containment | Interaction | Focus+ Context |
|---|---|---|---|---|---|
| Traditional Layout | x | - | - | - | - |
| Tree-Map | - | - | x | x | - |
| Information Slices | - | - | x | x | x |
| WebTOC | - | x | - | x | - |
| Cone Tree | - | x | - | x | x |
| Hyperbolic Browser | - | x | - | x | x |
| H3 | - | x | - | x | x |
| Cheops | - | x | - | x | x |
| Generalized Fisheyes | - | x | - | x | x |
| TreeJuxtaposer | - | x | - | x | x |
| DOITree | - | x | - | x | x |
| SpaceTree | - | x | - | x | x |

**Table 2.3:** Comparison of techniques for visualizing hierarchical data.

## 2.3.2 Tree-Maps

The *Tree-Map* visualization published by JOHNSON and SHNEIDERMAN (1991) and SHNEIDERMAN (1992) has received considerable attention over the past years. The original idea intends to present tree structures in a two-dimensional space-filling approach with a nested representation similar to Venn diagrams. Each node in the *Tree-Map* is drawn as a rectangle with a given size. The size of each node is chosen with respect to an inherent node attribute such as the node size. For drawing a *Tree-Map*, the available screenspace is partitioned into rectangular sections. The first rectangle forms the root of the tree. The tree layout algorithm partitions the root rectangle into $n$ parts with $n$ being the number of the root's children. The algorithm is recursive, however, nodes are partitioned vertically at even levels and horizontally at odd levels. Several new layouts have been proposed over the past years some of which are presented in Figure 2.9. The images show the same file structure using different display algorithms. The original *slice-and-dice* approach is seen in Figure 2.9(a). The *squarified tree-map* method seen in Figure 2.9(b) supports low aspect ratios. *Cushion tree-maps* in Figure 2.9(c) visualize structure by shading. Relations between data items in the tree are encoded trough a containment hierarchy, size of nodes, or additional shading. Data parameters are usually encoded using color. A technique for direct visualization of relational information on Tree-Maps will be discussed in Section 2.5.3.



(a) Original Tree-Map by SHNEIDERMAN (1992). (b) Squarified Tree-Map by BRULS et al. (2000). (c) Cushion Tree-Map by VAN WIJK and VAN DE WETERING (1999).

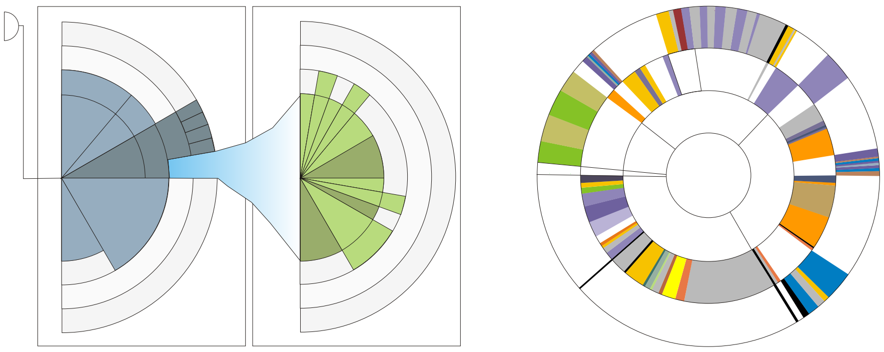
**Figure 2.9:** Tree-Map Visualizations.

### 2.3.3 Information Slices

A version of the tree ring (cf. Appendix A.1) representation was introduced by ANDREWS and HEIDEGGER (1998). Large hierarchies are visualized as a series of one or more semi-circular discs. Each disc represents approximately five to ten tree levels. Larger trees are visualized using cascades. Child nodes are fanned out according to their total size. Figure 2.10(a) shows levels 5–10 of a tree with levels 1–4 iconified at the top left of the visualization. Similar techniques using a full circle to display hierarchies have been introduced by CHUAH (1998) as well as STASKO and ZHANG (2000). An example is displayed in Figure 2.10(b). Relations in these visualizations are shown through the placement and size of nodes and coloring by node parameters. The *Information Slices* project shows a focus+context technique in that it displays only a certain number of levels in the hierarchy and iconifies the context nodes.

### 2.3.4 WebTOC

An example for the visualization of websites with a hierarchical table of contents was presented by NATION (1998) where a typical tree visualization is overlaid with statistical information. A graphical representation is placed next to the textual representation

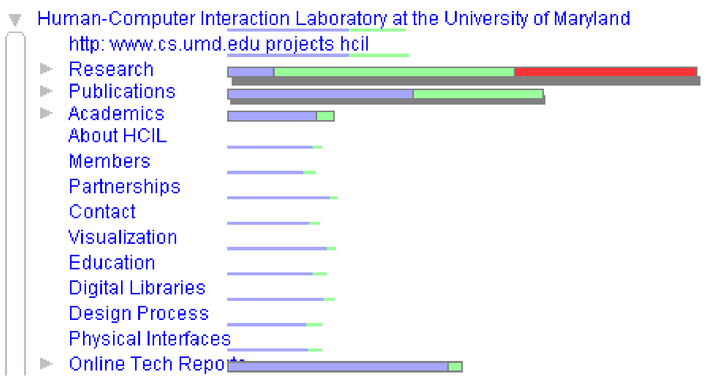


(a) Information Slices Image adapted from (ANDREWS and HEIDEGGER, 1998).

(b) Sunburst. Image adapted from (STASKO and ZHANG, 2000).

**Figure 2.10:** Radial tree visualizations.

of a node in the tree. It consists of a colored line where the size of the line represents the file size and its color stands for the file type (cf. Figure 2.11). For non-leaf nodes lines for linked documents are collapsed into a thicker "size bar". The size of the shadows under these size bars represents the number of items subordinate to the document. Textual representations for nodes can be removed to allow for easier comparison of line sizes and the identification of patterns. Relations between nodes are, therefore, not simply displayed through the linked representation but also through comparison of graphical representations. Size and color are used here as encoding techniques. *WebTOC* belongs to the compressed tree layout method since the tree is represented by a node-link diagram where the space problem is solved by providing a scroll bar and interactive methods to collapse and expand tree branches.



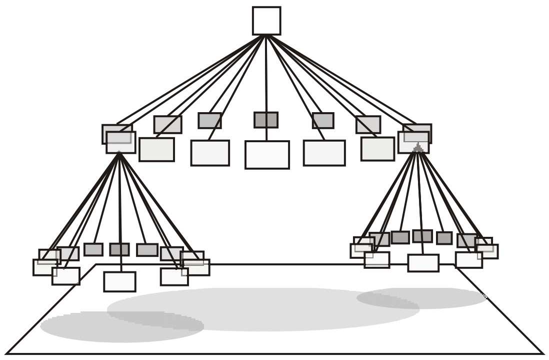
Human-Computer Interaction Laboratory at the University of Maryland
http: www.cs.umd.edu projects hcil
Research
Publications
Academics
About HCIL
Members
Partnerships
Contact
Visualization
Education
Digital Libraries
Design Process
Physical Interfaces
Online Tech Repo

**Figure 2.11:** Example of a *WebTOC* taken from the University of Maryland's Website.

### 2.3.5 Cone Tree and Cam Tree

Probably the earliest three-dimensional visualization of hierarchical data was presented by ROBERTSON et al. (1991). Nodes of a tree are laid out on three-dimensional cones. The root node is placed at the apex of the cone while the child nodes spread evenly on the baseline of the cone. This process is repeated recursively for all other child nodes. Each cone in the tree has the same height while the cone base diameters are adjusted on each level so that the lowest level will still fit in the available view frustum. The traditional *Cone Tree* is created with the root node at the top of the display making it difficult to place labels for all nodes (cf. Figure 2.12(a)). An approach to overcome this problem is presented in the form of *Cam Trees* where the root node is on the left of the display with the child nodes stretching to the right (cf. Figure 2.12(b)). Here, labels can be displayed in the graphical representation of the nodes themselves.

One of the main problems in three-dimensional visualizations is occlusion. It is addressed here by introducing transparency for the display of nodes and shadows for presenting an additional two-dimensional representation of tree properties. *Cone Trees* were introduced as a focus+context technique making use of the inherent perspective projection in the three-dimensional visualization. Nodes closer to the virtual camera are displayed relatively larger than those further away. Important aspects of this visualization are the presented interaction and animation techniques. When the user selects a node, it is smoothly rotated to its closest position with respect to the virtual camera allowing an easy assimilation of changes across views. Parent-child relations between nodes are encoded through links and position in the tree layout. In addition, color as well as labels can be used to display node properties.


(a) Schematic overview of a Cone Tree.

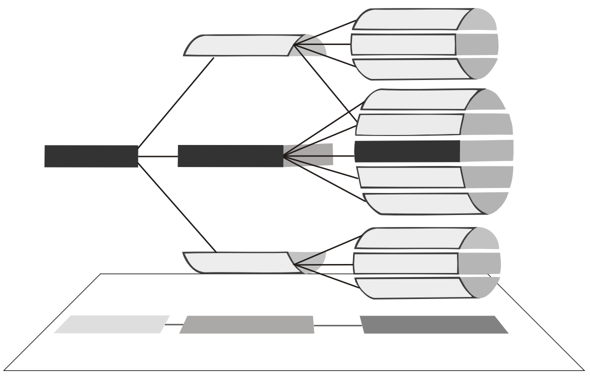
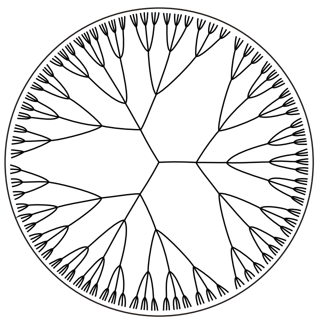

(b) Schematic overview of a Cam Tree.

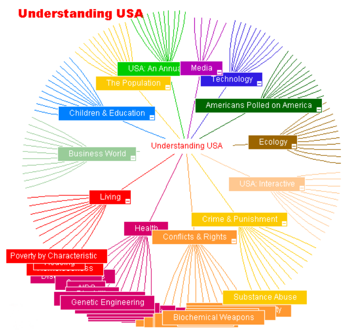
**Figure 2.12:** Cone and Cam Trees.

### 2.3.6 Hyperbolic Browser

Hyperbolic geometry is a non-Euclidean geometry meaning that EUCLID's fifth postulate[4] does not hold. Without going into the sophisticated mathematical detail, a hyperbolic plane can be displayed by using the POINCARÉ mapping with the result shown in Figure 2.13(a). The POINCARÉ disk model is useful in visualization since angles have their Euclidean measure and it is displayed in a bounded region of the Euclidean plane so that it can always be viewed in its entirety. With this mapping a tree structure of any size can be fitted within a finite area (a circle). The node in focus is displayed in the center while all nodes away from the central node and toward the perimeter of the circle are displayed exponentially smaller. The node in the center is displayed larger than any other node and, therefore, this technique can be categorized as a focus+context visualization. A commercial implementation of a hyperbolic browser can be seen in Figure 2.13(b).

As an extension, the *H3* layout technique (cf. Figure 2.13(c)) for drawing large directed graphs as node-link diagrams in 3D hyperbolic space was introduced by MUNZNER (1997). Hyperbolic representations of trees allow for a huge number of nodes to be represented. However, nodes at the perimeter are usually pruned for increasing rendering performance. Relations between nodes are encoded through direct linking of nodes and through their placement on the POINCARÉ disk. Additional relational information could be encoded through coloring or label placement.


(a) Uniform hyperbolic tree.



(b) StarTree by Inxight Software.

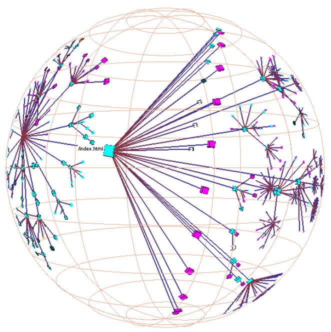

(c) H3 Browser.

**Figure 2.13:** Hyperbolic tree visualizations.

4 If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two right angles, then the two lines inevitably must intersect each other on that side if extended far enough (EUCLID's fifth postulate).

### 2.3.7 Cheops

A different visualization technique was introduced by BEAUDOIN et al. (1996). The *Cheops* explorer relies on a compression technique for visualizing huge hierarchies of up to one million nodes. High compression rates are obtained through the reuse of visual components. Nodes are displayed as triangles and layered according to their place in the hierarchy. In Figure 2.14 the tree on the left is shown as a *Cheops* visualization on the right. On the right, one middle triangle can stand for different child nodes of the parent layer. Node 7, for example, stands for either the last child of Node B, or the middle child of Node C, or the first child of Node D. To resolve the created ambiguity users can select branches of interest. The selected branches are then colored and shown in full. The original research does not describe techniques to encode additional relational information.

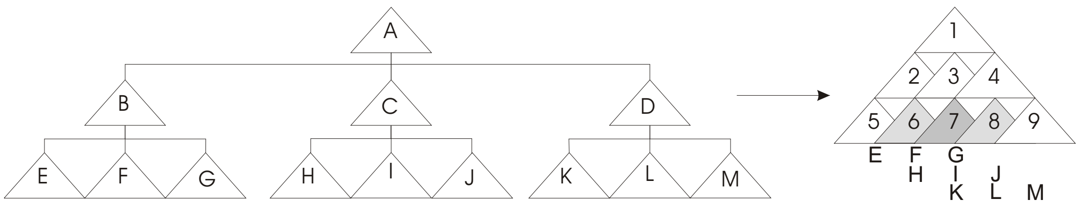


**Figure 2.14:** Creation of a *Cheops* tree visualization.

### 2.3.8 Focus+Context Exploration in Hierarchies

Focus+Context display of data is a common characteristic of many visualization systems. Examples for focus+context visualizations have already been introduced. Please refer to Tables 2.2 and 2.3 for exact references. This section will be concerned with visualizations where focus+context was primarily introduced to aid navigation and analysis in hierarchical data.

In his fundamental work on fisheye views FURNAS laid the framework for many focus+context techniques on hierarchical data sets (FURNAS, 1986). His basic strategy involves the specification of a degree-of-interest (DOI) for each node in the tree. The DOI of a node specifies the interest of the user in this particular node. It is calculated by combining an a-priori global structural importance and an a-posteriori importance that depends on direct user interaction. His concept is also the basis for interest calculations for the work presented in this thesis and will, therefore, be described in more detail in Chapter 4. Many other focus+context visualization techniques for tree structures draw from this simple but powerful calculation for a node's DOI. Three techniques are of particular interest and are, thus, discussed in the following.

The *TreeJuxtaposer* approach introduces focus+context navigation in large phylogenetic trees (MUNZNER et al., 2003). Re-arrangement of the tree layout is calculated using a technique that allows for guaranteed visibility of highlighted or in-focus areas at all times. TreeJuxtaposer was developed to support the comparison of large trees with several hundred thousand nodes at a time.

Another technique allowing focus+context navigation in hierarchies was presented by CARD and NATION (2002). The *Degree-of-Interest Tree* technique uses a general node-link diagram as its basic representation. Logical filtering, geometric distortion, semantic zoom, and clustering are used to produce a focus+context view of the tree. If a user states interest in a node, internal Degree-of-Interest (DOI) calculations predict the interest in other nodes and adjust the visualization accordingly. A resulting visualization with several nodes of interest can be found in Figure 2.15. Triangles in this visualization indicate node clusters. A very similar approach has been implemented in the *SpaceTree* project as described by PLAISANT et al. (2002).

As a generalization of the layout of hierarchical structures introduced in this section, the following section will briefly discuss issues and approaches relevant for the graph drawing community.

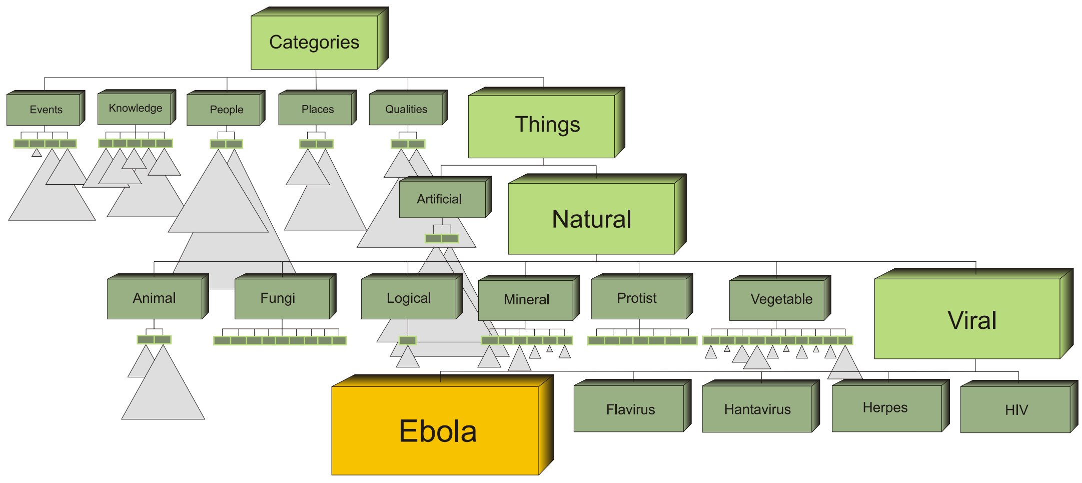


**Figure 2.15:** *Degree-of-Interest Tree* adapted from (CARD and NATION, 2002).

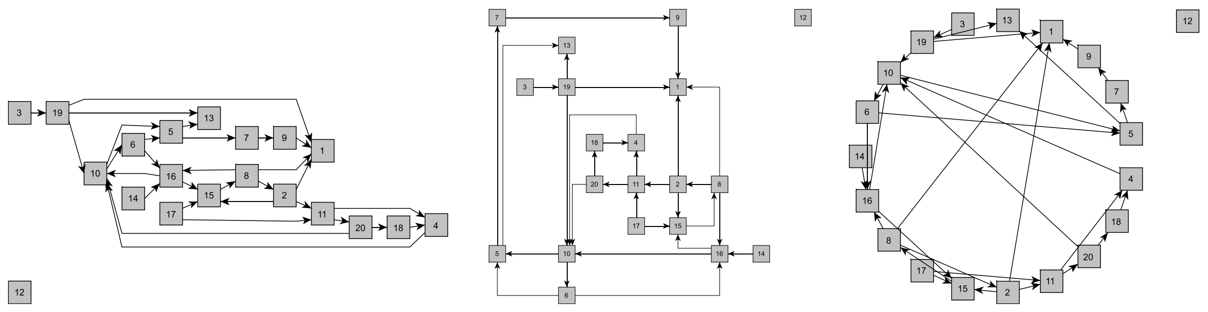
## 2.4 Graph Drawing

Many different approaches for graph layout exist that are typically classified as a branch of graph theory named *graph drawing*. These graph drawing techniques play an important role in information visualization. A good spatial layout of graphs increases the understanding of these often very complex structures while a poor layout can obscure structure and distract from important aspects of the data. General graphs are typically represented by node-link diagrams with a dot for every vertex and an arc connecting the endpoints of every edge. If the graph is directed the direction of an edge is indicated by an arrow. Connection is usually used as an encoding technique since, in contrast to tree structures, graphs may include cycles and other complexities that eliminate the option of using containment. Graphs are almost exclusively used to model relational structures. Data entities are stored in the nodes while relations are represented as the edges of a graph. Starting from this basic representation, the spatial layout of nodes and edges poses the major problem in graph drawing. Layout algorithms can be categorized according to the type of layout they generate (e. g., grid layout) or according to the methodology on which they are based (e. g., non-deterministic layouts). HERMAN et al. (2000) provide an excellent overview of graph visualization techniques. In information visualization specifically the aesthetic constraints for graph layout are of interest since these greatly influence the perception and interpretation on the side of the user. Aesthetic criteria involved in graph layout according to BATTISTA et al. (1999) include:

| | |
|---|---|
| **Crossings:** | Minimization of edge crossings. Ideally a planar graph. |
| **Area:** | Minimization of the area of the drawing. |
| **Total Edge Length:** | Minimization of the sum of the lengths of the edges. |
| **Maximum Edge Length:** | Minimization of the maximum length of an edge. |
| **Uniform Edge Length:** | Minimization of the variations in edge length. |
| **Total Bends:** | Minimization of the total number of bends along an edge. |
| **Maximum Bends:** | Minimization of the maximum number of bends on an edge. |
| **Uniform Bends:** | Minimization of the number of bends on an edge. |
| **Aspect Ratio:** | Minimization of the aspect ratio of the drawing. |
| **Symmetry:** | Display symmetries of the graph in the drawing. |
| **Angular Resolution:** | Maximization of the smallest angle between two edges incident at a node. |

Empirical studies suggest that, after reducing the path length, the two most important factors are *continuity* of the display of paths (including the minimization of bends and maximization of the angular resolution) and *edge crossings* (WARE et al., 2002).

Graph layout algorithms usually specialize on meeting just a few of these criteria since they are not mutually exclusive. A symmetric tree layout, for example, might require a certain number of edge crossings. Additional constraints can be placed on the drawing algorithm. This allows the user to pre-arrange certain shapes and give more control to the outcome of the drawing. Also, constraints can greatly increase the computational calculations for graph layout. Constraints may include placing a given vertex in the center or at the boundary, clustering, or pre-arrangements of paths (top-bottom or left-right). Figure 2.16 gives an overview of three different graph layouts and the constraints placed on them.



(a) Hierarchical Layout. Constraints on the minimal node and edge distances and the orientation have been selected in the creation of the graph.

(b) Orthogonal Layout. Contraints on the underlying grid and length of the edges have been applied.

(c) Radial Layout. Constraints on the minimal node distance and the radius have been applied.

**Figure 2.16:** Different layouts of the same graph. Layouts were created using YEd (2004).

There is not just one way to draw a graph and no algorithm will always draw the best graph for each purpose. Therefore, a number of paradigms for graph drawing were developed with basic steps on how to draw a graph. Each stresses different aesthetics or has other desirable features such as speed. An overview of these paradigms is given by Battista et al. (1999). Applications of graph drawing include genealogy, cartography (subway maps, for example), sociology, software engineering (visualization of connections between program modules), and visualization of hypertext links.

## 2.5 Visualization of Relations

Visualizing relations is one of the most essential tasks in information visualization. Understanding relations between items in a visualization helps the viewer to build a mental model of the underlying data. A mental model is needed to understand the scheme or situation to which the presented data refers. For making decisions based on the visualization of data the interpretation of this internal model is essential. Relations can be of the types described above being an inherent attribute of the data such as a linear structure of time, another metric, or a hierarchical parent-child relation. Other types of relations are given through additional parameters or attributes of the data or the addition of contextual data like labels, captions, etc. Typical encoding mechanisms for relations are color, shape, orientation, position, and the visualization through direct connections with lines or arrows which is an essential part of node-link diagrams and graphs. The latter type of encoding will be discussed further in this section. Again, a historical visualization in Figure 2.17 will serve as an introductory example. The figure displays a gamut with direct links between certain tones. Relations are displayed through direct visual connection and additional textual information. Clever use of cuts through these relational arcs helps the viewer to follow the arcs from the origin to their destination. The first two techniques are very similar to this historical example but include encoding techniques such as transparency or interactive animation. The last example shows direct relations displayed on a Tree-Map visualization.

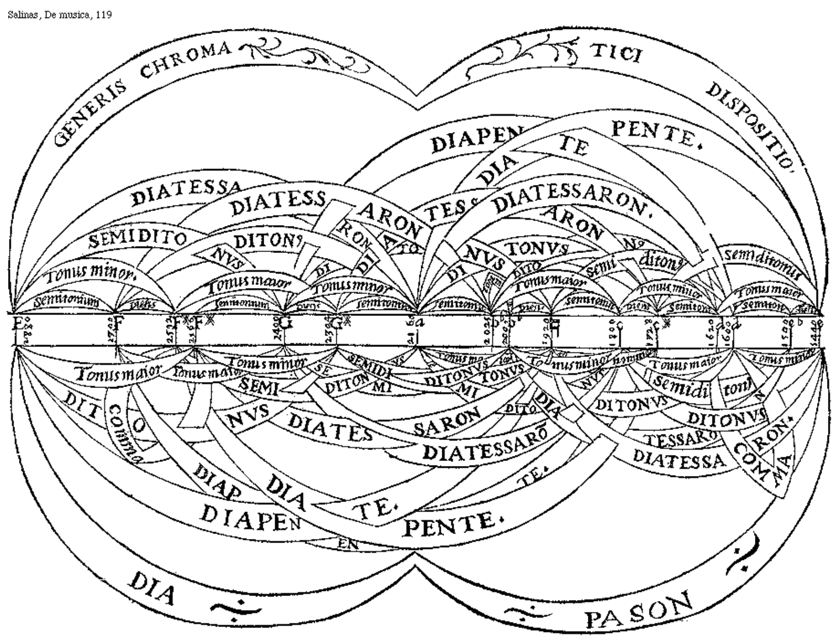


**Figure 2.17:** Historical diagram of a meantone tuning taken from (DE SALINAS, 1577).

## 2.5.1 Arc Diagrams

WATTENBERG (2002) introduces a new visualization technique for the display of complex patterns of repetition in strings. Such patterns are contained in melodies, DNA sequences, compiled code, or just simple text. To visualize repetitive patterns transparent arcs are drawn between pairs of substrings. These pairs are found according to the following specifications: A *maximal matching pair* consists of identical symbols, does not overlap, consists of consecutive substrings, and has maximal length. A *repetition region* is a substring that itself consists of more than one immediate repetition of a substring. The consecutive substrings are called *fundamental substrings*. For the construction of the *Arc Diagram*, *essential matching pairs* are detected in the string. These pairs are either a *maximal matching pair* not contained in any *repetition region* (e. g., 1234 in 1234abc1234), a *maximal matching pair* contained in the same *fundamental substring* of any repetition region that contains it (e. g., 23 in 123123abc123123), or two consecutive *fundamental substrings* in one *repetition region* (e. g., 01 in 010101).

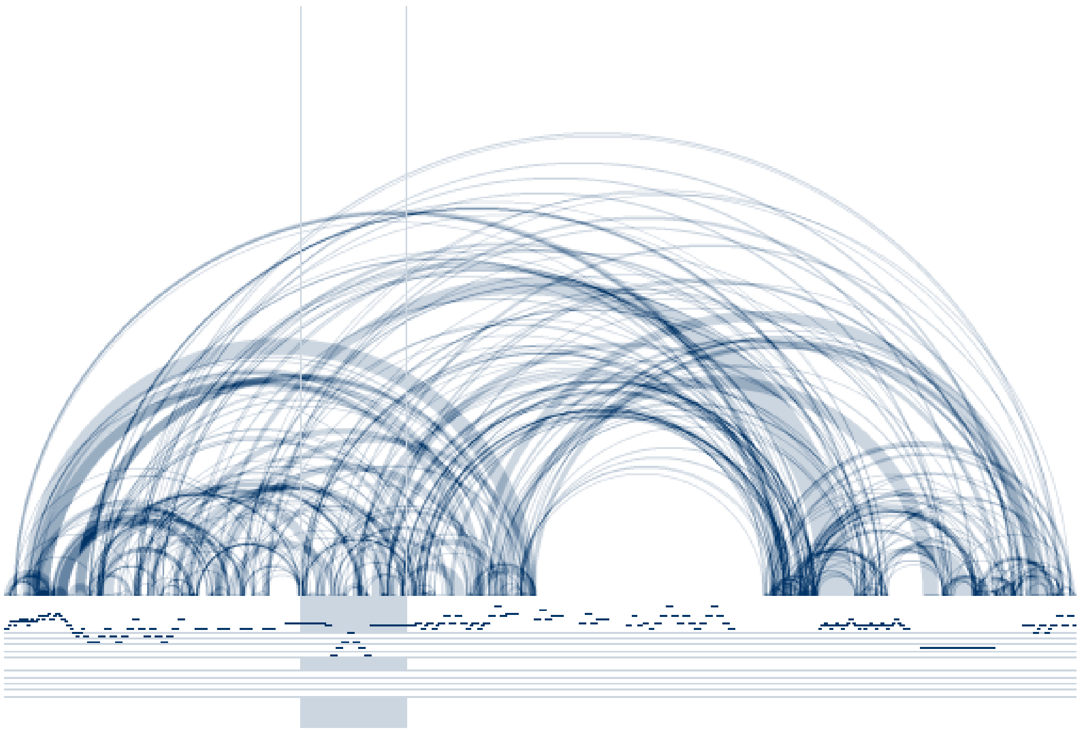
Therefore, the *Arc Diagram* visualizes only a subset of all possible matching pairs with the aim to show only those sequences that are essential to the understanding of the string's structure. The arcs used for visualizing the string pattern are semi-circular and, therefore, correspond in height to the distance between the items of each essential matching pair. Figure 2.18 shows two *Arc Diagrams* visualizing pieces of music. The following visualization technique is inspired by *Arc Diagrams* in the use of similar arcs for displaying relations between e-mail messages in a mailbox.

## 2.5.2 Thread Arcs

*Thread Arcs* were introduced as an interactive technique for visualizing relations in email communication (KERR, 2003). The goal of the visualization is to give users a greater context for the messages in their mailbox and allow for easier performance of actions on groups of messages. *Thread Arcs* were specifically designed to communicate two conflicting attributes of e-mail threads, the arrival sequence of individual messages and their reply-to relationship. The chronology of e-mail messages is encoded in position as a linear order of dots with each dot representing a single message. Reply-to relationships are drawn as arcs. In contrast to the *Arc Diagram* described above, these arcs may also shift below the message line to allow for easier reading. Arcs can also be flattened at the top to save display space. The size of the thread and the number of replies per messages are easily discernible from this visualization as can be seen in Figure 2.19. Interaction techniques allow the user to select and highlight single messages and their answers. Attributes of email messages such as the author(s), date, depth in the thread, etc. are encoded in color.

## 2.5.3 Graph Links on Treemaps

The display of graphs as treemaps was introduced by FEKETE et al. (2003). Using this technique a graph is turned into a general tree with additional relational information. These relations are displayed as Bézier curves on a tree-map for which the basic technique was presented in Section 2.3.2. Although the authors claim to present con-


(a) Bach's Brandenburg Concerto 2.

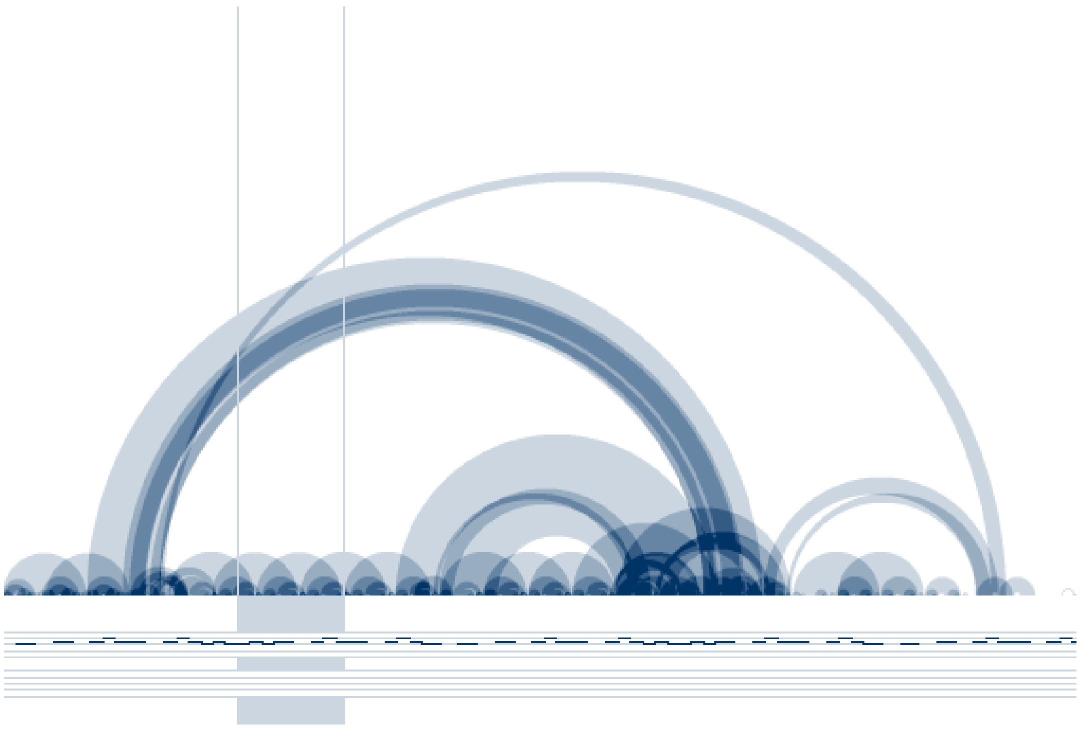
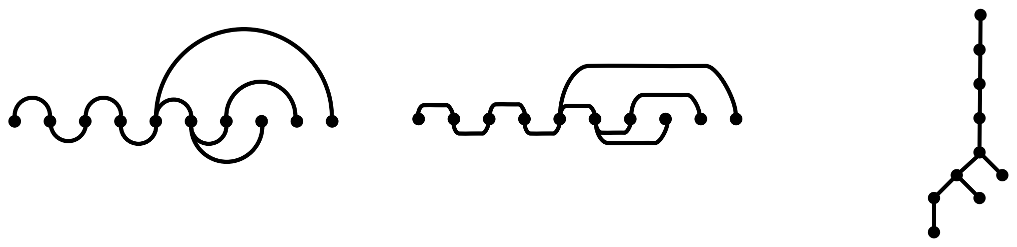

(b) Part from Ravel's Bolero.

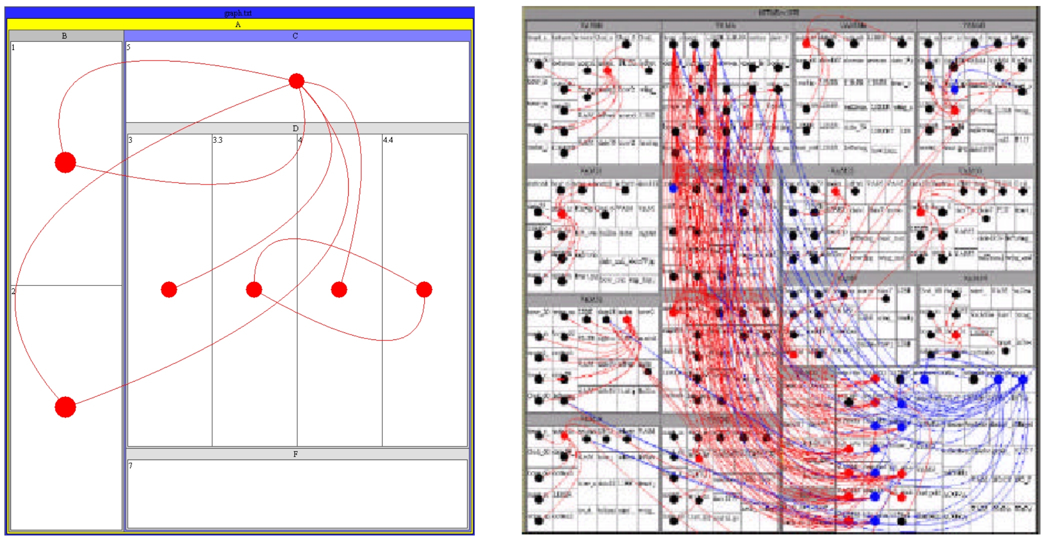
**Figure 2.18:** Arc Diagrams. The images were taken from (WATTENBERG, 2001).



(a) Thread Arc. (b) Thread Arc wit flat top. (c) Traditional diagram as comparison.

**Figure 2.19:** Thread Arcs.

vincing visualization results the presented images and the online demo[5] do not show a clear picture. The initial approach of the visualization was to draw straight lines as edges but the idea was discarded due to too much visual clutter. However, even with curved edges the result is not convincing as can be seen in Figure 2.20. This is due to the static layout of nodes in the treemap. Basically, the technique is a general layout technique for graphs not taking into account one of the most important aesthetic criterion of minimal edge-crossings (as stated by PURCHASE (1998)).



(a) Picture of the online demo with a small tree and just a few links (b) Picture presented by FEKETE et al. (2003).

**Figure 2.20:** Links overlaid on Treemaps.

5 Available at `http://www.cs.umd.edu/hcil/treemap/`.

## 2.6 Summary

This chapter gave an overview of the research field that forms the framework for this thesis. The field of visualization was introduced with a particular emphasis on information visualization. Specific visualization techniques were discussed that will be relevant for the development of the visualization technique presented in the following chapter. A further overview of previous research in information visualization introduced the reader to visualization tools for linear and hierarchical data and graph drawing. The last section discussed visualization techniques that were specifically developed to emphasize relations in data. The following chapter will now explain the design of a new visualization for relations in hierarchical data with a specific emphasis on a design for ordered tree structures.

CHAPTER 3

# Visualization of Relations in Hierarchical Data

Visualizing relations is one of the most important tasks in information visualization as has been explained in Section 2.5. The display of relations with direct links or edges has proven to be a strong encoding technique for relational information as will be further explained in this chapter. However, the linked display of relational information in hierarchical structures poses many problems, the most severe being edge-node and edge-edge intersections. An attempt to solve the display of direct links on tree structures will be presented in this section. The first section will introduce the particular problems that should be solved with the developed visualization. Section 3.2 introduces general requirements for the creation of a meaningful information visualization. These requirements will be obeyed in the creation of the visualization. The underlying data structures will be defined in Section 3.3 as a prerequisite for the description of the proposed visualization in the last section.

## 3.1 Problem Analysis

According to CARD et al. (1999) there are two purposes for a visualization. The first purpose is to communicate an idea requiring the idea to be already present and the second is to create or discover an idea. The visualization presented in this thesis serves the second purpose. A given hierarchical data set is presented graphically to the user to help him or her gain new insight and form a decision.

A prerequisite for the visualization constitutes the hierarchical structure of the underlying data set. To account for the user's mental model of the data the display should be generated in such a way that the hierarchical structure is pointed out. As an enrichment to the data external relational information on connections or relations between data items are available. In general, the detection of patterns in data is enhanced through organizing data visually according to relations (Card et al., 1999). However, in this data set two different forms of relations are present: one inherent and one external to the data. Finding a visualization that aids pattern detection for both types of relations will be one of the main challenges.

Another difficult problem in building a meaningful information visualization is the effective use of *screen real estate*. This term describes the amount of space available on a display for an application to provide visual output. A visualization usually includes a large dataset and a number of visible controls to be displayed. The goal is to minimize the use of hidden commands or scrolling which prevent a good overview and, therefore, make it more difficult to gain a better understanding of the data. On the other hand, a cluttered display is to be avoided. An appropriate use of whitespace should be achieved to prevent information overload. The visualization in this thesis is developed as a support tool for the user of application software such as a reading environment, text editor, or calendar. Hence, the visualization has to share screen real estate with an application that is at the focus of the user's attention and should require as little screen real estate as possible.

The development of a certain visualization depends to a great extent on the tasks that need to be supported. The possible tasks are as widely spread as the types of data to be displayed. Typical tasks include *object generation* (e. g., the generation of images, text documents, 3D models, etc.), *exploration* of the data set with the objective of becoming acquainted with the data, *data diagnostics* with the aim of finding patterns or errors in the data, or specialized tasks such as *navigation* (i. e., in geographical information systems). Providing interaction mechanisms for many possible tasks is another challenge in visualization. The user of the interactive visualization presented here will have different tasks at hand depending on the data set provided. Potential tasks on different data sets and their support will be described in Chapter 5.1. In general, this visualization tool is built to support pattern detection for external relational information in hierarchical data structures.

## 3.2 Requirements for a Visualization

One of the problems of information visualization is its inability to formulate precise criteria for the effectiveness of graphical representations. Such a formulation is difficult since the effectiveness depends largely on conventions and the viewer's perceptual capabilities. Figure 3.1 presents a ranking of encoding techniques according to three types of data to be displayed (Mackinlay, 1986).

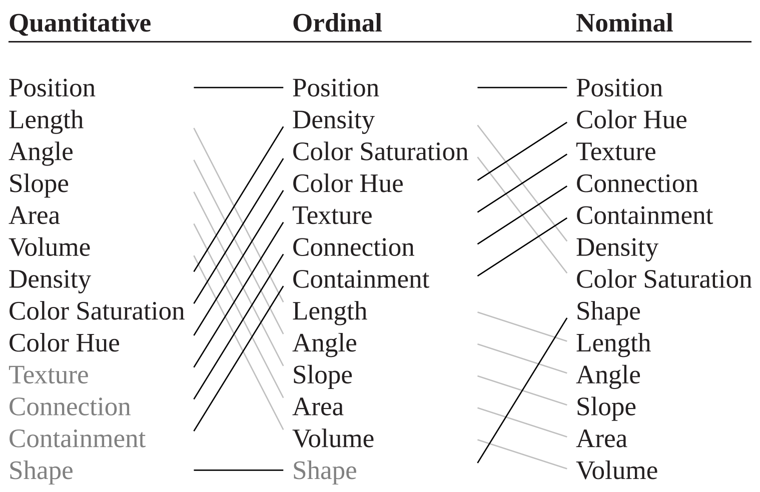


**Figure 3.1:** Efficiency of encoding techniques in descending order. The techniques are compared according to three different data types. Gray techniques are less effective in the given context and, therefore, not relevant for the encoding task. Diagram adapted from (MACKINLAY, 1986).

The chart in Figure 3.1 builds on the psychological observation that people accomplish the interpretation of graphical presentations with different degrees of accuracy depending on the selected encoding techniques (CLEVELAND and MCGILL, 1984). The ordering is, however, only empirically verified in parts. Since, so far, no global empirically verified theory of human perceptual capabilities exists the following general visualization criteria will be given:

**Mapping:** The mapping of data to a visualization has to preserve the data (CARD et al., 1999). Additional data must be excluded from the representation. The mapping should also be easy to interpret and avoid possibilities of interpretation errors.

**Mental Model:** Humans tend to reduce the complexity of the real world by forming its most important concepts into *mental models*. These models guide the user's actions and behaviors. A *conceptual model* is the model upon that an interface or visualization tool designer bases his or her work (NORMAN, 1988). Both models need to match—otherwise human-computer interaction might fail. Also, the mental model which a user might already have of a given data set needs to be supported.

**Overview:** For many tasks involving information exploration, a global view of the information is important. An overview aids navigation and analysis of the data. It provides an initial context and helps the user to form a mental model of the data set. Patterns should be recognizable and outliers and exceptions should be identifiable in an overview of the data. In a next step, the viewer may then formulate a more detailed search, identify patterns, or make a Gestalt overview (JERDING and STASKO, 1998).

**Space Utilization:** Efficient space utilization is important in information visualization. As pointed out in Section 3.1, screen-real-estate is a limited resource. A balance between an appropriate use of whitespace and used screen space has to be achieved. Occlusion of items contributes to errors and disorientation.

**Interaction:** Data should be rearranged and explored interactively. A different display of data can lead to additional insight (SPENCE, 2000). The interaction mechanisms should require low cognitive load on the user. Smooth transitions and no abrupt layout changes are necessary for a user to keep a mental model of the data. The interaction mechanisms should support the possible tasks of the user.

**Affordances:** Affordance is a term that refers to properties of an object which suggest how it might be used. In a visualization, items should tell the users if they can be clicked on, dragged, rotated, etc. Understanding an item's function depends on many factors including conventions (SPENCE, 2000). To make a good interactive interface it must be created with the appropriate affordances to ease the task to be carried out by the user (WARE, 2000).

**Aesthetics:** The elements that contribute to an aesthetic appeal of a visualization may depend on many factors. These include color, texture, symmetry/asymmetry, focal point, contrast, perspective, dimensionality, pattern, unity/Gestalt, and proportion. These factors should be chosen carefully to create an appealing visualization.

**User Studies:** Visualizations should be tested on the intended viewers. Since visual perception and interpretation are led by conventions and personal perception capabilities a visualization tool designer needs to verify his or her tool with user studies.

After problems of the visualization task for this thesis and general visualization criteria have been described, the following section will introduce its specific data and data representations.

## 3.3 Data and Data Representation

As has been already stated in the problem analysis in Section 3.1, one prerequisite for the visualization developed in this thesis is a hierarchical data structure. How this data structure is specified will be the topic of this section.

Hierarchical data structures are usually represented in directed tree structures. KNUTH (1997) describes trees as the most important nonlinear data structure in computer algorithms. In the following, general definitions from graph theory are given:[1]

**Definition 3.1** *A* tree *is a connected graph with no cycles.*

**Definition 3.2** *A* directed tree *is a digraph whose underlying graph is a tree.*

A directed tree, in other words, consists of a number of nodes and parent-child relations with the following characteristics: Every node has just one *parent* and any number of *children*. The number of children of a node is called its *degree*. The starting point of the tree is the *root* node specified by the following definition:

**Definition 3.3** *A* rooted tree *is a directed tree with a distinguished vertex* $r$*, called the* root*, such that for every other vertex* $v$ *there is a directed path from* $r$ *to* $v$*.*

The root node is the only node with no parent. Undirected trees where any node may act as a root node play a role in mathematics but are not as important in computer science and will, therefore, not be discussed further. The connection between parent and child nodes is called an *edge*. *Leaf* nodes are nodes without children. The *height* or *depth* of the tree is the maximum number of levels in a tree, not including the root level. All nodes, excluding the root, partition the tree in disjoint sets of nodes that each form a *subtree*. In this thesis the relative order of child nodes plays an important role, hence, we deal with *ordered trees*. These are defined as follows:

**Definition 3.4** *An* ordered tree *is a rooted tree in which the children of each vertex are assigned a fixed ordering.*

After these general definitions for tree structures haven been introduced, the following discussion will now be concerned with the data structure created for the visualization task of this thesis.

In addition to the inherent parent-child relations in the data, external relations might be specified. Adding external relations to tree nodes might transform a tree structure into a general graph. However, the two sets of relations will be distinguished and the data will still be referred to as having a hierarchical layout. The underlying data structure is known to the user as a tree structure. Therefore, to support the user's mental representation of the data, the visualization will resemble one of the already described representations for trees (cf. Section 2.3). A technique for overlaying a representation of external relations on such a tree structure will be developed in this section. The following definitions for such a data set correspond to conventions presented by HUANG (2001). This *arc tree* data structure is defined as follows:

1 For further definitions regarding *graphs*, *directed graphs*, *connectedness*, or *cycles* please refer to Appendix A.2.

**Definition 3.5** *An* arc tree *is an ordered, rooted tree* $T = (N, E, R)$. *It consists of a finite set* $N$ *of nodes, a finite set* $E$ *of edges, and a finite set* $R$ *of external relations. Incidence functions are defined on the tree:*
$\varphi : E \rightarrow N \times N$*;* $\varphi(e) = (u, v)$ *with* $e \in E$ *and* $u, v \in N$
$\phi : R \rightarrow N \times N$*;* $\phi(r) = (u, v)$ *with* $r \in R$ *and* $u, v \in N$.

Both relations and tree nodes carry additional domain-specific attributes. Nodes might carry user-specified names, IDs, or weights. Relations might be further specified by type parameters, weights, or IDs. The following definition further specifies these conditions:

**Definition 3.6** *An* attributed arc tree $A(T) = ((A(N), A(E), A(R))$ *consists of a finite set* $A(N)$ *of attributed nodes, a finite set* $A(E)$ *of attributed edges, and a finite set of attributed external relations* $A(R)$. *Each attributed node* $a(v) \in A(N)$ *consists of* $(v, DA(v))$ *where* $DA(v)$ *is a set of domain-specific attributes associated with node* $v$ *in the tree. Respectively for the attributed edges and external relations.*

A graphical representation of a node, relation, or edge is called a *glyph*. Each glyph has a set of attributes $A_g$. The set of possible attributes $A_g = \{a_g^1, a_g^2, \ldots, a_g^n\}$ consists of different parameterizations of the graphical variables presented in Section 2.1.2. If $a_g^i$ is the shape variable possible values for $a_g^i$ include *rectangle, circle, or arc*.

A drawing of an attributed arc tree is specified as follows:

**Definition 3.7** *A* drawing *of an* attributed arc tree *consists of three functions:*
*The function* $D_N$*:* $A(N) \rightarrow \mathbb{R}^2$ *assigns a display location, width, and height to each attributed node* $a(v) \in A(N)$.
*The function* $D_E$*:*$A(E) \rightarrow \mathbb{R}^2$ *assigns location, width, and height to each attributed edge, specified by its endpoints.*
*The function* $D_R$*:*$A(R) \rightarrow \mathbb{R}^2$ *assigns location, width, and height to each attributed external relation, specified by its endpoints.*

An attributed visualization $AV(A(T)) = (\text{GLYPH}(A(N)),\text{GLYPH}(A(E)),\text{GLYPH}(A(R)))$ finally consists of a set of glyphs for the attributed nodes, attributed edges, and attributed relations. Each glyph $g(a(v)) \in \text{GLYPH}(A(N))$ consists of $(A_g, D_N(A(v)))$, a set of attributes and a display location, width, and height on the plane. Similarly, a set of glyphs $g(a(r)) \in \text{GLYPH}(A(R))$ consists of $(A_g, D_R(A(r)))$, respectively for $g(a(e))$.

Given these definitions, the challenge will be how to map an attributed tree to an attributed visualization. The proposed solution will be described in the following sections.

## 3.4 Visualization

This section is concerned with creating a visualization from an *arc tree* data structure. An initial data set is described and turned into an *arc tree visualization*. Involved in the design of the visualization are a spatial tree and relation layout. Encodings for attributes of nodes and relations are further discussed followed by the introduction of a labeling scheme.

### 3.4.1 Initial Data Set

The initial data set given as input to the visualization tool can be derived, for example, from an XML-file description. An XML parser generates a hierarchically structured arc tree $T = (N, E, R)$. The sets $N$ and $E$ are filled automatically while parsing, whereas $R$ initially remains empty. This is straightforward since a well-formed XML document has an inherent tree structure. The created tree is later handed off to a renderer for generating a visual representation of the tree. The structure of the tree, therefore, corresponds to the structure of the XML document. The built tree structure $T$ is an ordered tree. Parsed XML elements form the nodes of the tree. Tree attributes can be read or created by the parser to form an attributed arc tree $A(T) = ((A(N), A(E), A(R))$. The set $R$ and its attributes $A(R)$ can be created from XML file specifications, read from an external file, or be defined by properties of the nodes to be connected. The following domain specific attributes are later used in the visualization process:

- $DA(N) = \{id, weight, level, degree, nrExternalRelations, type, doi, title\}$.
- $DA(E) = \{\}$.
- $DA(R) = \{id, type, weight\}$.

### 3.4.2 Spatial Tree Layout

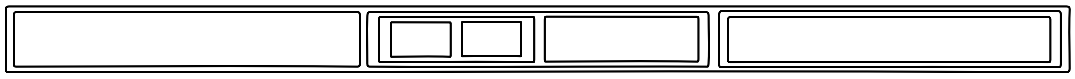
An essential part of the *arc tree visualization* is the spatial tree layout. The layout takes the ordered structure of the XML file into account. A sequential order of leaf nodes is usually of vital interest to the user. In the case of book data, for example, the nodes have to be arranged in linear order to support the mental representation of the book's structure in the user's mind. The visualization tool, therefore, will regard two constraints in creating a tree layout:

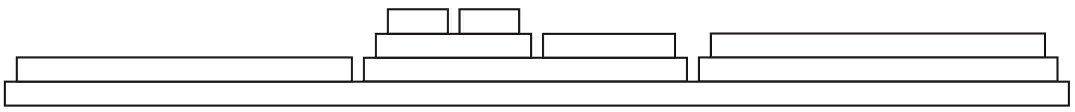
1. Consideration of a layout for ordered trees.

2. Spatial constraints as stated in Section 3.1: minimization of screen real estate and appropriate use of whitespace.

During the spatial tree layout, a *geometric mapping* reserves a spatial location, width, and height for later display of node glyphs. During this process, domain specific attributes of the nodes such as weight or level information are used to accumulate space. In this space glyphs can be displayed once their final shape has been determined.

Generally, there are two main options for tree layout: *connection* (also called *explicit tree visualization*) as in node-link diagrams or *containment* (also called *implicit tree visualization*) as in space-filling approaches. Node-link diagrams are superior in revealing a tree's structure but can easily acquire large aspect ratios through the inclusion of a large amount of whitespace. Space-filling techniques, in contrast, shine when users are interested in leaf nodes but are not as good at displaying the tree's structure. However, they typically require less screen space. Due to the given constraints a space-filling approach was selected. The created layout resembles in its structure a linear Tree-Map approach as described by TURO and JOHNSON (1992). The tree layout algorithm partitions a rectangular display space for the root into $n$ parts with $n$ being the number of the root's children. For each child, the display space is then again divided vertically according to the number of children at the next level in the tree. This algorithm was selected to establish an ordered tree layout and to provide a spatially constrained display environment. It also conforms to traditional tree layouts that flow from one side of the display to the other. The visual metaphor (cf. Figure 3.2(b)) behind the chosen layout is a stack of nodes similar to an upside-down icicle plot as presented in Appendix A.1. This stack of nodes is viewed from the top with the result presented in Figure 3.2.


(a) The proposed space-filling tree layout for ordered trees.


(b) The tree layout metaphor from which the proposed tree layout is derived resembles a stack of nodes.

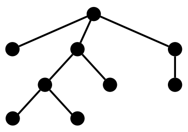

(c) Node-link diagram.

**Figure 3.2:** The proposed tree layout and its node-link counterpart.

Since this tree visualization technique uses a limited amount of display space it can easily be used as a modular component of information displays. Therefore, the following requirements for a visualization, as presented in Section 3.1, are met with the proposed layout:

1. The hierarchical structure is pointed out to the user through a space-filling approach. A layout for ordered trees is supported.
2. Use of screen real estate is limited. The visualization can be used as a modular component of information displays.

For displaying all nodes with an equally spaced offset around each node the following layout algorithm was developed.

### Tree Layout Algorithm

A tree layout algorithm specifies the drawing $D_N{:}A(N) \rightarrow \mathbb{R}^2$ and calculates position, height, and width for the nodes in $N$. The following domain specific attributes are used in creating the layout: $DA_{D_N}(N) = \{weight, level, degree\} \subset DA(N)$. The following factors are respectively defined:

*width:*
The width of the display area,
*height:*
The height of the display area,
*totalWeight:*
The sum of all weights of a node's children,
*offset:*
An equal amount of space around a node that separates it from its parent or possible siblings,
*offset allocation factor (oaf):*
The number of times an offset size needs to be considered in the display of a node,
*maximal offset size (mos):*
The maximal possible offset size for this tree so that all nodes can be displayed,[2] and
*space allocation factor (saf):*
The amount of space, relative to the display area, reserved for this node and its offsets.

Figure 3.3 gives an overview of some of these factors. In general, the algorithm recursively specifies how many offsets have to be considered in the display of a node and how much space is available for the display of a node and its offsets with respect to its ancestors. The *offset allocation factor* for Node B, for example, is calculated by adding

2 The actual offset size should be less than the *mos*.

Offset 1 and $\frac{1}{2}$·Offset 2. Therefore, $\frac{3}{2}$ offsets have to be taken into account in calculating the display size for Node B. Similarly, for Node D Offsets 3, 4, 5 and $\frac{1}{2}$·Offset 2 are added to calculate an *offset allocation factor* of $\frac{7}{2}$. The size that is available for the display of a node including its offsets is determined by the *size allocation factor*. If no offset size would be specified the display space of Nodes B,C, and D would be half of the root's display size leading to a *size allocation factor* of $\frac{1}{2}$. The *maximal offset size* at Level 1 (for Node B and Node C) is $\frac{1}{3}$ of the size of the root node. If this offset size is chosen, however, the display size for all nodes on Level 1 is zero. Therefore, the actual offset size should always be a fraction of the maximal offset size. Depending on the chosen proportion smaller or larger offsets can be created.

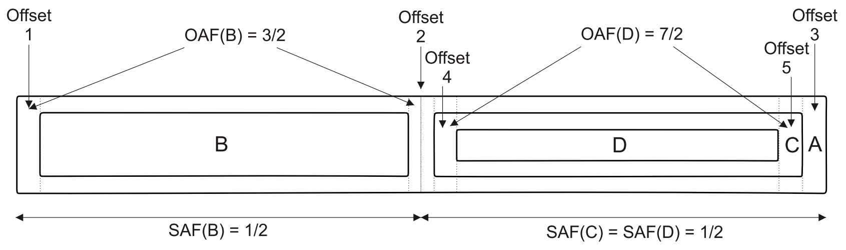


**Figure 3.3:** Factors in calculating an offset for nodes in the spatial tree layout.

The code in Listing 3.1 contains the algorithm for calculating offset sizes. The function *calculateOffsetSize(Node n)* calculates the *mos* for all nodes in the horizontal direction first. In the second function *checkBorderSize()* the calculated *mos* is compared against the maximal possible offset size in $y$-direction which is dependent on the number of levels in the tree.

Afterwards, the node sizes and positions are calculated by setting an initial width and height for the display area and recursively calculating sizes for the nodes. Chosen offset size and size allocation factors are taken into account to calculate the final display space available for each node. As can be seen in the proposed algorithm, the weight of a node plays an important role in the calculation of the size allocation factor for a node and, therefore, directly influences how much space will be available for its display. How these weights can be specified will be the topic of the following section.

### Tree Node Metrics

How much space will be reserved for the display of a node can be specified according to certain metrics that are applied to the nodes in the tree. A *node metric* is used to measure or quantify certain abstract features associated with a node to allow for comparison or ranking of nodes. Node metrics can be of *structural type* representing structural information about the tree or *content based* representing domain specific

```
calculateOffsetSize(Node n){

 parent = n.getParent()
 mos = width
 if (parent == null){

   n.setOAF(0.0) //the root has no borders
   n.setSAF(1.0) //the root takes up all the space
 }
 else{

   oaf = (parent.getOAF() + parent.getDegree() + 1) / parent.getDegree()
   saf = (node.getWeight() / parent.getTotalWeights()) * parent.getSAF()
   n.setOAF(oaf)
   n.setSAF(saf)
   if (n.isLeaf()){

     possibleOffsetSize = (saf * width) / oaf
     if (possibleOffsetSize < mos) mos = possibleOffsetSize
   }
   for (each child of n) calculateOffsetSize(child of n)
}

checkOffsetSize(){

   //we check if the offset is small enough to hold all levels
   yOffset = height / (tree.getDepth()*2)
   if (yOffset < mos) mos = yOffset
}
```

**Listing 3.1:** Calculating node and offset sizes.

attributes about the node itself. A node metric algorithm changes the domain specific weight attribute of a node as given in

$DA(N) = \{id, weight, level, degree, nrExternalRelations, type, doi, title\}.$

In the previously described tree layout algorithm the node's weight attribute is used to calculate the display size and position for each node. Two different structural metrics have been conceived to alter tree layouts. These involve the following node characteristics:

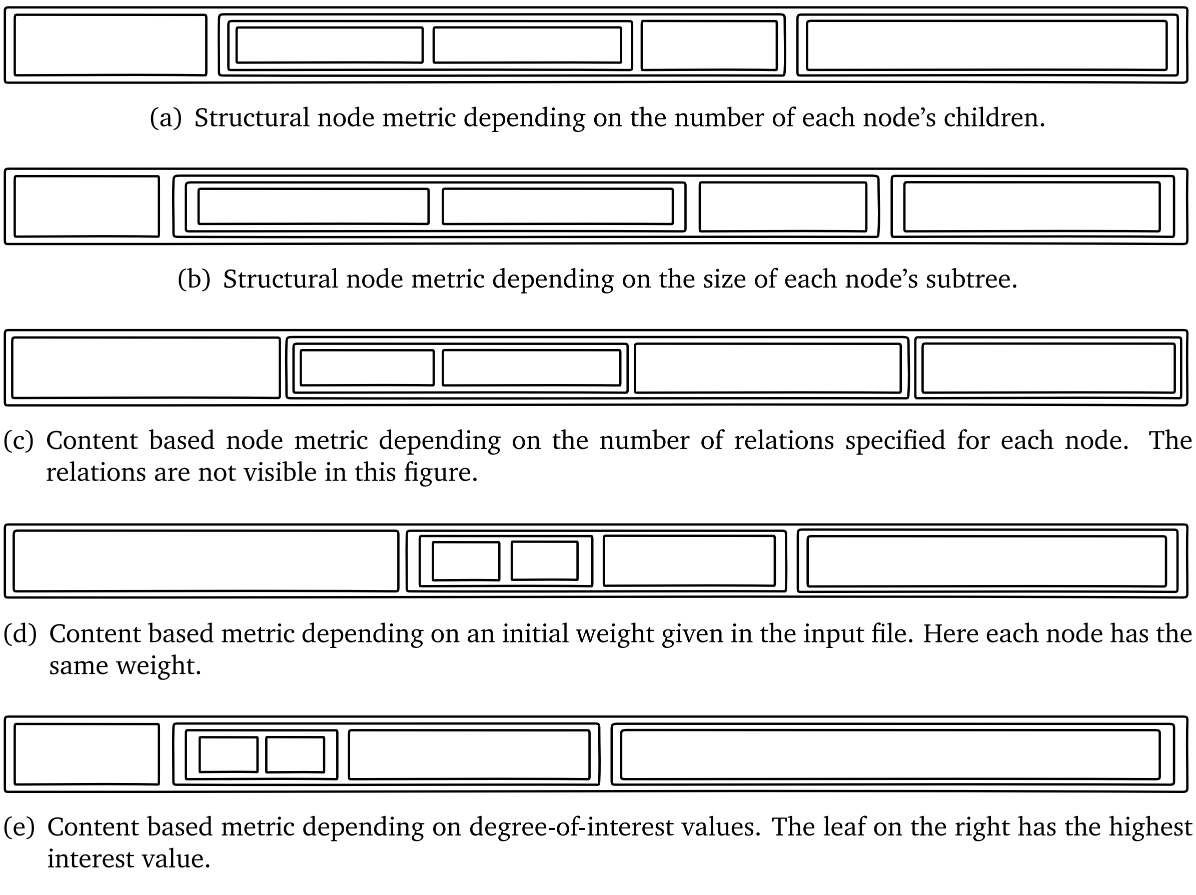
1. *Degree* of a node meaning the number of the node's children. A node with more children will occupy more space than a node with less or no children.

2. *Size* of the subtree below the node in the tree. A node with many descendants requires more display space than nodes with less descendants.

In addition, three content based metrics have been conceived. These involve the following node attributes:

1. *Number of external relations* ending or starting at each node. Nodes will be displayed larger if they are connected by more relations.
2. *Initial weights* given in the XML file description. This metric gives the user the freedom to initially specify weights for nodes or certain types of nodes.
3. *Degree-of-interest* (doi) values specified by the user interactively. This metric will be described in Chapter 4.

A visual overview of the conceived tree metrics is given in Figure 3.4 based on a small example tree of just 9 nodes.


(a) Structural node metric depending on the number of each node's children.

(b) Structural node metric depending on the size of each node's subtree.

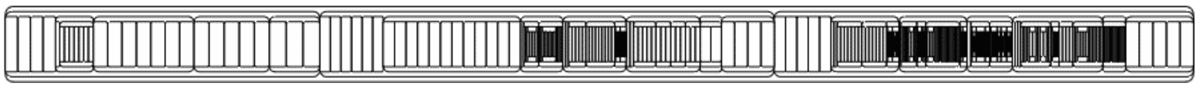
(c) Content based node metric depending on the number of relations specified for each node. The relations are not visible in this figure.

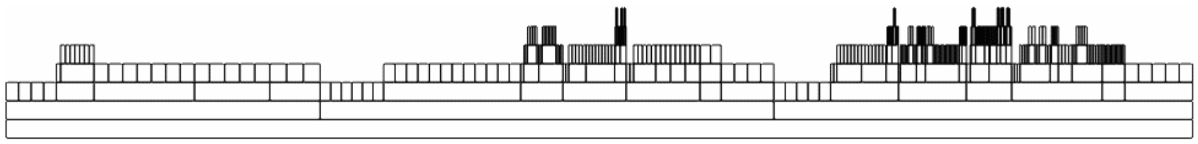
(d) Content based metric depending on an initial weight given in the input file. Here each node has the same weight.

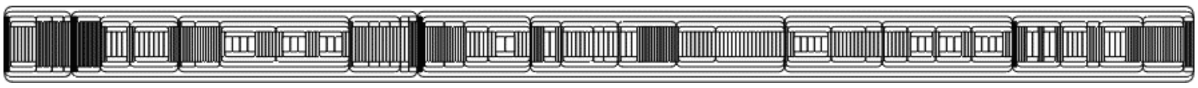
(e) Content based metric depending on degree-of-interest values. The leaf on the right has the highest interest value.

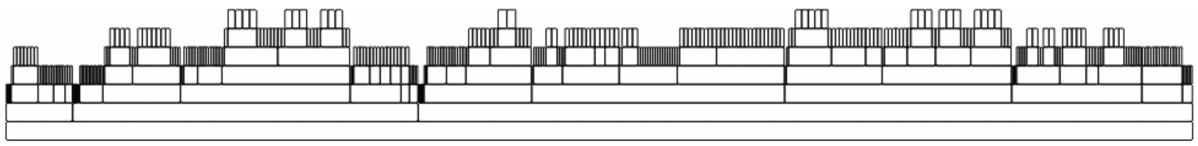
**Figure 3.4:** Tree metrics implemented for the tree layout.

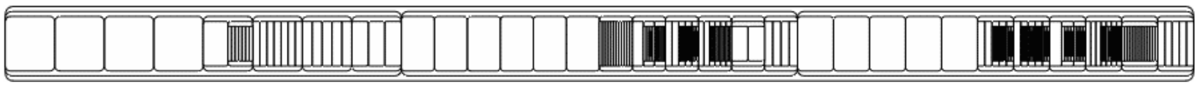
How significant the choice of a node metric is can be seen in Figure 3.5 where a larger tree with 419 nodes and a depth of 6 is displayed using the different node metrics.

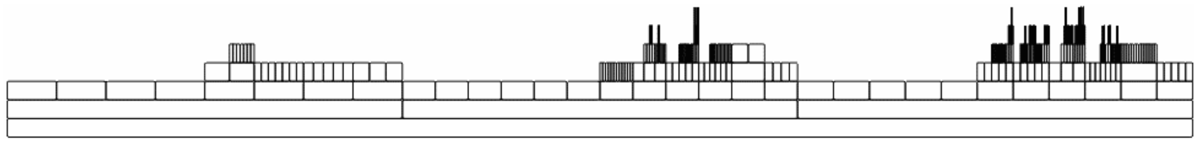

(a) Structural node metric depending on the number of each node’s children.

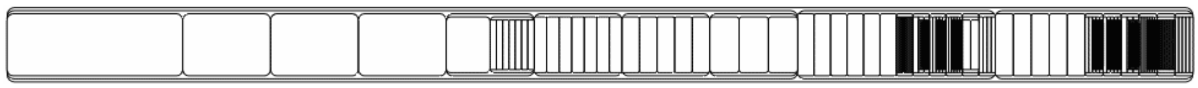

(b) The same metric applied to a side view of the tree.


(c) Structural node metric depending on the size of each node’s subtree.


(d) The same metric applied to a side view of the tree.


(e) Content based metric - each node has the same weight.


(f) The same metric applied to a side view of the tree.


(g) Content based metric - nodes are assigned weights according to interest values.

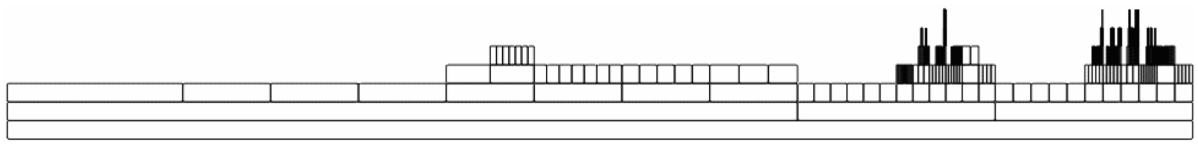

(h) The same metric applied to a side view of the tree.

**Figure 3.5:** Tree metrics implemented for the tree layout on a tree with 419 nodes and depth 6.

### Evaluation

The proposed layout has the following advantages: all nodes are laid out in a sequential order, leaf nodes are easily distinguishable from internal nodes, and the space for the tree can be constrained in length and height. However, only a given number of leaf nodes can be displayed sequentially, constrained by the horizontal display resolution. Therefore, focus+context techniques will be introduced in Chapter 4. One disadvantage of the proposed tree layout is that it can hardly be used to compare node weights according to the node sizes since the size of a node is not proportional to the node's weight. The display space for a node is calculated in relation to the sizes of its ancestors and the weights of its siblings. Figure 3.6 shows a layout where each node's weight has the same value. Node $A$ in Level 2 has the same weight as Node $B$ in Level 1 but both nodes were assigned different sizes. Node weights can be compared easily only according to the size of a node's siblings (larger siblings will have a larger weight). If it is important for a task that the node weights are easily comparable a node-link tree layout should be chosen.

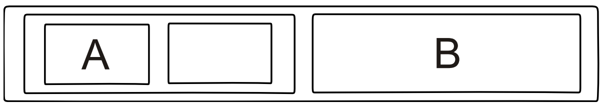


**Figure 3.6:** Node $A$ and Node $B$ have the same weight but different size. Node weights cannot be easily compared according to node sizes in this tree layout.

One other aspect of a space-filling tree layout is that the drawing $D_E : A(E) \to \mathbb{R}^2$ does not specify a position or size for the edges. The edges are not drawn. Structural relations are defined by the nested layout. Many other types of relations, however, can be incorporated in this layout. Therefore, specifications for $D_R$ are discussed in the following.

## 3.4.3 Spatial Relation Layout

In this thesis external relations $R$ as given in an arc tree $T = (N, E, R)$ are encoded using direct links. Since the spatial tree layout does not use edges to display structural relations between nodes this type of encoding can be used to display other types of relations without the danger of confusion. As has been mentioned in Section 2.1.2, the Gestalt principle of continuity assumes connectedness. If two objects are connected by a line, this visual connection can create a link that represents any number of relations (Ware, 2000). It has been shown by Palmer and Rock (1994) that connectedness can be a more powerful grouping principle than proximity, color, size, or shape. Therefore, direct connections will be introduced as an encoding technique for the relations in $A(R)$ to emphasize the importance of the additional relational data.

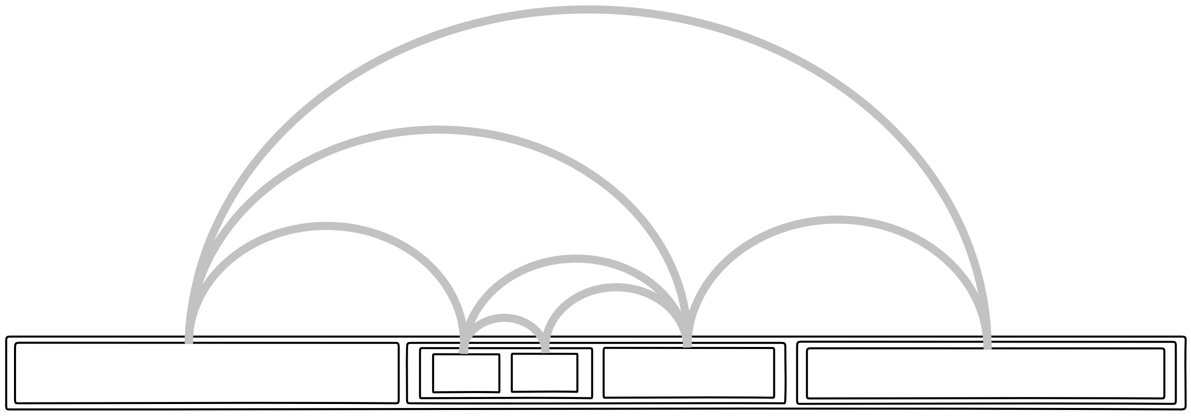
As soon as additional edges are introduced on tree layouts, the drawing and spatial layout of these links becomes a graph drawing task. The following drawing constraints are considered in creating a geometric mapping for $A(R)$:

1. *Static Node Layout:* The sequential ordering of tree nodes cannot be changed.
2. *Intersections:* A minimal number of edge-edge and edge-node crossings should be achieved.
3. *Area Bounds:* The area of the whole drawing should be constrained and fit in a given display space.
4. *Total Bends:* A minimization of the total number of bends along an edge should be achieved.
5. *Angle of Incidence:* Avoid small angles of incidence at nodes to allow for easier comparison of edges.

In order to achieve minimal edge-node crossings and to minimize the use of screen real estate all edges are drawn above the tree display. The encoding for relations is inspired by the works on Arc Diagrams (WATTENBERG, 2002) and Thread Arcs (KERR, 2003) that were both introduced in Section 2.5. To visually represent the relations arc shapes are created. These arcs are drawn directly from the source node of a relation to the destination node. Arcs are advantageous in minimizing the number of total bends (with only one bend per arc) and creating a symmetric appeal. With a cleverly placed arc, edge crossings can be avoided. The drawing function $D_R : A(R) \to \mathbb{R}^2$ positions the endpoints of the arc in $x$-direction according to the center points of the nodes to be connected. The arc height has been chosen to be in direct proportion to the distance between the connected nodes. A prototypical relational layout can be found in Figure 3.7. As can be seen, edge crossings are avoided with this layout as long as there are no relations from a node that lies within an already connected pair of nodes to a node that lies outside of this pair. In order to minimize the needed screen real estate a modular refinement of the arcs was needed which will be described in the following.

## Arc Size

The initial approach for the layout of the arcs uses a semi-circular design as seen in Figure 3.8(a). In this approach, the arc's radius corresponds to the half distance between the two nodes to be connected. This layout has a symmetric appeal and node distances can be easily derived from comparing arc heights. However, in the worst case of the first and last node being connected the display area would need to be at least half in height as in width in order for all connections to be fully visible. Therefore, an adaptable arc shape was needed to support smaller arc heights. Chaikin's corner cutting


(a) Relation layout using semi-circles as node shapes.

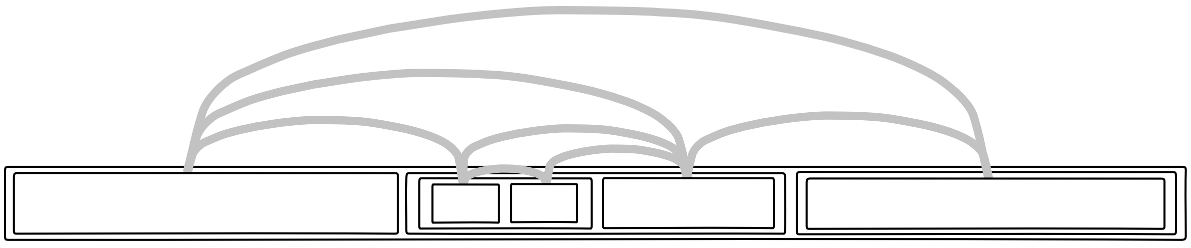
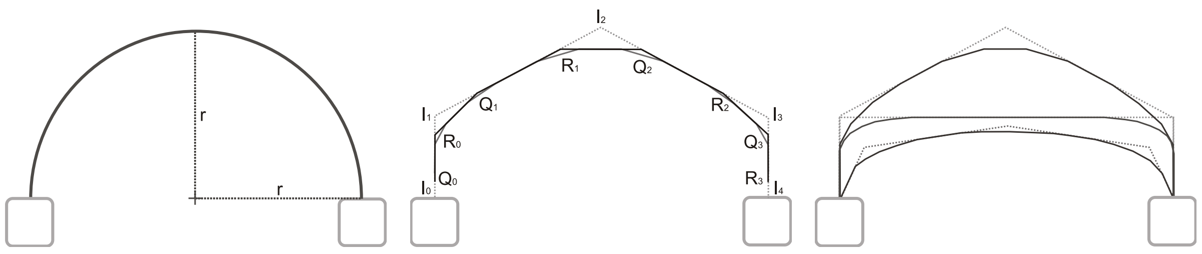

(b) Relation layout created using Chaikin curves.

**Figure 3.7:** Relation set R encoded through direct connection using two different arc shapes.

algorithm provides the functionality needed to produce arcs with a high level of modularity (CHAIKIN, 1974). A corner cutting algorithm creates smooth curves through a recursive refinement process. In this process intermediate points are calculated which lie on the lines joining the points of a control polygon. A new control polygon for the next refinement step is easily created by cutting the corners off the previous one. Mathematically, an initial control polygon given by a set of points $\{I_0, I_1, \dots, I_n\}$ is refined through the creation of a new set of points $\{Q_0, R_0, Q_1, R_1, \dots, Q_{n-1}, R_{n-1}\}$. Each new pair $(Q_k, R_k)$ is created by setting $Q\frac{1}{4}$ and $R\frac{3}{4}$ of the distance along the line connecting $I_k$ and $I_{k+1}$. Therefore,

$$Q_k = \frac{3}{4}I_k + \frac{1}{4}I_{k+1}, R_k = \frac{1}{4}I_k + \frac{3}{4}I_{k+1}$$

It has been shown that this algorithm produces results equivalent to a quadratic uniform B-spline curve (RIESENFELD, 1975). Figure 3.7 gives a comparison between the semi-circular relation layout and one of the many possible arc shapes created using Chaikin's algorithm. Figure 3.8(b) gives an example for a curve with one annotated refinement step.



(a) Circular arc shape.

(b) Interpolation with Chaikin's algorithm.

(c) Different arc shapes produced from varied control polygons using Chaikin's algorithm.

**Figure 3.8:** Proposed arc shapes.

## Evaluation

The advantage of the proposed tree layout for displaying direct links lies in a minimization of edge-edge and edge-node intersections. If either the $x$- or $y$-axis had been chosen to encode level information on the tree edge-node intersections would have been much more severe. This is especially true once relations are introduced that connect to branch nodes as can be seen in Figure 3.9.

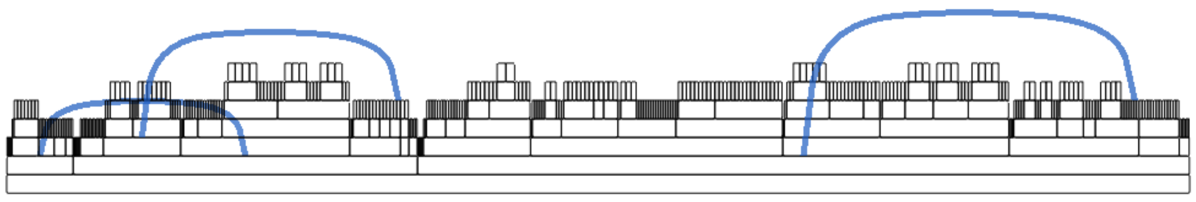


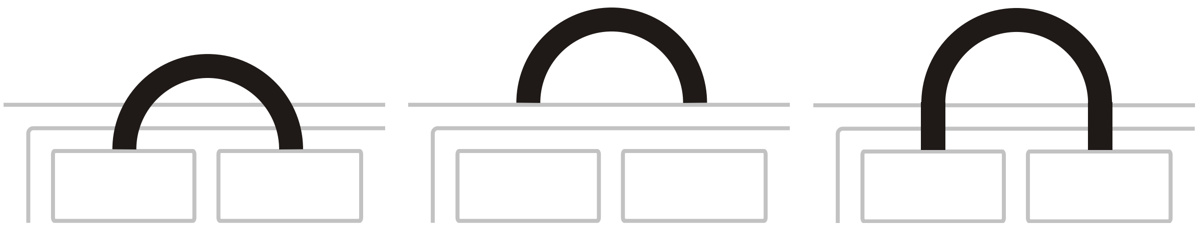
**Figure 3.9:** Edge-node intersections are more severe if level information is encoded using position.

To alleviate the problem of edge-node intersections in the proposed tree layout three methods are introduced to connect nodes and edges (cf. Figure 3.10).

**Direct Connection:** Relations are directly connected to the nodes. Using this method minimal edge-node intersection are introduced where the edges cross lower level offset space to connect to a node. More severe edge-node intersections may appear if the height of an arc is small so that the whole arc is drawn on top of the tree layout. This edge-node connection has been applied in Figure 3.7(b) where its problems can also be observed at the smallest arc.

**On-Top:** Relations are drawn directly above the tree layout without a direct connection to the node. The start- and end-points of the edge are placed directly above the center of its reference nodes. The connection has to be made by comparing underlying node centers with the endpoints of an arc. This intersection type has the advantage that no part of the arc occludes any part of the tree layout.

**Top-to-Node:** This intersection type combines the previous two. Arcs are drawn directly above the tree layout but the endpoints extend down to the referenced nodes as straight lines. The advantage lies in a minimal occlusion of node parts and in a direct connection to referenced nodes.



(a) Direct edge-node connection.
(b) On-Top egde-node connection.
(c) Top-to-Node edge-node connection.

**Figure 3.10:** Proposed edge-node connections to alleviate edge-node intersections.

This section introduced methods for the spatial layout of tree nodes and relations. Both are part of a *geometric mapping* in which geometric attributes of the data such as display size and position are specified. Based on this, the following sections discuss how further node and relation attributes can be encoded in a graphical mapping process.

## 3.4.4 Visualization of Tree Nodes

During *graphical mapping*, attributes of data items are visually encoded to create an adequate representation. A glyph is selected as a visual representation and for each attributed node $a(v)$ the set $DA(v)$ (respectively for edges and relations) is mapped to the set $A_g$ for the respective glyph $g(a(v))$. Geometric and graphical mapping are part of the *visual mapping* process in the visualization pipeline as described in Section 2.1.2. This section is concerned with the following graphical mapping:

$$(DA(N) = \{id, weight, level, degree, nrExternalRelations, type, doi, title\}) \to A_{g(N)}.$$

The following graphical attributes are used for visually encoding domain specific attributes of nodes and external relations in the tree:

$$A_{g(N)} = \{shape, color, 3Ddepthcues, size, position\}.$$

The first important choice in creating an encoding is the type of shape associated with a node.

### Shape Selection

During geometric mapping, rectangular space was reserved for the display of tree nodes which can be seen as a bounding box for the actual shape to be displayed. For the *arc tree visualization* it is important that the shape for each node contains enough surface area to allow for further layering of tree nodes. It should be avoided that higher level nodes become unnecessarily small through the wrong choice of a node shape. In addition, the Gestalt law of proximity states that objects that are placed close to each other are perceived as a unit. Therefore, a rounded rectangle was chosen as the shape to encode tree nodes. Rounded rectangles are advantageous in providing a large surface for further layering and in offering a strong visual proximity through their straight contour. In addition, a rounded rectangle was chosen over a regular rectangle since it provides a small but distinguishable space between two shapes if the border between the two becomes relatively small as can be seen in Figure 3.11. This small additional space is important since in the interaction process objects will have to be selected to perform transformations on the display of data. Another advantage of the rounded rectangle is its aesthetic appeal. Therefore, in this step the *shape* of a rounded rectangle is chosen for each glyph of a node with a given *id*.

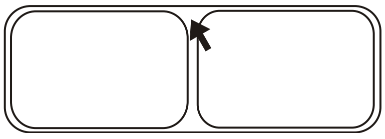


**Figure 3.11:** Shape selected for encoding nodes. The arrow indicates additional space available for selecting the underlying node in an interaction process.

### Encoding of Size and Position

In the *arc tree visualization* size and position are directly adopted from the values calculated during the geometric mapping. In other words, a rounded rectangle fills the whole space reserved for its display and is centered at its calculated position. Therefore, *weight* and *degree* of a node are encoded in size and position of its glyph as calculated by the tree layout algorithm.

### Color Coding

According to (WARE, 2000), the most important role for color in visualization is coding of information. Colors are a useful coding technique since they are perceived as attributes of objects. In the context of the *arc tree visualization* two types of color codes are used: colors for coding nominal information and color sequences.

Nominal color coding describes a technique for classifying objects through the use of color. For the use of color as a classification factor WARE has collected a number of perceptual factors to be considered (WARE, 2000):

**Distinctness:** Colors should be perceived as being as distinct as possible. Visible distinctness can be approximated by comparing distances in a uniform color space like the CIE-LAB Color Space.

**Unique Hues:** A set of hues, namely *red, green, yellow, blue* as well as *black* and *white* have been found to be special in building opponent pairs of color. In addition, cross cultural studies have shown that these colors have names which are quite similar throughout different languages (BERLIN and KAY, 1969). Therefore, these colors provide a good choice in coding a small number of item types.

**Contrast with Background:** In general, color-coded objects can appear on a number of different backgrounds. Color contrast can alter color appearance considerably by making one color look like another. To reduce contrast effects, a small black border can be drawn around objects. In addition, there should be a luminance difference between object and background color to ensure color discrimination.

**Color Blindness:** If only few colors are needed the yellow-blue color direction should be preferred over the red-green direction. The reason for this is that the majority of color-blind people is insensitive to changes in the red-green direction.

**Number:** HEALEY (1996) suggests that only five to ten colors can be rapidly distinguished in nominal color coding.

**Field Size:** Objects that are color coded should not be too small for its colors to be perceived. Small objects should have strong, highly saturated colors while larger objects should have low saturation for allowing a good contrast of smaller objects on such a background.

**Conventions:** Color-coding conventions (e.g., red = danger) sometimes need to be taken into account. However, such conventions are not independent of culture.

Therefore, WARE (2000) suggests using the following colors for nominal color coding:

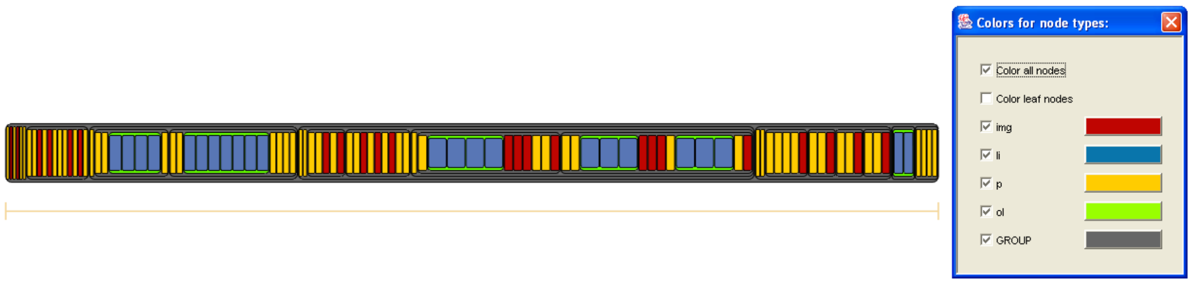
*red, green, yellow, blue, black, white, pink, cyan, gray, orange, brown,* and *purple*.

In the *arc tree visualization* nodes can be colored according to *type* as a domain specific attribute. Since this encoding is nominal, MACKINLAY (1986) suggests using color hue over color saturation (cf. Figure 3.1). The user is given the following options in adapting node colors for separate node types:

- For each unique node type a hue is selected from the suggested colors presented above. If more node types are available, node colors are chosen randomly. The user can also set individual colors for node types of interest. This freedom allows users with a type of color blindness to choose colors that can be distinguished more easily. In addition, the user can adjust colors to follow certain conventions or to avoid color contrast between nested node types.

- To avoid a cluttered interface in cases when too many different node types are present the user can choose to which node type a color should be applied. Other types remain uncolored or are set to a certain default color such as white or gray.
- Type colors can be applied to leaf nodes only or to all nodes of the tree. This also avoids a cluttered display.
- Borders can be drawn around tree nodes to make similarly colored parent-child pairs or siblings easier to distinguish.

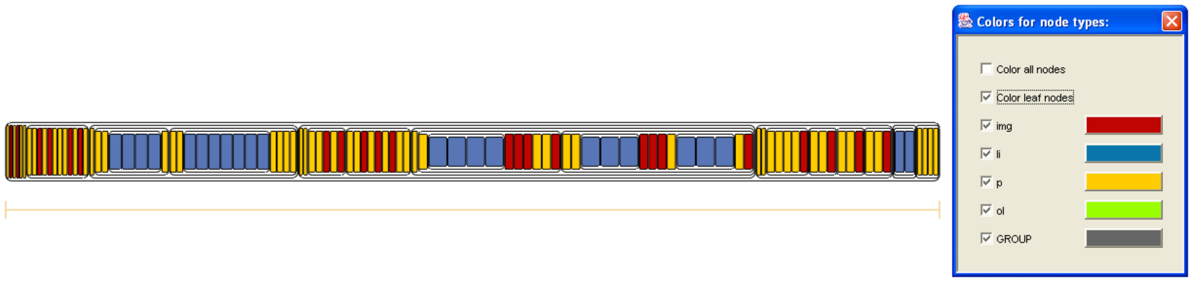
Figure 3.12 shows an example of type coloring applied to a tree containing textual data. As can be seen in Figure 3.12(a), all nodes are enclosed in gray parents of the "group" type. In addition, blue "li" environments are always enclosed in green "ol"-type nodes. Figure 3.12(b) shows that leaf nodes only contain paragraphs (type "p"), images (type "img"), and list environments (type "li"). If some of these text-layout conventions are not correctly applied this type of color coding has been built to reveal such errors. Examples will be provided in Section 5.1 where case studies are discussed.


Colors for node types:

☑ Color all nodes
☐ Color leaf nodes

| | |
|---|---|
| ☑ img | 🟥 |
| ☑ li | 🟦 |
| ☑ p | 🟨 |
| ☑ ol | 🟩 |
| ☑ GROUP | ⬛ |

(a) Color code applied to all nodes in the tree.


Colors for node types:

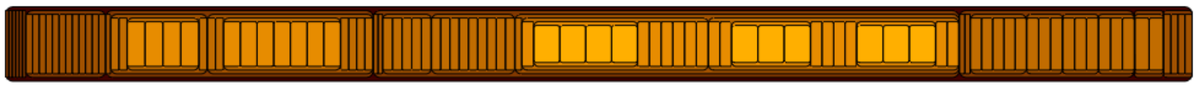
☐ Color all nodes
☑ Color leaf nodes

| | |
|---|---|
| ☑ img | 🟥 |
| ☑ li | 🟦 |
| ☑ p | 🟨 |
| ☑ ol | 🟩 |
| ☑ GROUP | ⬛ |

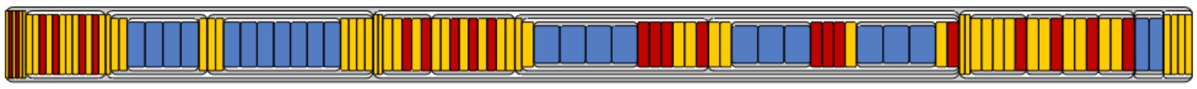
(b) Color code applied to the leaf nodes only .

**Figure 3.12:** Colors mapped to node types represented in the visualization of the tree. The color selection dialog is seen on the right.

Unlike in a node-link diagram, this layout does not provide an implicit ordinal axis for the levels and, therefore, the distance of nodes to the root node. Level information can be retrieved by comparing size and position of nodes in the nested layout. However, level information is hard to retrieve if a small offset is chosen and if many nodes are displayed at the same time. Therefore, color was chosen to also encode level information of tree nodes. Tree levels form a numerical scale between 0 and MAX_LEVEL. Color scales are commonly used to encode numerical values where each value is assigned its own color. Appendix A.3 introduces the color scales available in the *arc tree visualization* and their respective advantages and disadvantages. Depending on the number of levels or if additional type coloring is applied or not certain color scales are more appropriate than others. MACKINLAY (1986) suggests to use color saturation over color hue (cf. Figure 3.1) to encode ordinal data. Nevertheless, the choice of the color scale is left open to the user. Color scales with an emphasis on an increasing color saturation (e. g., gray scales or optimal color scales) are available as well as other scales that place an emphasis on certain hues (e. g., rainbow scale). The individual level color can be chosen as well. Since two types of color encodings can be applied to the tree the type-coding has been chosen to take priority over the level-coding. The user has to ensure that the color codes remain unambiguous. Figure 3.13 gives two examples of color-coding for level information.



(a) Heated Object Scale to color-code level information.

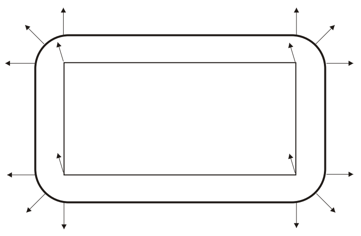


(b) Individual alternating white-gray level colors and additional type coloring chosen for the leaf nodes.

**Figure 3.13:** Examples for color-coding of level information.

## Degree-of-Interest Coding:

In general, a tree contains two types of nodes: branch nodes that have children and leaf nodes without children. Branch nodes can be collapsed and expanded through user interaction. This interaction technique will be further evaluated in Chapter 4. At this point only the visual encoding for the status of a node is of interest. A 3D-depth cue is used to visualize a node that is expandable while nodes that cannot be further expanded are flat-shaded. The illusion of a 3D-depth of a node was chosen as an encoding technique to give the node the affordance of being clickable in comparison to a regular button.


(a) Normal placement to create a 3D-depth cue through lighting calculations.

(b) Application of shading to a sample node to create a 3D-depth cue.

**Figure 3.14:** Realization of node shading to encode that a node is expandable.

## 3.4.5 Visualization of Relations

Another step in the graphical mapping for an *arc tree visualization* is the visualization of relations. During this step, a mapping has to be derived from the set of domain specific attributes to the set of glyph attributes:

$(DA(R) = \{id, type, weight\}) \rightarrow A_{g(R)} = \{shape, color, transparency, width\}$

Proposed mappings will be discussed in the following.

### Shape Selection

In the *arc tree visualization* there are two choices for a glyph shape that both follow the basic shape used to calculate display space during the geometric mapping, a semi-circle or a Chaikin curve. These glyphs will be further parameterized depending on the encoding for $DA(R)$. The used encoding techniques will be discussed in the following.

### Encoding of Size and Position

The position and size of an arc is directly taken from the position calculated during the geometric mapping. The width of an arc, however, is coded according to a specified *weight* for a relation. The larger the weight of a relation the wider is the arc. If more than one relation is specified to connect from Node $a$ to Node $b$ in the tree, only one arc is drawn for the relation and both weights are added to calculate the width of the joint arc.

### Transparency Coding

During interaction with the tree, certain branches can be collapsed or expanded. If a node to which a relation connects resides inside a collapsed branch the relation is drawn to the first visible ancestor of the node. To distinguish relations that point directly to their destination node from those that point only to a visible ancestor, transparency is used as an encoding technique. If an arc can be drawn directly to both nodes it connects it is drawn with a low degree of transparency, almost opaque. A low degree of transparency has been chosen over a fully opaque display to make arcs easier to follow at intersections. If one of the two nodes resides in a collapsed branch the node is drawn with a high degree of transparency to point to a rather imprecise state of the arc.

### Color-Coding:

Color is currently not used as an encoding technique since a cluttered display has to be avoided. Color-coding has been used to a great extent during graphical mapping of tree nodes and it was felt that additional color-coding for arcs should be avoided. Arcs are, therefore, all set to one specific color. However, a possible color-coding for arcs could involve type coloring similar to the type coloring chosen for tree nodes.

Figure 3.15 gives an overview of the encodings for relations. The two arcs in the center have a larger weight and are, therefore, drawn with a different width than the other arcs. One node at the very right is collapsed and the two relations ending at this node's children are drawn more transparently than the other arcs.

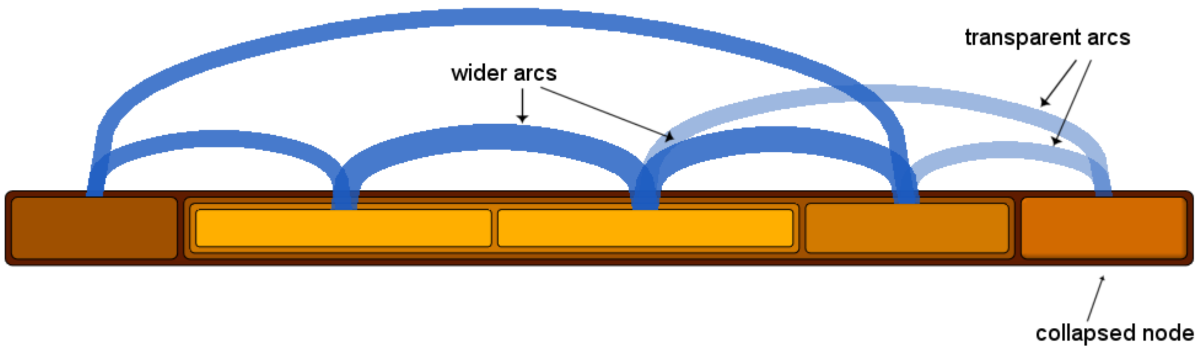


**Figure 3.15:** Encoding for relations.

### 3.4.6 Labeling

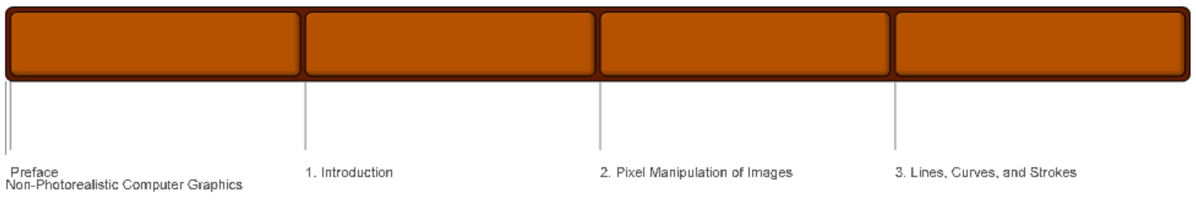
A common characteristic of tree nodes is an association with additional information such as a name or title. Placing textual information at tree nodes is called *labeling*. Several different techniques for labeling of nodes exist depending on the type of the tree layout. In node-link diagrams textual information is usually placed inside a node's representation if enough display space is available. If nodes are too small a common approach involves placing labels only for a selected group of nodes such as leaf nodes. Graph drawing techniques have been developed to generate layouts for graphs with non-uniform vertices in which labels can be placed (HAREL and KOREN, 2002). Another very common approach to labeling graphical objects involves placing the labels outside of the objects. Labels and objects are related through lines connecting anchor points on the objects and their respective labels.

For nested tree layouts such as the *arc tree visualization* labeling is difficult since branch nodes are covered to a given extent by other branch or leaf nodes. Depending on the chosen offset only a small space is available for placing labels inside a node's representation. In many tree-map implementations (cf. Section 2.3.2) this problem has been circumvented by offering context information through a tooltip by hovering the mouse over the node (see, for example, SCHLECHTWEG et al. (2004)). In contrast to the tree-map visualization the one-dimensional layout of the *arc tree visualization* allows the placement of external labels since lines connecting nodes and labels can be arranged to minimize interference with the tree visualization. According to HARTMANN et al. (2004) the following heuristic criteria apply for placement of labels in a visualization:

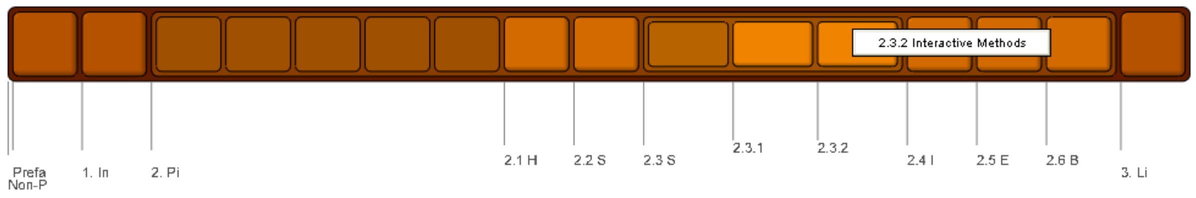
- Labels should be placed near their referenced object,
- The length of the line connecting label and object should be minimized,
- The connecting line should be orthogonal to a main axis,
- Intersections between connecting lines should be prevented, and
- The placement of the anchor point should ease the identification of the pictorial element.

Based on these criteria an external labeling scheme has been developed for the *arc tree visualization* (cf. Figure 3.16). Labels are placed below the layout in a hierarchical fashion. Labels referencing nodes at the same level are placed at the same vertical height to reduce overlap of labels. Anchor points connect to the tree layout below the lower left corner of the bounding box of the object to avoid interference with the tree layout.

However, an overlap of labels cannot be avoided especially when labels are long and/or tree nodes are small. To avoid overlap a more sophisticated label layout needs to be implemented for the *arc tree visualization*. One concept could involve the mere reduction of label length when nodes become small. In focus+context views the size of the labels could be controlled by the placement of a lens and the calculation of interest values. To access the full textual information when labels are reduced tooltips are a helpful mechanism as can be seen in Figure 3.16(b).


Preface
Non-Photorealistic Computer Graphics
1. Introduction
2. Pixel Manipulation of Images
3. Lines, Curves, and Strokes

(a) Initial label layout.



(b) Labels are shortened when nodes become smaller. The full textual information can be accessed via a tooltip.

**Figure 3.16:** Label layout in the *arc tree visualization*.

## 3.5 Summary

This chapter introduced a layout for hierarchical data structures with additional relational information between nodes in the tree. As a prerequisite to the development of a spatial tree and relation layout a set of definitions for the underlying data structure, the relational information, the drawing, and the visualization of the data were presented. The proposed layout emphasizes the following characteristics.

- The support for a layout of ordered trees,
- A constrained spatial layout to allow the use of the visualization as a modular component of information displays,
- Support for the display of external relations through direct links, and
- Mappings for internal aspects of the data such as node types or weights through the use of different encoding techniques such as color, size, position, 3D-depth cues, etc.
- A labeling scheme for tree nodes.

The proposed tree layout can display at most $x$ leaf nodes at a time, if $x$ is the horizontal display resolution. However, a display of this many nodes at a time is hardly useful. Drawing a one-pixel offset around each node of one-pixel width, reduces the number of displayable nodes to $\frac{x}{2} - 1$. In comparison to the resolution of common desktop monitors many trees to be displayed are rather large. The display of large trees would lead to nodes being drawn that are only a few pixels wide which is hard to recognize. Therefore, focus+context techniques will be proposed in the following chapter.

CHAPTER 4

# Interaction with a Focus+Context Visualization

Interaction and navigation are two important aspects of an information visualization. According to SPENCE (2000), a visualization needs to be interactive because the content of data is often unknown. Data exploration is necessary for these unknown aspects to be discovered. In particular, when displaying large hierarchical data sets, no layout algorithm alone can overcome the need for individual exploration of data. Therefore, user specification of visualization parameters is essential. Some possibilities for adjusting color have been discussed briefly in the last chapter. This chapter will be concerned with possibilities to directly adjust the amount of data to be displayed. First, general operations on the data set will be described. These will be extended by focus+context operations for aiding navigation in the *arc tree visualization*.

The proposed interaction techniques all belong to the category of *direct manipulation*. Direct manipulation is a term coined by SHNEIDERMAN (1983) that defines operations that make changes to data *directly* by clicking, dragging, or resizing it. It stands in contrast to *indirect* operations that involve typing commands on a command line or using menus and dialog boxes. The basis for direct manipulation is an adequate iconic representation of objects to be manipulated. These representations also need to have the right affordances to be recognized as being manipulable. PREIM (1999) describes the following characteristics of direct manipulation interaction:

- There exists a graphical representation of the application,
- Manipulation is accomplished through the use of an input device (e.g., a mouse) and transmitted directly to the application,
- The effect of manipulation is seen immediately on the screen (close feedback),

- ○ Direct-manipulation actions are, once learned, easy to remember (rapid recall), and
- ○ The interface is independent of language.

How direct manipulation can be realized in the *arc tree visualization* will be the topic of the following sections.

## 4.1 Input Device

The only input device necessary for interacting with the *arc tree visualization* is a pointing device. The mouse has been chosen since it (or an equivalent such as a touchpad or tackball) is generally available for personal computer systems. No more than three input modes are necessary for controlling the *arc tree visualization*. Two modes are necessary for selecting and deselecting target objects in the visualization and one for calling detail on demand. Therefore, a mouse with two[1] or three buttons offers the exact amount of discrete input options. Users of desktop computer systems are also very familiar with the use of a mouse in direct manipulation interfaces. Three mouse buttons have been assigned the following functions in the *arc tree visualization*.

**Mouse Button 1:** The left mouse button in a right-handed setup.
The amount of data to be displayed changes immediately after a mouse click on a specific target and the display is redrawn to give close feedback to the user.

**Mouse Button 2:** Typically the middle mouse button.
A click with this button calls tooltips with additional information on the objects in the display. Detail-on-demand as required in the *information seeking mantra* (cf. Section 2.1.2) can be accessed. Tooltips are only but then immediately shown when called for. The user, therefore, demands additional information on an object by pressing this mouse button. This stands in contrast to the typical implementation for tooltips. In common graphical user interfaces, tooltips appear after a certain period of time in which the mouse has not been moved. This interaction style often causes detail to be displayed when not explicitly called for.

**Mouse Button 3:** The right mouse button in a right-handed setup.
A click with this mouse button reverses actions called by using Mouse Button 1. The amount of data to be displayed changes immediately and the display is redrawn to give close feedback to the user.

1 The third button in a two-button mouse is usually imitated by pressing both the left and right mouse button simultaneously.

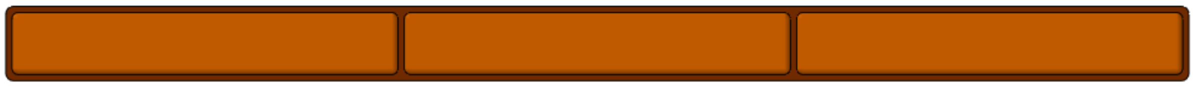
The following sections are concerned with operations called by pressing Mouse Buttons 1 or 3.

## 4.2 Gardening Operations

In complex tree structures interaction techniques for selectively placing focus on certain nodes or branches and deselecting those of lesser interest are absolutely necessary. This is particularly true when tree structures become larger than the display space. In the following, *gardening operations* will be introduced as interaction techniques on a tree structure. Gardening operations are operations to selectively prune or grow the view of a tree following a definition given by ROBERTSON et al. (1991).

### 4.2.1 Growing and Pruning a Tree

Growing and pruning a tree are, in general, very simple interaction techniques. The user can selectively expand nodes by clicking on a node of interest with Mouse Button 1. Once a node has been selected, it is highlighted with a defined highlight color and its child nodes are shown. The tree layout is then adapted depending on the type of node metric chosen as explained in Section 3.4.2. If, for example, the node metric depends on the number of visible children the weight for the selected node is recalculated and more display space is reserved to allow for a larger display of child nodes. Figure 4.1 gives an example of one growing operation applied to a tree. Nodes that are further expandable are drawn with lighting applied to give them a 3D-depth cue. The goal is to ensure that a 3D-appeal, similar to a regular button, ensures the nodes' affordance of being further expandable.


(a) Tree with levels 0 and 1 shown.

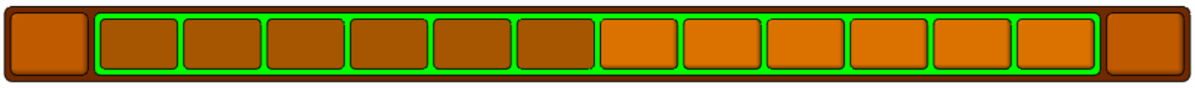

(b) The same tree with the middle node in level 1 expanded. The middle node was clicked on and is highlighted. The node metric selected for the tree layout depends on the number of visible descendants of a node.

**Figure 4.1:** One growing operation applied to the tree through a node click.

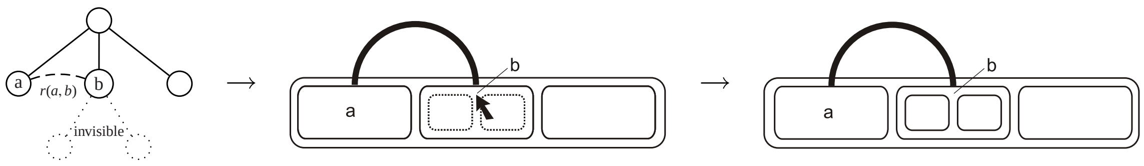
Pruning a tree is just as simple. By pressing Mouse Button 3 on a node a subtree is collapsed. Two interaction modes are offered. The first mode collapses the branch below the node just clicked on. When a large number of nodes have been opened and the offset between nodes is rather small, it might be hard to select branch nodes. Therefore, the second interaction mode is offered that collapses the subtree under the parent of the node just clicked on. The collapsed node is highlighted as a reference to the user.

How these operations affect the display of relations will be the topic of the following section.

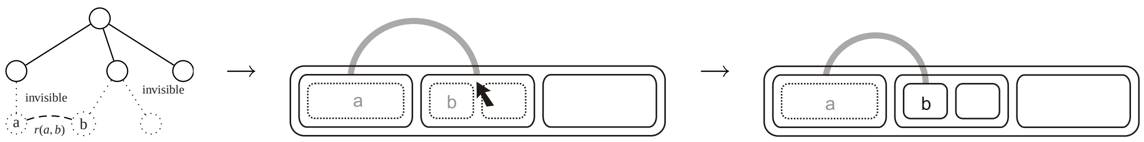
## 4.2.2 Necessary Layout Changes

When additional relational information is displayed for a tree and gardening operations are applied, layout changes for the display of relations are necessary. The following cases have to be regarded for the layout of a relation $r = (a, b) \in R$ after a growing or pruning operation has been applied to a node in a subtree containing either Node $a$, Node $b$ or both:

1. Both nodes $a$ and $b$ remain visible. No layout change is necessary. The following image gives an example of this case after the application of a growing operation.

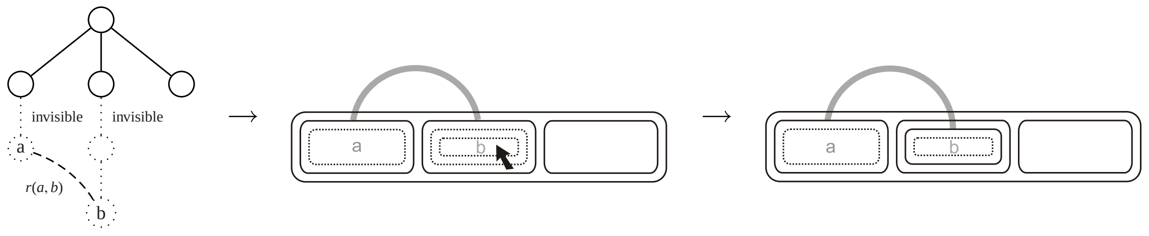


2. Either node $a$ *or* $b$ is hidden inside a collapsed subtree. The relation connects *directly* to the visible node and *indirectly* to the hidden node. The indirect connection ends at the first visible ancestor of the hidden node. The glyph encoding the relation is displayed transparently to show the imprecise state of the relation. The following image gives an example for this case after the application of a growing operation. If more than one relation connects the visible and the first visible ancestor node both are combined to one wider arc.



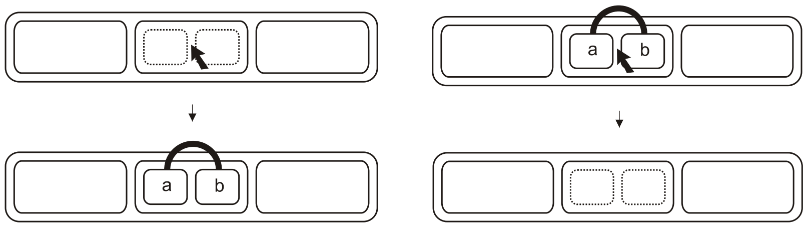
3. Nodes $a$ *and* $b$ are hidden inside collapsed but *different* subtrees. The relation connects *indirectly* to both hidden nodes. In both cases the connection ends at the first visible direct ancestor of each node. The glyph encoding the relation is displayed transparently to show the imprecise state of the relation. The following

image gives an example for this case after the application of a growing operation. If more than one relation connects the first visible ancestors the relations are combined to one wider arc.



4. Nodes a *and* b are hidden inside the *same* subtree. A separate encoding technique is necessary which will be discussed in the following.

According to Case 4 both Nodes a *and* b lie inside the *same* subtree. In this case two different visual effects are possible before or after a gardening operation is applied to a direct ancestor of both nodes: If a growing operation is applied to the parent of a or b a new arc has to be created (cf. Figure 4.2(a)). In contrast, if a pruning operation is applied to the parent of a or b a previously showing arc is deleted (cf. Figure 4.2(b)).



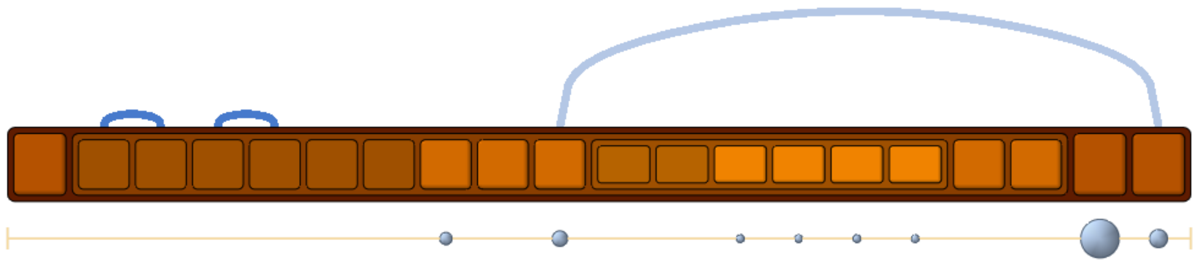
(a) A growing operation applied to the parent of a and b.
(b) A pruning operation applied to the parent of a and b.

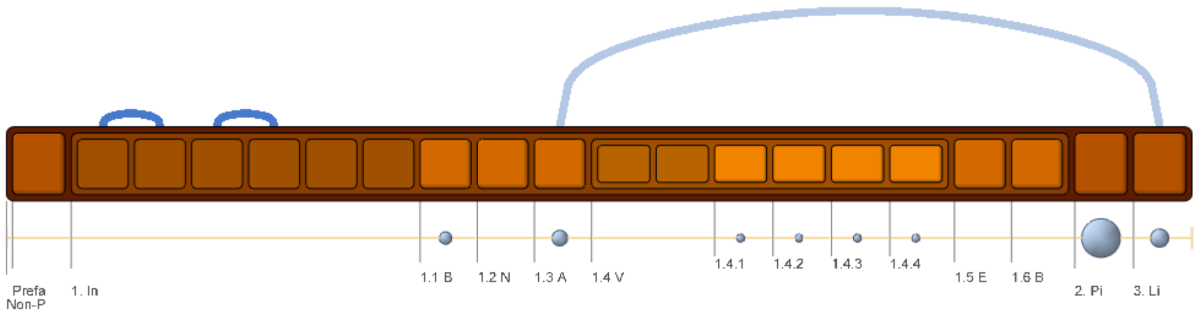
**Figure 4.2:** Gardening operations applied to a direct ancestor of two connected Nodes a and b. Note how the relation connecting both nodes appears or disappears depending on the operation.

To avoid confusion, an indication should be made that there are further relations available below a node. Such an encoding is necessary for three reasons:

1. It prepares the user for the appearance of additional arcs in the expanded layout after a growing operation.
2. It gives an indication to relations that disappear after a pruning operation.
3. If the user is primarily interested in relations it points the user to possible nodes of interest.

Encoding options for hidden relations could include a reference icon directly on the root of the subtree containing the hidden relations, drawing a self-loop, or drawing special glyphs outside of the already created visualization. Appendix A.4 gives an overview of encoding design concepts of which one was chosen for the *arc tree visualization*. To avoid visual clutter it was felt that this information should be provided outside but close to the already developed tree and relation layout. Therefore, a new graphical representation was developed. As an encoding, circular buttons are drawn below a node containing hidden relations. These buttons are placed directly below the drawn arc tree as can be seen in Figure 4.3. This button layout can be arranged to not interfere with the proposed labeling scheme (cf. Figure 4.3(b)). The diameter of


(a) Hidden relations in a collapsed subtree below a node are encoded using circular buttons.



(b) The encoding does not interfere with the proposed labeling scheme.

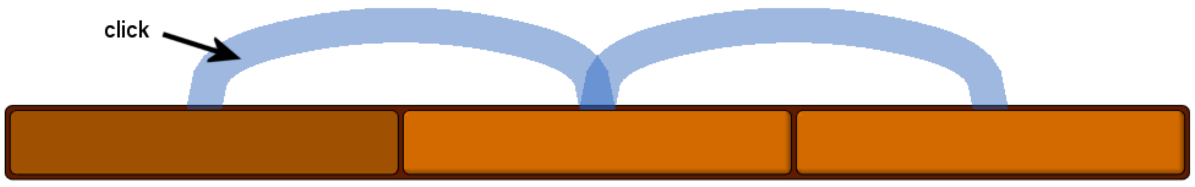
**Figure 4.3:** Encoding for hidden relations in the *arc tree visualization*.

a circle is set proportional to the number of hidden relations so that the more hidden relations a node contains the larger its circular button. All buttons are arranged on a line to allow for easier comparison of circle sizes. Circular buttons were chosen as an encoding technique over the other design concepts for the following reasons:

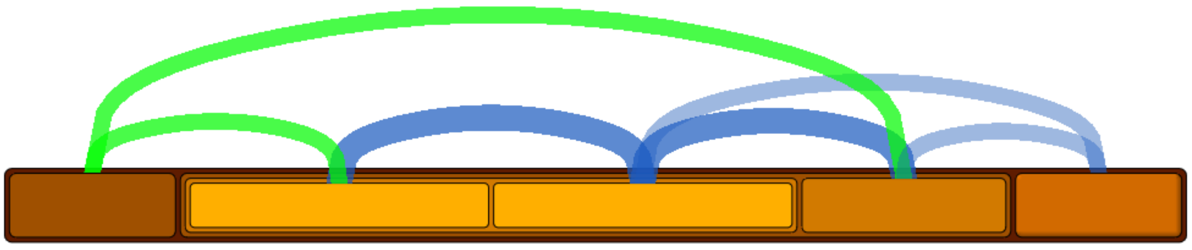
- A large surface area is provided for direct manipulation interaction while little screen real estate is used,
- When many nodes are displayed at a time, clustering of buttons can be easily recognized and used as an indication for an accumulation of hidden relations in this part of the tree.,
- The size of a circle can be used as an indication for the number of hidden relations contained in a subtree,

- ◦ Circles can be drawn with lighting applied to give them the affordance of regular buttons, and
- ◦ The layout has an aesthetic appeal.

If a relation is encoded according to Case 2 or Case 3, growing the tree is also possible by clicking on the arc with Mouse Button 1. After a click occurred, the tree is grown to show all hidden nodes and their siblings at the endpoints of the represented relations. Then the arc clicked on is highlighted. If more than one relation was represented by the arc, it has to be split to show all previously combined arcs. In this case all split arcs are highlighted as can be seen in Figure 4.4.



(a) Tree with level 0 and 1 shown.



(b) The same tree after the left arc in 4.4(a) was selected.

**Figure 4.4:** One growing operation applied to a tree through an arc click.

Figure 4.4 also serves as a more complex example for the need of an encoding for hidden relations. Once the left arc has been clicked on, the hidden nodes at its endpoints are shown. The now visible tree structure causes the right arc to be split as well since its endpoints are now visible. The wide arc in the center connecting the two brightest nodes, however, did not have any connection to either the left or right arc in the previous layout. It appeared unexpectedly which might be confusing for the user.

Growing the tree is also possible through interaction with the circular buttons. Once a button is clicked with Mouse Button 1, it expands the referenced node by one level. The referenced node and its internal relations that might appear are highlighted. If no hidden relations directly connect to the now visible nodes new circular buttons are created for those child nodes that contain the hidden relations. In contrast to the explained growing operation, the represented node could have been expanded to show *all* nodes connected through hidden relations. However, this interaction might cause many nodes to become visible at a time. This would cause large and abrupt layout changes which would be confusing for the user. Therefore, the tree is only grown by *one* level when the user clicks on a button encoding hidden relations.

### 4.2.3 Summary

An implementation of gardening operations gives the user a tool for changing the size of the displayed tree structure. The interaction starts with a global display of the root node. With growing operations the user can selectively zoom in on regions of interest and zoom out using a pruning operation to see which global structure a branch involves. Pruning operations are, therefore, a filter mechanism in that they hide unwanted information. According to the *visual information seeking mantra* presented in Section 2.1.2, gardening operations satisfy the zoom & filter principle. Table 4.2.3 summarizes the implementation of gardening operations in the *arc tree visualization*.

For large hierarchies it is often difficult or even impossible to perceive global and local information in one view since nodes and relations become too small in the display. With gardening operations the user can explore the data structure interactively and integrate global and local information in his or her mind. This integration, however, is not an easy task specifically for novice users of a visualization tool. In information visualization there exist several possibilities to help the user with the integration. One solution from the field of focus+context techniques will be presented in the following section.

## 4.3 Focus+Context Navigation

Focus+context navigation in tree structures introduces an automation for choosing which portions of the tree to grow and which to prune depending on a statement of interest by the user. Which gardening operation is applied to a node in the tree, therefore, depends on an interest function that is defined from a user-centered perspective. The user selects a node of maximal interest, the *focus node,* by clicking on a target node with the mouse. Depending on the selected focus degree of interest (DOI) values are calculated for *all* nodes in the tree. Pruning operations are applied automatically to nodes below a user specified threshold. These nodes are considered “uninteresting” and can, therefore, be collapsed to just give context information. This leads to a logical filtering of nodes. Card and Nation (2002) call this strategy an *attention-reactive user interface*. The following section will now be concerned with the calculation of a degree of interest function in the *arc tree visualization*.

| Operation | Target | Interaction | Response |
|---|---|---|---|
| Growing the tree | Node | Click on the node's representation with Mouse Button 1. | The node's children become visible and the spatial tree layout is recalculated. The node clicked on is highlighted. |
| | Visible Relation | Click on the relation's representation with Mouse Button 1. | The nodes referenced by the represented relations are shown and the spatial tree layout is recalculated. The represented relations are highlighted. |
| | Hidden Relation | Click on the circle representing the hidden relation with Mouse Button 1. | The children of the target node are shown and hidden relations visible at this level are revealed. Both are highlighted. Representations of further hidden relations are shown if necessary. The spatial tree layout is recalculated. |
| Pruning the tree | Node | Click on the node's representation with Mouse Button 3. | The subtree below the node is collapsed and the spatial tree layout is recalculated. Another mode can be chosen in which the subtree below the node's parent is collapsed. |

**Table 4.1:** Gardening operations in the *arc tree visualization*.

### 4.3.1 Degree of Interest Calculation

The interaction technique for applying gardening operations on the tree remains unchanged during focus+context navigation in the tree. Growing and pruning operations are applied for selected nodes depending on the mouse button pressed. The selected node is considered the *focus* node on which DOI calculations are based. When a relation is selected, both connected nodes serve as focus nodes. In this case two DOI values are calculated for a node in the tree. Both foci are then blended by simply selecting the higher of both values for a node.

Before describing the actual calculation of DOI values in the *arc tree visualization*, several constraints on the calculation will be explained.

## Constraints

The *arc tree visualization* is unlike previous visualization methods for trees that involve DOI calculations as explained in Section 2.3.8. Therefore, the calculation of degree-of-interest values will be explained in more detail. Several constraints have to be satisfied for using interest values in the *arc tree visualization*. These will be discussed in the following.

1. *Tree Structure:* Interest values should take the tree structure into account. As a basis for DOI calculations the position of the focus node in the tree is regarded.
2. *External Relations:* External relations might be present in the *arc tree visualization*These relations have to be considered in determining DOI values.
3. *Independence:* Degree-of-interest values should be used for providing a navigation aid in the tree regardless whether external relations are present or not. The calculation should, therefore, be independent of the presence of external relations but take these into account if available.
4. *Node Metric:* The calculated DOI values are to be used as a node metric for adjusting the tree layout according to the selected focus.
5. *Thresholding:* The degree-of-interest is to be used as a threshold for the automatic application of pruning operations. Therefore, the following conditions have to be met.
    a) *Arrangement of DOI Values:* The degree of interest of a node has to be equal or higher than the DOI of its children. A node closer to the root of the tree is considered to be more important since it provides general context information. This constraint is also important since the automatic application of a pruning operation depends on a selected threshold for a node's DOI value. The pruning operation was defined as a filter mechanism to hide "unwanted" or "uninteresting" information. A node with a DOI below the threshold can be pruned only if it has descendant nodes of lower or equal DOI. If the DOI values were defined otherwise a pruning operation might hide "interesting" nodes in a subtree below an "uninteresting" node.
    b) *Arrangement of Focus DOI:* All ancestors of the focus node have to receive exactly the *same* DOI as the focus itself. This constraint is important for the following reasons:
        - The ancestors of the focus node should never be pruned regardless of the chosen threshold,
        - If the DOI is chosen as a node metric more display space should be reserved for ancestors of the focus node in comparison to their siblings.

This ensures more display space for the focus due to the nested arc tree layout, and

- No node should have a *higher* DOI than the focus node.

The remainder of this section will explain the degree of interest calculation in more detail taking the described constraints into account.

## Basic Degree-of-Interest Calculation

The calculation of a degree-of-interest for a node in the *arc tree visualization* draws from ideas presented by FURNAS (1986). In his work the DOI of a node is calculated by combining an *a-priori* structural importance of a node with an *a-posteriori* importance which depends on user selection of a focus node. For the calculation of a DOI the concept of distance between nodes in the tree is important.

**Definition 4.1** *The* distance *between nodes in a tree is defined as the length of the path between source and target node.*

The algorithm by FURNAS (1986) for calculating a DOI comprises three basic steps.

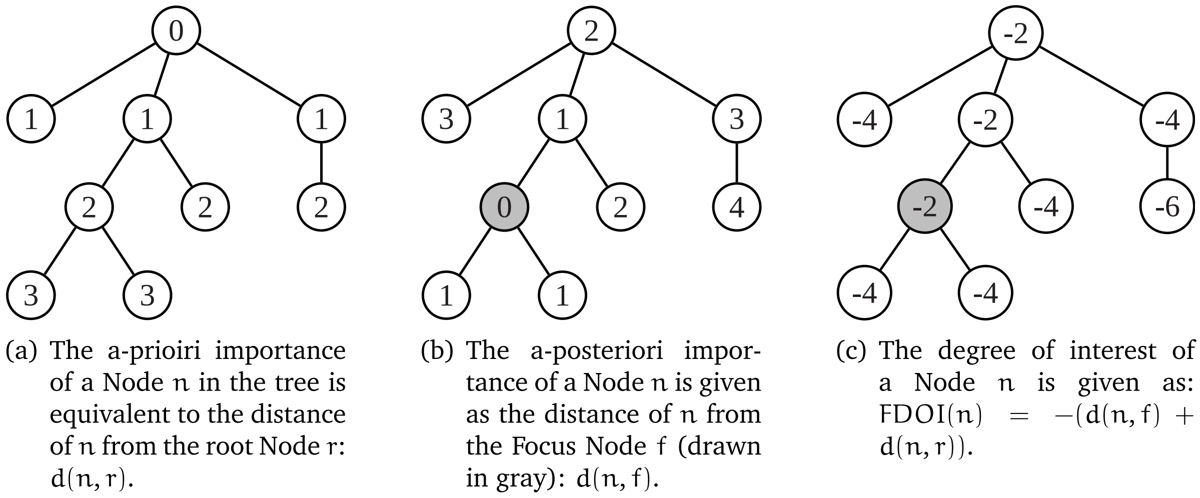
1. *Calculation of an a-priori importance*: The a-priori importance of a Node $n$ in the tree is equivalent to the distance of $n$ from the root node $r$: $d(n, r)$ (cf. Figure 4.5(a)).
2. *Calculation of an a-posteriori importance*: The a-posteriori importance of a Node $n$ is given as the distance of $n$ from the Focus Node $f$ that was selected by the user: $d(n, f)$ (cf. Figure 4.5(b)).
3. *Calculation of the final* DOI *values*: The interest value of a Node $n$ is given as: $FDOI(n) = -(d(n, f) + d(n, r))$ (cf. Figure 4.5(c)). (The minus sign ensures the correct logic behind the calculation: a *larger distance* from the focus means *less important*).

Figure 4.5(c) also shows the existence of a *focus area*. A focus area and its context are defined as follows:

**Definition 4.2** *A* focus area *comprises all nodes of equal degree of interest as the focus node. The* context *area comprises all nodes not inside the focus area.*

This algorithm fulfills three of the initial constraints:

- The tree structure is taken into account as well as the selected focus node,
- The ancestors of the focus node all have an equal degree of interest, and



(a) The a-prioiri importance of a Node $n$ in the tree is equivalent to the distance of $n$ from the root Node $r$: $d(n, r)$.

(b) The a-posteriori importance of a Node $n$ is given as the distance of $n$ from the Focus Node $f$ (drawn in gray): $d(n, f)$.

(c) The degree of interest of a Node $n$ is given as: $\mathrm{FDOI}(n) = -(d(n, f) + d(n, r))$.

**Figure 4.5:** FURNAS' algorithm for calculating a degree of interest for tree nodes. The gray node was selected by the user as the node of highest interest (focus node).

- ○ All children of nodes outside the focus area have a lower degree of interest than their parent.

Some adjustments to FURNAS' algorithm were necessary to allow for the use of the FDOI values in the *arc tree visualization*. These adjustments are necessary to fulfill the remaining constraints. First of all, the degree-of-interest values are to be used as a node metric to create layout changes for nodes of more or lesser interest. For example, focus nodes can be displayed largest while nodes of lesser interest receive less display space. Since node weights are defined as positive, non-zero values FURNAS' algorithm has to be adjusted to allow for positive degree of interest values. Another adjustment is necessary since external relations might be present in the *arc tree visualization*. Nodes connected directly to a focus node by a relation are considered "interesting" nodes as well. Therefore, their degree of interest might have to be higher than the value previously calculated. Therefore, the calculation of degree of interest values for the *arc tree visualization* comprises three steps:

1. Calculation of DOI values according to FURNAS (1986).
2. Recalculation of FURNAS' degree of interest values to positive, non-zero numbers that can be used as a node metric.
3. Adjustment of degree of interest values if external relations are present.

These steps also ensure the independence constraints stated earlier. The calculation is independent of the presence of external relations but take these into account if available. The following sections will describe the last two steps in more detail.

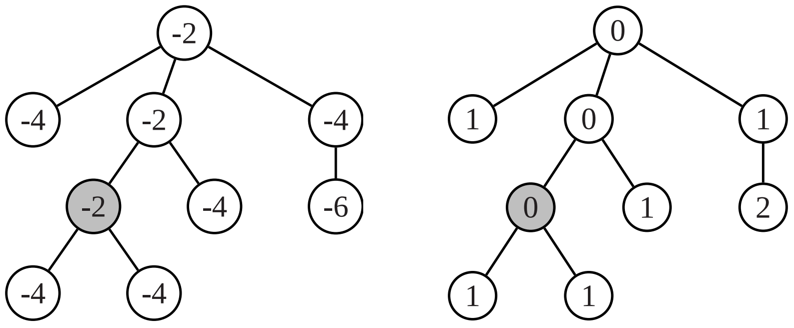
## Adjustments to the Basic Calculation

For FURNAS' degree of interest values to be used as a node metric the initial FDOI values calculated by his algorithm have to turned into positive, non-zero numbers. This adjustment is performed in two steps. First intermediate DOI values are calculated before these are turned into the final DOI values in the range [0 . . . 1].

The intermediate IDOI values are determined by a simple calculation:

$$\mathrm{IDOI}(n) = \frac{\mathrm{FDOI}(n) - \mathrm{FDOI}(\mathrm{focus})}{-2}. \tag{4.1}$$

The resulting IDOI values as seen in Figure 4.6 are zero for nodes in the focus area. All other nodes have a higher IDOI depending on their distance to the closest node in the focus area. The IDOI values have a range of: [$\mathrm{IDOI}(n) \geqslant 0$].



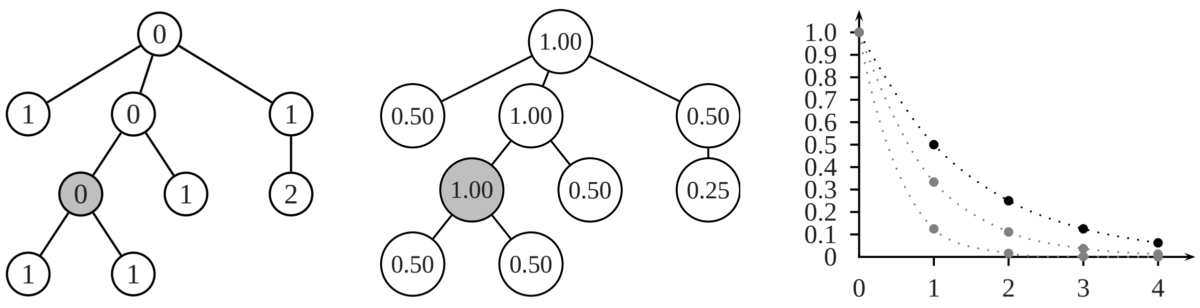
(a) FDOI values. (b) Intermediate IDOI values.

**Figure 4.6:** An intermediate step for recalculating FURNAS' FDOI values for later determining final DOI values for the arc tree visualization.

In the next step, these intermediate values have to be turned into non-zero values. If a node weight equals zero the node is not drawn. Also, to be used as a node metric the focus node should receive a higher degree of interest than nodes outside the focus area. To fulfill these two criteria a non-linear distance function is proposed. The distance of a node is defined to decline exponentially which also allows for faster integration of the focus into the context area. Final DOI values for a Node $n$ ($\mathrm{DOI}(n)$) are determined by:

$$\mathrm{DOI}(n) = \frac{1}{m^{\mathrm{IDOI}(n)}}; m > 1. \tag{4.2}$$

The range of DOI values is now: [$0 < \mathrm{DOI} \leqslant 1$]. If values for $m$ are chosen to be $0 < m < 1$ the focus node receives a smaller DOI value than context nodes. Used as a node metric this would lead to a miniaturization of the focus node and larger sizes for nodes in the context. For $m = 1$ all nodes receive the same DOI value resulting in a *normal view* following a definition by NOIK (1994). Values for $m \leqslant 1$ are, therefore, not considered to be useful in the context of the *arc tree visualization* and will not be used. Figure 4.7 serves as an example for the calculation of DOI values in the arc tree visualization.



(a) Intermediate IDOI values.

(b) The intermediate values are taken to calculate final DOI values ($m = 2$) for the arc tree visualization.

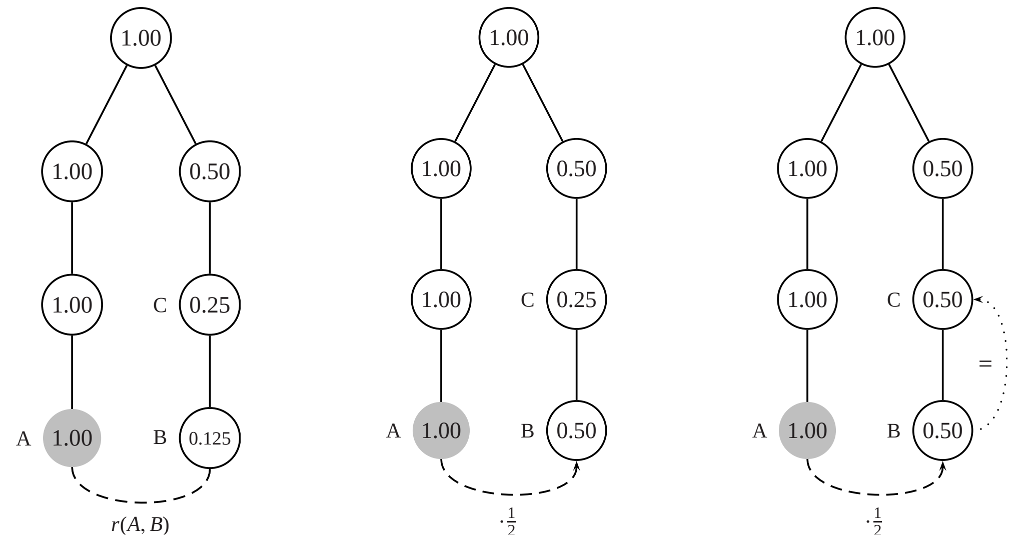
(c) Distance function used. The plot shows three functions from top to bottom using $m = 2$, $m = 3$, $m = 8$, respectively.

**Figure 4.7:** Adjustment to FURNAS' algorithm.

## Considering Relations in DOI Calculation

Relations play an important role in the context of the *arc tree visualization*. Nodes connected with the focus node through a relation are considered "important" nodes. Therefore, an additional step is needed to calculate new DOI values for these nodes. In general, new DOI values are specified for all nodes connected by visible relations from nodes in the focus area. The algorithm "walks" along these relations and adjusts DOI values as needed. For example, instead of the previously defined DOI a Node $n_1$ connected directly with the focus node will receive a degree of interest $\mathrm{DOI}(n_1) = \frac{1}{m} \cdot \mathrm{DOI}(\mathrm{focus})$ unless it already has an equal degree of interest. This is continued recursively so that a Node $n_2$ connected from $n_1$ receives a degree of interest of $\mathrm{DOI}(n_2) = \frac{1}{m} \cdot \mathrm{DOI}(n_1)$ unless its degree of interest is already equal or higher.

Changes to DOI values due to relations also have to take the tree structure into account. One of the constraints at the beginning of this section specified that the degree-of-interest of a child should always be equal or lower than the DOI of its parent. Therefore, if a node receives a new DOI value its ancestors must be adapted as well. If the DOI of a parent is lower than the DOI of its child node the child's degree-of-interest is passed on to the parent node and recursively to all other ancestors. An example of these two steps can be seen in Figure 4.8.



(a) Original tree and its DOI values. The current focus is filled in gray.

(b) Intermediate step. Node B is adapted since it is directly connected to the focus node.

(c) B's parent C has to be adapted. C's parent already has an equal DOI. No further steps are necessary.

**Figure 4.8:** Steps in adapting DOI values for Node B connected to Focus Node A.

The full algorithm takes not only those relations into account that *directly* connect to the current focus but also those that might *indirectly* connect to the focus through one of its descendants. The descendants' relations have to be considered due to the display of relations as explained in Section 3.4.5 and Section 4.2.2. If a relation ends at a node that is currently pruned it is drawn to end at the next visible ancestor of the pruned node (unless it is a hidden relation—in that case the relation is not drawn at all). Therefore, even if a node is currently pruned its relations might be shown at a higher level visible node. The focus node might then be connected to other nodes through lower level relations. This case can be seen in Figure 4.9.

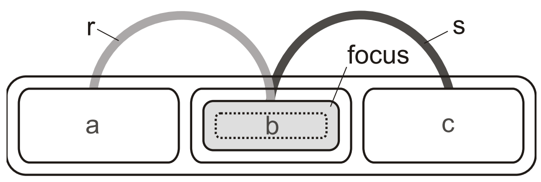
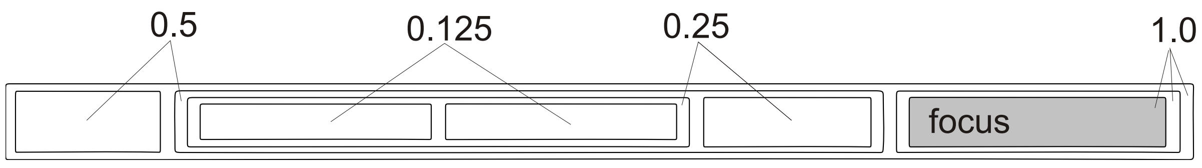


**Figure 4.9:** An arc tree with relation $r(a, b)$ and $s(focus, c)$. Relation $r$ connects to the focus node since the subtree below the focus is currently pruned. Relation $s$ is a *direct* relation while $r$ *indirectly* connects to the focus.

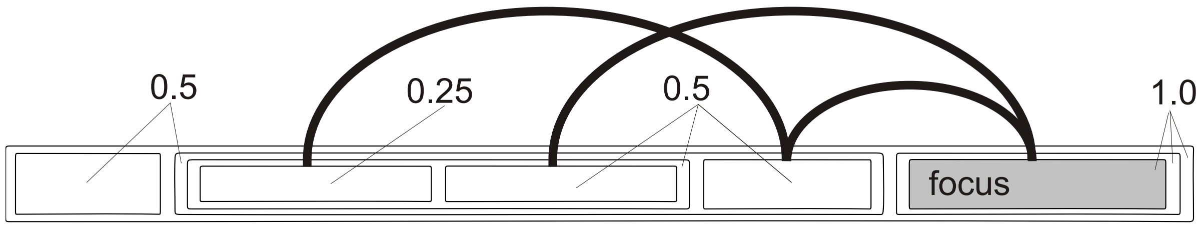
Therefore, the full algorithm for the recalculation of DOI values with consideration of relations comprises the following steps:

1. Collect all relations currently connected to the focus *directly* or *indirectly*.
2. Follow these relations recursively and adjust DOI values if needed.
3. If a new DOI was set for a node adjust its direct ancestors if needed.
4. Repeat the previous steps for all other nodes in the focus area.

To exemplify the algorithm Figure 4.10 introduces a simple example where new DOI values are calculated for nodes connected by relations.



(a) DOI values calculated without relations in the tree.



(b) DOI values redefined according to the displayed relation in the tree.

**Figure 4.10:** Calculations for new DOI values for nodes in the focus area with $m = 2$.

## 4.3.2 Lens Interaction

In an information visualization setting a *lens* generally comprises a movable region on the screen comparable to a large magnifying glass that transforms information below it (GUTWIN and FEDAK, 2004; LEUNG and APPERLEY, 1994). In the *arc tree visualization* the user may place a lens on a focus node by directly selecting the focus with Mouse Button 1. The lens' transformation function serves as an *information filter*. The information filter selects a subset of the nodes in the tree to show to the user. When information filtering is performed, the user of a system is assigned a rather passive role in contrast to information retrieval where information is actively sought. The lens works for the user as an intelligent agent. Information has to be extracted by the user but the lens searches in the background for "interesting" nodes and defines

DOI values from a user-centered perspective. The key consideration in this process is to help the user avoid information overload.

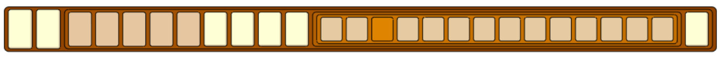
Information overload in the visualization is avoided by an automation for pruning nodes in the tree. Nodes whose DOI value lies below a certain threshold ($s = \frac{1}{m^t}; t = 0, 1, 2, \ldots$) are selectively pruned. A t-*order view* of the tree is specified by the exponent t defining the threshold,[2] following a definition by FURNAS (1986). In contrast to his algorithm, the nodes below the threshold are not completely omitted from the display but collapsed to make all of its descendants invisible. Table 4.2 gives an overview of the first t-order-views and their visual effects when navigating the tree with a lens.

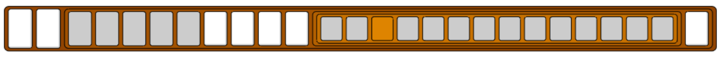
| Order | Visual Effect During Navigation |
|---|---|
| 0 | All nodes with a distance > 0 from the focus area are collapsed. Only the focus and its ancestors remain expanded. If not already expanded, the focus is grown to show the first-level descendants (its children). |
| 1 | All nodes with a distance > 1 from the focus area are collapsed. The siblings of the focus and the siblings of the focus' ancestors remain expanded. At most the second-level descendants of the focus are visible. Therefore, the tree will not grow higher than two levels from the focus node. |
| 2 | All nodes with a distance > 2 from the focus area are collapsed. The grandchildren of the ancestors of the focus remain expanded. At most third-level descendants of the focus are visible. |

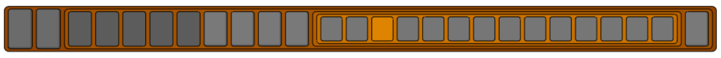
**Table 4.2:** Visual effects of the first three t-level-views of a tree.

Table 4.3 gives an example of focus+context interaction on the tree. An encoding technique to highlight the focus area might be of interest to the user to avoid disorientation (CARPENDALE et al., 1997). The focus node is highlighted through user selection in a specified highlight color. Apart from highlighting the selected *focus node* several different encoding techniques are conceivable to distinguish nodes in the *focus area* from context nodes. Figure 4.11 gives an overview of conceived encoding techniques. However, since most include color changes these were not included in the *arc tree visualization* because they interfere too much with the proposed color-coding mechanisms as introduced in Section 3.4.4. The following section will be concerned with how the focus area can, nevertheless, be distinguished from other nodes through the size in which it is displayed.

2 The exponent t corresponds to the intermediate IDOI values.


(a) Context nodes are colored with the same hue but a different saturation and brightness component.


(b) Context nodes are colored with a light gray color.


(c) Context nodes are colored with a gray color similar in brightness to the original color.

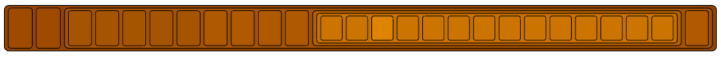

(d) Context nodes are displayed transparent. This mode seems rather unemployable since the colors are blended and the focus is hardly distinguishable from context nodes.

**Figure 4.11:** Possible encodings to distinguish nodes in the focus area from context nodes.

### 4.3.3 Coping with Size

Graphical focus+context views have been introduced (for an early overview refer to (LEUNG and APPERLEY, 1994)) as an extension to FURNAS' initial approach (FURNAS, 1981). In graphical focus+context views the degree-of-interest is mapped to spatial properties of the objects, usually spatial distance and size. One early example was introduced by SARKAR and BROWN (1992) as "Graphical Fisheye Views of Graphs". In their visualization, nodes of greater interest are magnified while correspondingly nodes of lesser importance become demagnified. Positions of all nodes and link bend points are then recomputed to create a new focus+context display. One common characteristic of these graphical focus+context views is that the a-posteriori importance depends on the spatial distance from the focus point. Figure 4.12 shows a typical function for the calculation of magnification factors for objects or points. These functions typically decrease monotonously from the point of maximum magnification.

In the *arc tree visualization*, however, not the spatial but the structural distance from the focus determines the DOI of a node in the tree. Therefore, the DOI values for nodes in the spatial tree layout do not decrease monotonously over the Euclidean space from the focus point. A node located spatially between the focus and another node may have the lowest DOI of the three nodes depending on its place in the hierarchy and its connectedness to the focus. To allow for graphical aspects in the focus+context display of nodes in the *arc tree visualization* DOI values are used for determining a new node metric. In the tree layout algorithm a node's weight at-

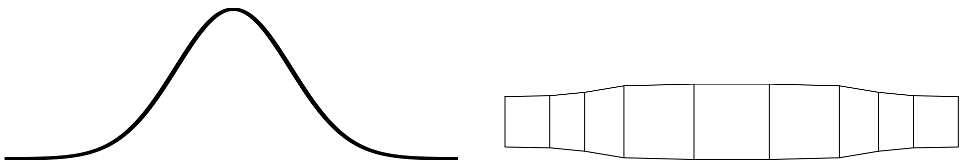

(a) Side view of the magnification function.  (b) Resulting focus+context view.

**Figure 4.12:** Example of a monotonic decreasing function for the calculation of a graphical focus+context view. Image created with the Elastic Presentation Framework by CARPENDALE and MONTAGNESE (2001).

tribute is used to calculate the size and position for later display of the node. A node metric algorithm changes this weight attribute depending on structural or content information present at a node. For further information on node metrics please refer to Section 3.4.2. The following criteria were considered in the implementation of DOI information as a node metric:

1. Nodes with a higher DOI should be displayed comparably larger than their siblings and other context nodes.
2. Direct ancestors of the focus node should become larger as the focus moves deeper into the hierarchy. This allows for larger display of the focus with its distance from the root than with unchanged ancestors due to the nested tree layout.

Equation 4.3 is used to determine a mapping of these two criteria to the weight attribute of a Node $n$. The DOI of $n$ is mapped directly to the weight attribute if $n$ is outside the focus area (i. e. $\mathrm{DOI}(n) \neq 1$). If $n$ is inside the focus area (i. e. $\mathrm{DOI}(n) = 1$) it is adjusted depending on its distance from the focus.

$$weight(n) = \begin{cases} \mathrm{DOI}(n) + distanceFromFocus(n) & : \mathrm{DOI}(n) = 1 \\ \mathrm{DOI}(n) & : else \end{cases} \tag{4.3}$$

$$distanceFromFocus(n) = level(focus) - level(n) \tag{4.4}$$

One drawback of the proposed layout lies in the fact that the focus node might not necessarily be the largest node displayed depending on the sizes of its ancestors and their siblings. This calculation of the weight for a node, however, tries to guarantee a more prominent display of the focus node at deeper levels in the hierarchy than with a direct mapping to alleviate the previously mentioned drawback of the layout. A visual example of this mapping can be seen in Table 4.4 which should be viewed in comparison to Table 4.3.

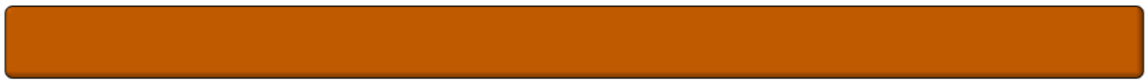
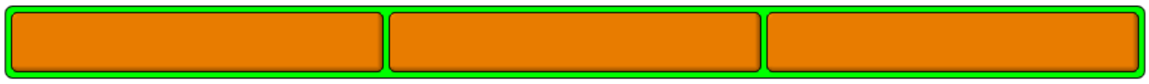
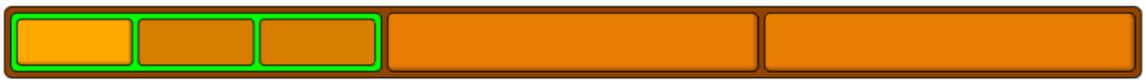
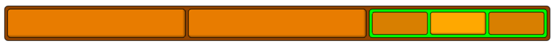
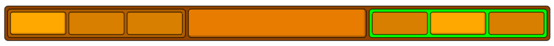
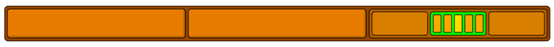
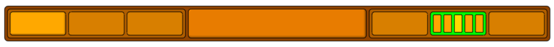
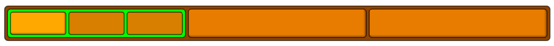
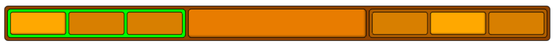
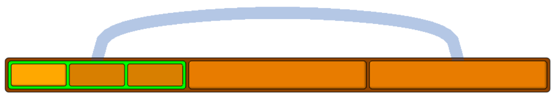
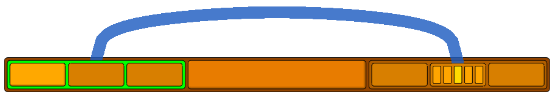
| Zero-Order View | First-Order View |
|---|---|
|  a) Starting point for interaction with the tree is the display of the root node. | |
|  b) The root is selected, highlighted, and expanded. The focus is now on the root node. | |
|  c) The left node is selected as the focus. Its children are shown. | |
|  d) The right node is selected as the focus. The previously expanded left node is collapsed. |  d) The right node is selected as the focus. The left node remains as is. |
|  e) The children of the previous focus can be further expanded without layout changes. |  e) The children of the previous focus can be further expanded without layout changes. |
|  f) The left node is selected. The node on the right is completely collapsed. |  f) The left node is selected. The previous focus node is collapsed since its DOI is below the threshold. |
|  g) The last click repeated with a relation present at the focus node. The layout change remains the same. |  g) The last click repeated with a relation present at the focus node. The previous focus node is not collapsed since its DOI is now above the threshold. |

**Table 4.3:** Sample steps in a focus+context navigation in the tree.

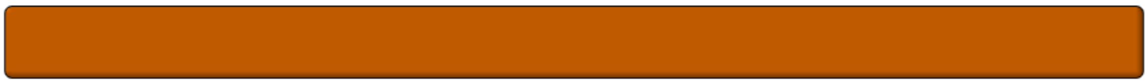
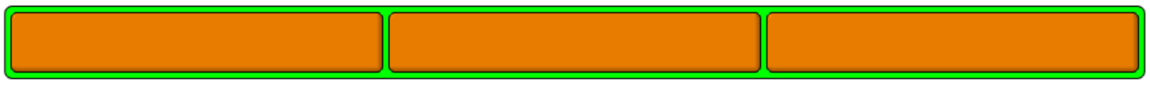
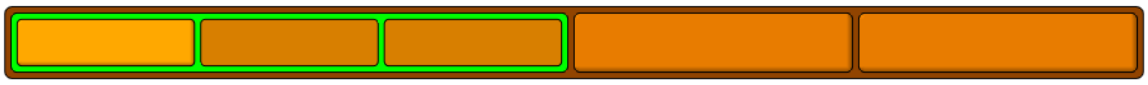
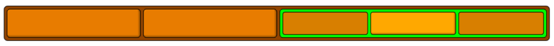
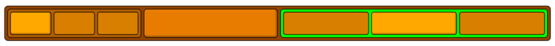
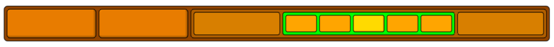
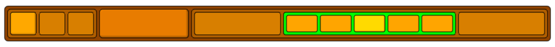
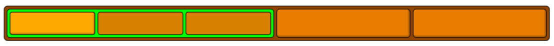
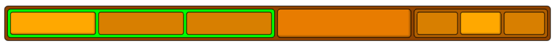
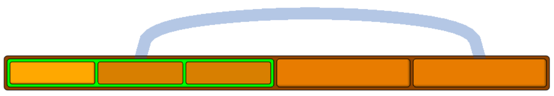
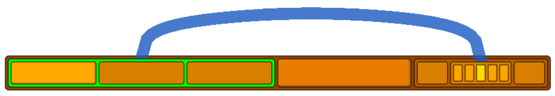
| Zero-Order View | First-Order View |
|---|---|
|  a) Starting point for interaction with the tree is the display of the root node. | |
|  b) The root is selected, highlighted, and expanded. The focus is now on the root node. | |
|  c) The left node is selected as the focus. Its children are shown. | |
|  d) The right node is selected as the focus. The previously expanded left node is collapsed. |  d) The right node is selected as the focus. The left node remains as is. |
|  e) The children of the previous focus can be further expanded without layout changes. |  e) The children of the previous focus can be further expanded without layout changes. |
|  f) The left node is selected. The node on the right is completely collapsed. |  f) The left node is selected. The previous focus node is collapsed since its DOI is below the threshold. |
|  g) The last click repeated with a relation present at the focus node. The layout change remains the same. |  g) The last click repeated with a relation present at the focus node. The previous focus node is not collapsed since its DOI is now above the threshold. |

**Table 4.4:** Steps in focus+context navigation with node weights that depend on the DOI.

## 4.4 Summary

The presented interaction techniques for the *arc tree visualization* introduce a method for selectively zooming in and out of parts of the tree structure. *Gardening operations* were introduced as interaction techniques to prune and grow parts of the tree depending on user interaction. These techniques act as a filter mechanism to eliminate unwanted information and avoid information overload. As an extension, focus+context methods were presented to provide an automation for the filtering mechanism. Depending on user placement of a *focus* node a *degree of interest* calculation proposes interest values for all nodes in the tree. The presence of relations was specifically recognized in the calculation of a degree-of-interest for a node. The layout of the tree is adapted by an automatic application of pruning operations depending on the calculated degree-of-interest values. The user can influence the layout of the tree by selecting a threshold for the application of pruning operations. In addition, the *spatial* layout of the tree can be adapted to correspond in size to the proposed interest values. An algorithm was developed to use the degree-of-interest values as a node metric for the calculation of node sizes and positions in the *arc tree visualization*. One drawback of the proposed layout lies in the fact that the focus node might not necessarily be the largest node displayed depending on the sizes of its ancestors and their siblings. Future work might include further implementation of semantic zoom which involves the display of more or less detail of an object depending on how large it is displayed. In particular, the proposed labeling scheme includes possibilities for additional semantic zoom. Depending on interest values more or less text could be displayed in the label for a node.

Chapter 3 and Chapter 4 have introduced the visualization and interaction possibilities with the *arc tree visualization*. The following chapter will now be concerned with the application of the visualization in different user task scenarios and its evaluation.

CHAPTER 5

# Case Studies and Evaluation

One of the prerequisites for designing the *arc tree visualization* was to design it as a support tool for the use in an application software such as a reading environment, text editor, or calendar. The presented case studies will introduce these and other possible application domains for the *arc tree visualization* and example tasks to be carried out in them. It will be shown how the stated problems for the visualization task in Section 3.1 have been solved and how the set constraints in Section 3.2 have been realized in the context of possible applications. However, at this stage of the development the visualization is not bound to a specific task or application domain. How an evaluation of the *arc tree visualization* can be carried out regardless of having a specific application domain in mind will be the topic of Section 5.2. Specific aspects for which an evaluation would be most appropriate at this stage of the development of the visualization will be highlighted.

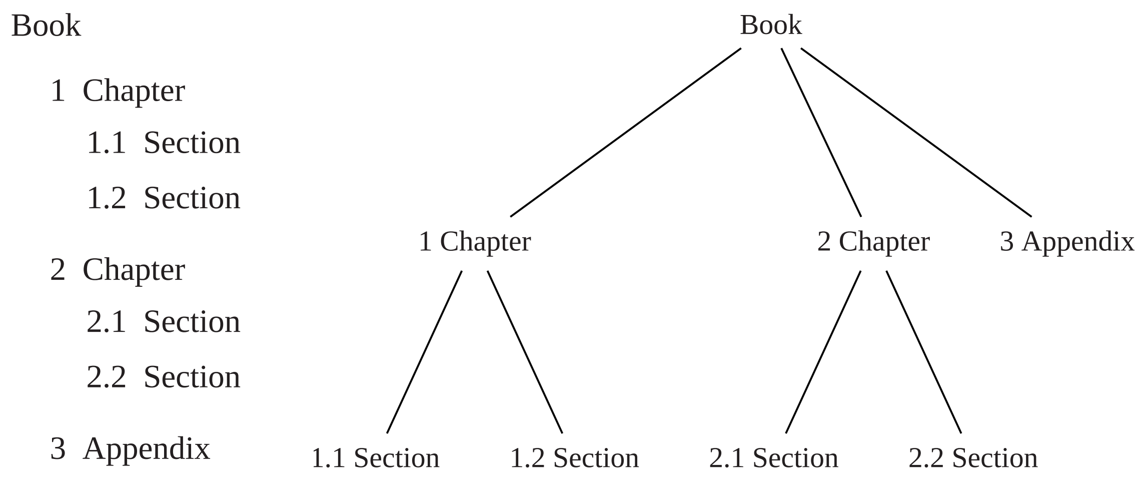
## 5.1 Case Studies

Providing interaction mechanisms for many possible tasks was presented as one of the challenges in the creation of the *arc tree visualization* (cf. Section 3.1). The user of an interactive visualization typically has different tasks at hand depending on the data set provided. Four case studies serve as demonstrations of the use of the *arc tree visualization* in different kinds of applications with different analysis tasks. The *arc tree visualization* will first be used as a visualization tool for structured documents in possible conjunction with a reading environment or a text editor application. The tasks to be supported in this case are reading and writing of scientific texts. In a second example the *arc tree visualization* will be used for analyzing sports tournament data. Calendar data will be used in the third case study followed by a seemingly different visualization of relations in a data set created from the hierarchical structure of a study program.

Different aspects of the visualization will be shown in all cases with an emphasis on visualization and navigation factors. Problems and constraints for building the visualization as stated in Chapter 3 will be discussed in context.

## 5.1.1 Structured Documents

Many written documents are organized according to a specific set of rules. A book, for example, could be described as comprising a title, a table of contents, a series of chapters, an appendix, and a list of references. Each of these parts of the book may be further structured. A chapter could be made up of a title and a preamble followed by a series of sections. Markup languages like SGML, XML, or HTML are examples of descriptions for structured documents. Textual documents structured with many of these languages can be created and maintained in plain text and later rendered in a variety of different ways. In the *arc tree visualization* XML is used as a description for the input of hierarchical data. Structured documents and relations in those documents are an example of a data set that can be displayed in the *arc tree visualization*, as can be seen in Figure 5.1.



(a) Table of contents. (b) The table of contents as a tree structure.

**Figure 5.1:** A hierarchical layout for structured documents.

A structured document can be regarded as a composition of a number of *group elements* (such as sections or chapters) and fine elements, called *chunks*, that form the leaves of the hierarchical structure. Examples of chunks include paragraphs, images, list items, equations, or tables. Such a structure that describes a document in a much finer level than a table of contents is visualized in Figure 5.2.

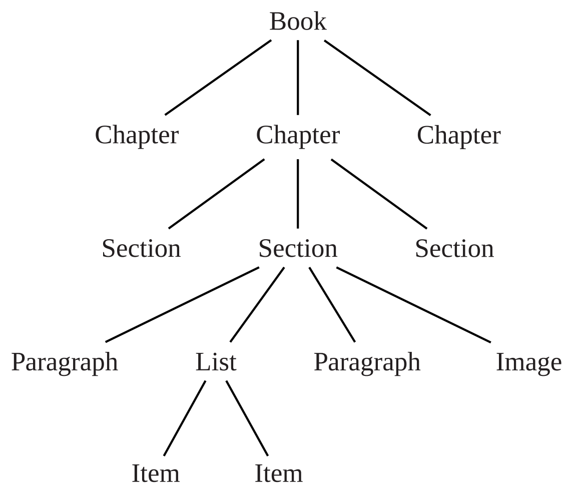


**Figure 5.2:** A structured document is comprised of *chunks* that form the leaves of the tree and *group elements* that themselves contain *chunks* or other *group elements*.

Apart from the inherent parent-child relations many other relations between elements in a structured document are possible. *Direct* relations in a text may include figure, table, section, page, or equation references and citations. Chunks could also be related *indirectly* in the number of keywords they contain from a query or through other attributes like the number of annotations a user made or the date of last change. Examples for relations are now evaluated further in a case study on how the *arc tree visualization* can be used for several tasks involving reading and writing of structured documents.

### Reading

The reader of a scientific textbook is usually interested in only one specific topic of the book. He or she might begin the search for references on a topic of interest by looking it up in the table of contents or by searching the index pages. When skipping to a section that comprises the sought information, the reader might soon discover that in order to understand the covered contents background information from previous sections is required. The book on *"Computer Graphics: Principles and Practice"* by FOLEY et al. (1990) sets a good example. As a standard reference book on topics in computer graphics it covers over 1000 pages of material. If a reader is, for instance, interested in the section on shadows he or she might not have the time to read all previous 700 pages to get the background knowledge covered in earlier chapters. Nevertheless, the reader will discover that background knowledge about light sources, the Phong illumination model, visible-surface determination, scan-line algorithms, or the $z$-buffer is needed when learning about shadows. An author of a book could help a novice reader in a certain research area such as computer graphics by providing additional information on required reading for sections of a book. If a reader is given information which chapters or sections need to be necessarily read before a topic of interest can be understood he or she might save a lot of time skipping back and forth between the section of interest and sections covering background information. The reader could collect information from all required sections first and then page down to the section of primary interest. The *arc tree visualization* can serve as a good visualization tool for such "requires knowledge of" or "is a prerequisite for" relations between elements of structured documents as is shown in the following.

As a reference serves a data structure representing specific parts of the book by FOLEY et al. (1990). In this example, following the definitions given in Section 3.3, a node $u \in N$ stands for a *text chunk* or a *group element*. The incidence function defines the connection for a relation $r \in R$ to the tree: $\phi : R \to N \times R$; $\phi(r) = (u, v)$ with $r \in R$; $u, v \in N$ and without loss of generality the information in $u$ is a prerequisite for understanding the information in $v$.

Figure 5.3 presents an *arc tree visualization* which gives an overview of the book with its 21 chapters and Appendix. It can be seen that the constraint concerning a layout for ordered tree as stated in Section 3.4.2 is important in this case. The viewer sees the chapters in the right order which corresponds to the mental model he or she might already have of the book's structure.

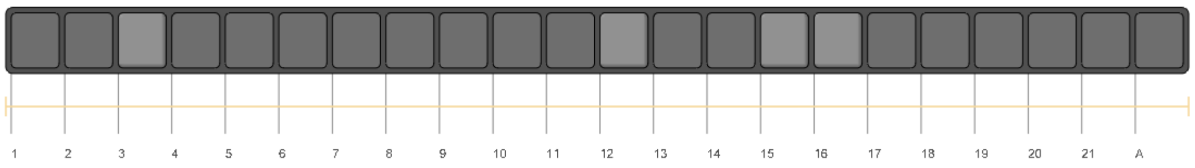


**Figure 5.3:** An overview of the chapters covered in *Computer Graphics: Principle and Practice* by FOLEY et al. (1990).

In this case study the reader is interested in getting information about algorithms for the implementation of shadows in computer graphics. From the table of contents he or she finds out that Chapter 16 Section 4 is concerned with shadows. Having no previous knowledge about shadow algorithms he or she might not be familiar with all the background information needed to understand the algorithms covered in this section. Typically, this deficiency is only discovered while reading the section of interest. If additional information on required reading is provided the *arc tree visualization* can give a good overview of sections needed to be read. Figure 5.4 shows additional information describing "requires knowledge of" relations for Section 16.4. The circular button below Chapter 16 tells the reader that some sections in the same chapter are required reading. The reader will also notice a relatively wide arc between Chapter 15 and 16 which represents more than one connection between both chapters. He or she might then first demand additional information on Chapter 15 and find out through the tooltip that Chapter 15 is concerned with visible surface determination which seems to be an important aspect of shadow algorithms.

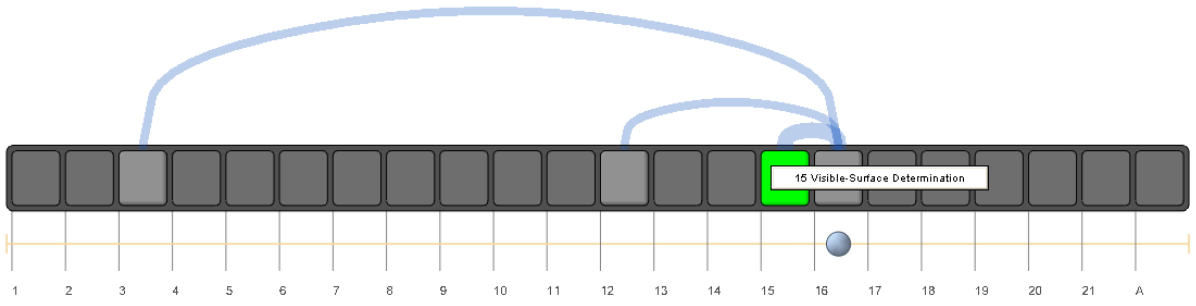


**Figure 5.4:** Relations represent connections to chapters that include information on which sections need to be read to understand Section 16.4.

Focusing on the endpoints of the other connections the user will, for example, discover that the connection to Chapter 3 stands for needed background information on anti-aliasing and Section 12.6.4 covers binary space partitioning trees (cf. Figure 5.5).

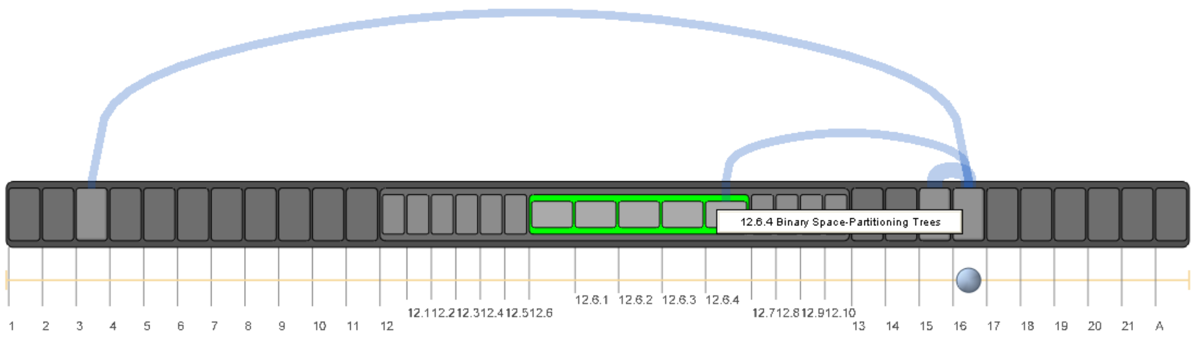


**Figure 5.5:** The focus is placed on Section 12.6. In a next step detail on demand is called for the node representing Section 12.6.4 which is a prerequisite for understanding Section 16.4.

The reader might decide to have enough knowledge on both topics and concentrate on the seemingly most important information in Chapter 15. A click on the arc connecting to Chapter 15 reveals four relations between both chapters. In a focus+context view eight focus nodes are placed in the visualization. This leads to the eight connected nodes being displayed larger in comparison to their siblings giving a clearer picture of the connected nodes in Figure 5.6. By calling detail on demand the reader will notice that the knowledge of four algorithms is needed for implementing shadows: Appel's Algorithm, a *z*-Buffer Algorithm, Scan-Line Algorithms, and the Weiler-Atherton Algorithm. Each of these algorithms is connected to one different shadow implementation covered in Section 16.4. Based on this information the reader can now decide to concentrate on a shadow algorithm for which he already has the required background, to concentrate on a specific one, or to go through all algorithms after learning the required background information.

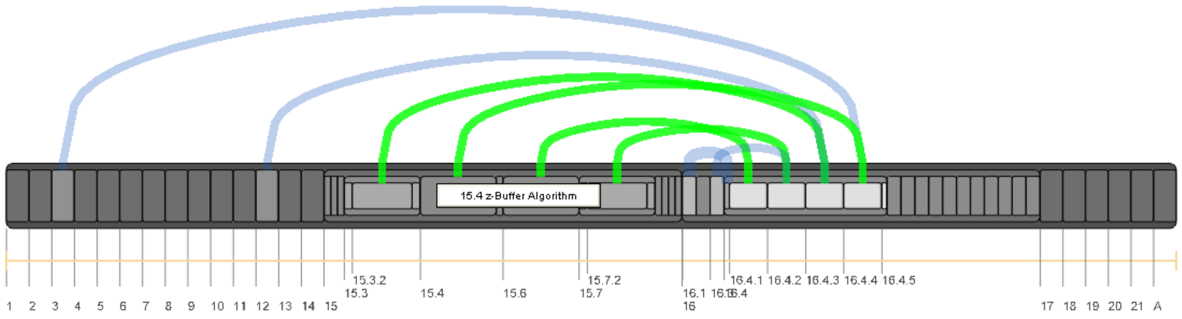


**Figure 5.6:** The focus is placed on all nodes connected by relations from Chapter 15 to Section 16.4.

The *arc tree visualization* might, for the presented example, be used as a modular component of an electronic reading environment. It was built to require a small amount of vertical screen real estate and could be integrated as a larger menu or status bar. One could also imagine the visualization to be rotated by 90° and be integrated vertically as a side bar. This integration would correspond to the reading order in typical electronic reading environments, such as the ACROBAT READER. The interaction on the visualization could be coupled with an interaction in the text currently being read. Clicking on a leaf node in the tree could cause the referenced text chunk to be shown in the reading environment. In contrast, the sections currently being read in the loaded document could be highlighted and focused in the visualization.

### Writing

For writing scientific texts a number of rules, styles, or formats have evolved. These styles typically describe the organization of a scientific text. One set of rules affects direct relations between text elements and their references in the text. Such direct relations may include figure, table, list item, or equation references. THE COUNCIL OF BIOLOGY EDITORS (1994) presents two rules regarding the positioning of tables, and figures:

- Tables are placed in a document as close as possible to their first mention in the text.
- Figures are placed in a document as close as possible to the first point in the text at which they are referred to.

The same rules hold for list items or equation references. Non of these text elements should be placed in a document without a reference in the text. The following case study involves the *arc tree visualization* as a tool for the analysis of these "is referenced at"-relations in a scientific text. In this case study a node $u \in N$ stands for a *chunk* or a *group element*, following the definitions given in Section 3.3. The incidence function defines the connection for a relation $r \in R$ to the tree: $\phi : R \to N \times R$; $\phi(r) = (u, v)$ with $r \in R$; $u, v \in N$ and without loss of generality $u$ is a figure element and $v$ a group element, paragraph, list, or a list item.

We assume that an author wants to check his or her text on errors regarding figure references. He or she wrote a book that covers topics on non-photorealistic rendering in three chapters and a preface as can be seen in Figure 5.7.

From the text the author of the book extracted references between figures and the paragraphs in which the figures are mentioned. He or she loads these references in the *arc tree visualization* with the result shown in Figure 5.8(a).

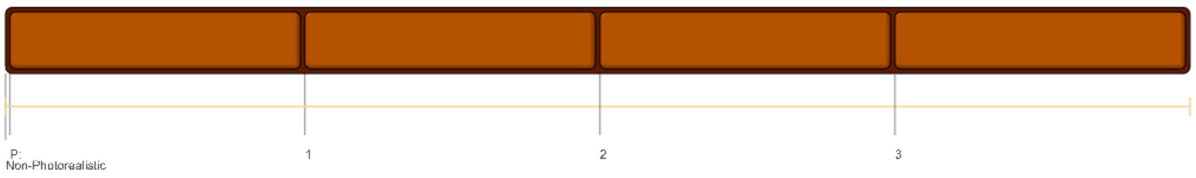


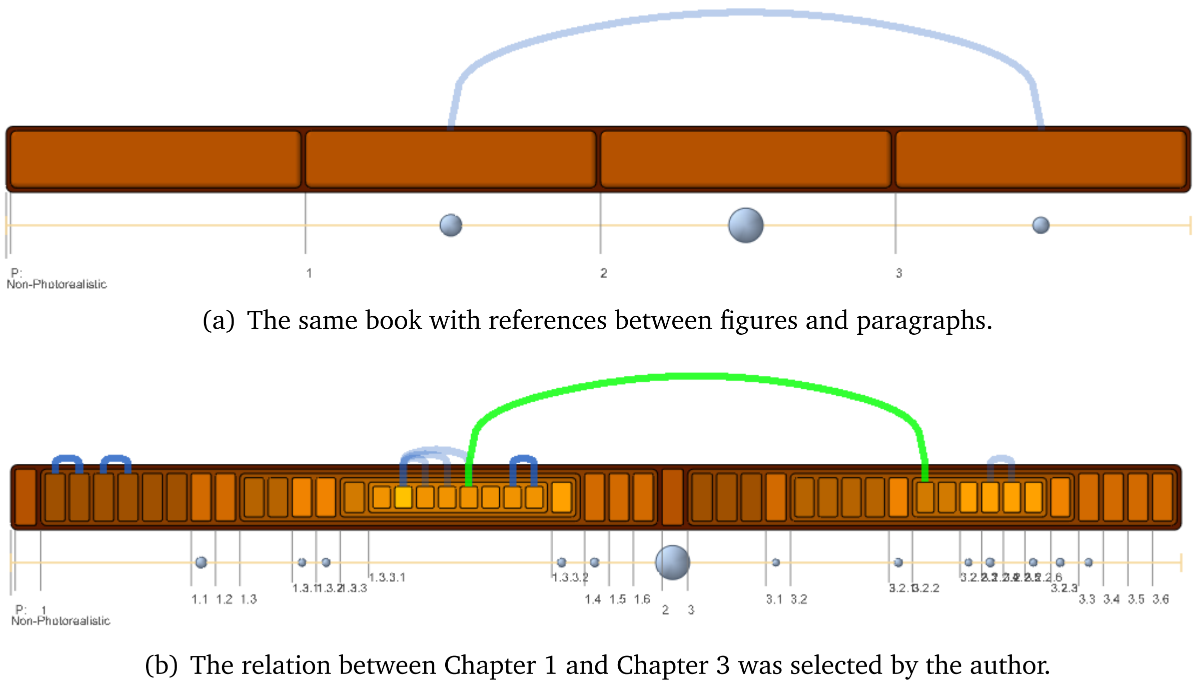
**Figure 5.7:** Part of a book covering non-photorealistic rendering in three chapters and a preface.

The first striking relation extends from Chapter 1 to Chapter 3 of the book. Since it is rather uncommon for pictures to be referenced from another chapter the author decides to click on the arc. Both endpoints of the relation are revealed as seen in Figure 5.8(b).

A large layout change is the result of this interaction. However, since the author is well acquainted with his or her own writings he or she will be able to integrate the changes with his or her mental model of the data set. A user less acquainted with the data might need smooth transitions between the two views to understand the layout change. The implementation of smooth transitions will be discussed further in Section 6.2.

In order to differentiate between the various types of nodes in the visualization the author opens a color chooser dialog and sets colors for the different types of nodes present in the visualization. The different colors and node types can be seen in Table 5.1 and Figure 5.9.

After the chosen colors are applied to all nodes, the visualization changes as shown in Figure 5.10. It can be seen that a paragraph in Chapter 3 referencing an image in Chapter 1 is the result of the wider arc between both chapters. This reference can now be checked and corrected by the author if necessary. The author also notes that the image stands in context to two other images and is also referenced in a list



(a) The same book with references between figures and paragraphs.

(b) The relation between Chapter 1 and Chapter 3 was selected by the author.

**Figure 5.8:** The same book augmented with relations and the first interaction step performed on the visualization.

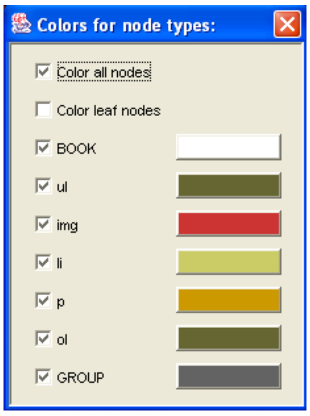
environment preceding the three images. The visualization of hidden relations directs the user to group nodes that contain further figures and references. These can, in turn, be opened to verify that the relations were correctly specified.

In the next step the author wants to verify that he or she did not miss any references in the preface. A lack of direct reference and hidden relations identified by the absence of a circular button below the preface node was noticed by the author. He or she clicks on the preface node and a low threshold for lens interaction causes all open chapters to close. The author then chooses a higher threshold and opens all group nodes in the preface. As can be seen in Figure 5.11 the preface contains only paragraph nodes and one list. No references to figures were missed in the preface.

The author of a text might not necessarily be interested in checking all relations in the document but concentrate on sections recently edited. With the *arc tree visualization* structural problems in the text can also be discovered. For example, the first section below a chapter should be preceded by an introductory paragraph, a section should commonly cover more than one subsection etc. Such writing styles could be verified by analyzing the layout of the tree structure. The *arc tree visualization* might be especially helpful for authors with little writing experience who are often less organized in the creation of a document than experienced writers. It could be integrated as a modular component in a text editor. An author could call the visualization on demand to verify

| Type | Name | Color |
|---|---|---|
| Book | BOOK | white |
| Group | GROUP | gray |
| Lists | ul, ol | dark green |
| List Item | li | light green |
| Paragraph | p | orange |
| Image | img | red |

**Table 5.1:** Node types and their colors chosen by the author.


Colors for node types:

☑ Color all nodes
☐ Color leaf nodes
☑ BOOK ⬜
☑ ul 🟩
☑ img 🟥
☑ li 🟩
☑ p 🟧
☑ ol 🟩
☑ GROUP ⬛

**Figure 5.9:** Color chooser dialog.

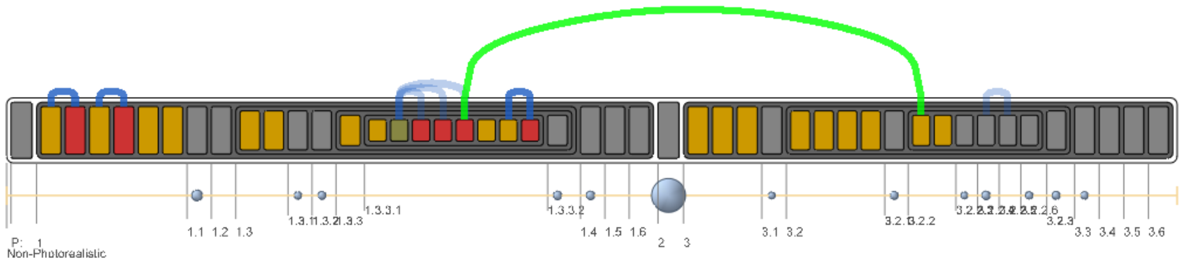


**Figure 5.10:** The relation between Chapter 1 and Chapter 3 was selected by the author. Additionally colors are applied according to node types.

his or her writing style and references. Found errors could then be directly corrected in the text with the visualization in mind.

In the last two case studies several of the previously mentioned constraints for the visualization can be observed:

- The visualization offers an ordered tree layout essential in the interpretation of structured document data,
- The display of arcs in the visualization allows for fast interpretation of external relations,

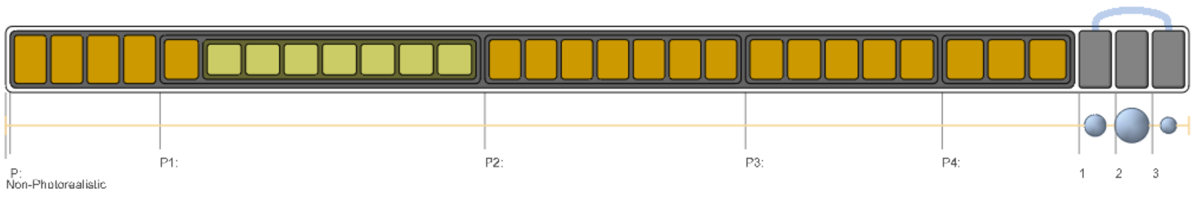


**Figure 5.11:** The focus is placed on the preface of the book. All other chapters are closed automatically.

- Interaction mechanisms allow the exploration of the data,
- A global overview of the data is given,
- Affordances are provided to help the identification of expandable nodes, and
- An effective use of screen real estate is provided.

These constraints can also be observed in the following case studies but will not be specifically mentioned again. The emphasis will now be placed on the presentation of possible application domains and the analysis tasks involved.

## 5.1.2 Sports Tournaments

Sports tournaments played with an elimination system are another common hierarchical data set. Figure 5.12 presents an extract from a typical tournament diagram. The diagram shows the semi-finals and the final game. The winner stands at the root of the tree.

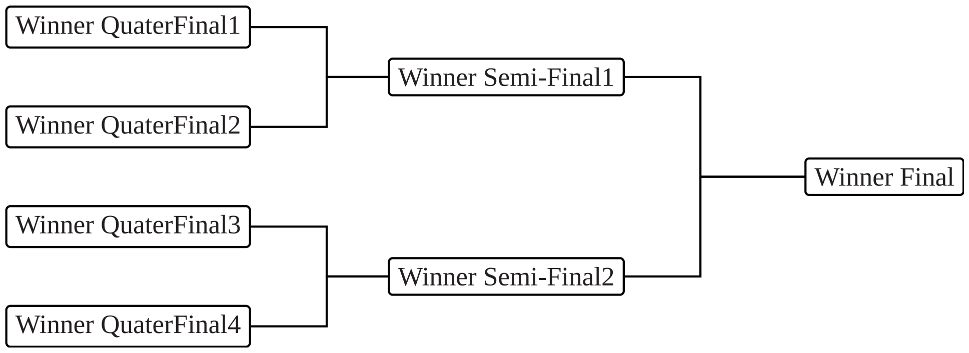


**Figure 5.12:** An extract from a typical tournament diagram. The diagram shows the semi-finals and the final game. The winner stands at the root of the tree.

Tennis tournaments are commonly played in an elimination ladder system. Figure 5.13 gives an overview of the 127 games played in the ladies' singles tournament at Wimbledon 2004. The leaf nodes are formed by the 128 female players who competed in the first round of the tournament.

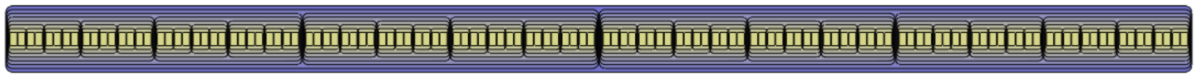


**Figure 5.13:** An overview of the Wimbledon 2004 ladies' singles games.

Several different types of relations can be conceived for a tournament like Wimbledon. Many people might be interested if players of their own nationality excelled at a tournament. Following the definitions given in Section 3.3, such an example is defined as follows. A node $u \in N$ stands for a player who competed in Wimbledon's ladies' singles tournament 2004. The incidence function defines the connection for a relation $r \in R$ to the tree: $\phi : R \to N \times R$; $\phi(r) = (u, v)$ with $r \in R$; $u, v \in N$ and without loss of generality $u$ competed against $v$ in one round of the tournament and either $u$ or $v$ have a given nationality.

For this year's games with German participation this leads to a clear picture in the overview display. Only three German players competed in the ladies' singles tournament and all of them lost in the first round as can be seen in Figure 5.14. Nodes representing German players were drawn in red. For one player additional detail was called and her name is revealed as Ms. Weingartner in the displayed tooltip.

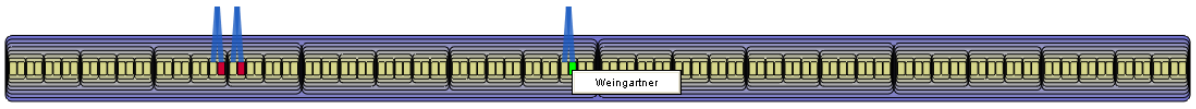


**Figure 5.14:** Relations represent games with German participation at Wimbledon's ladies' singles tournament 2004. Red nodes stand for German players.

For female players with a US passport the same visualization leads to a cluttered display as can be seen in Figure 5.15. Clearly, a large number of US-players competed and the larger the arc the further the players advanced in the tournament. Each red node in the tree signifies a US-player. Taking a closer look at the visualization the larger stack of red nodes signifies a rather successful US player who advanced far in the tournament. From this image it is hard to see which first round games involved US-players. As an extension to the *arc tree visualization*, in this case a filter mechanism for the view of relations might be of interest. Relations could be filtered by the level to which they connect.

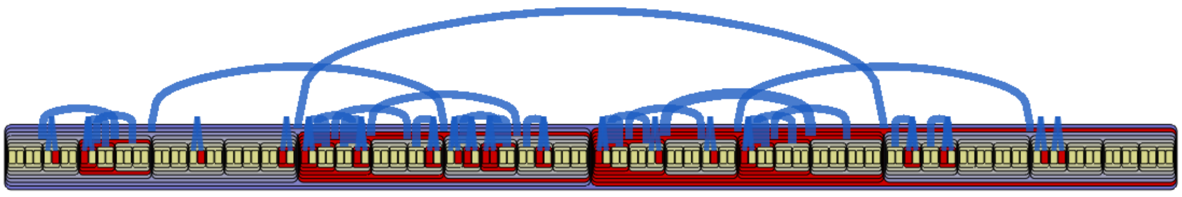


**Figure 5.15:** Relations indicate games with US participation at Wimbledon's ladies' singles tournament 2004. Red nodes stand for US players.

To find out more about the later games in the tournament the user could switch to a lower level view of the tree. Figure 5.16 shows this year's final. The red node stands for Serena Williams, a US player identified by the label displayed at her node. She did not win the game as can be seen by the blue colored root node. This example shows one further extension possibility for the *arc tree visualization*. If additional information is provided with the loaded relations these could be displayed on demand by clicking on the arc. In this example, a tooltip would display the results of the game as prototypically shown in Figure 5.16.

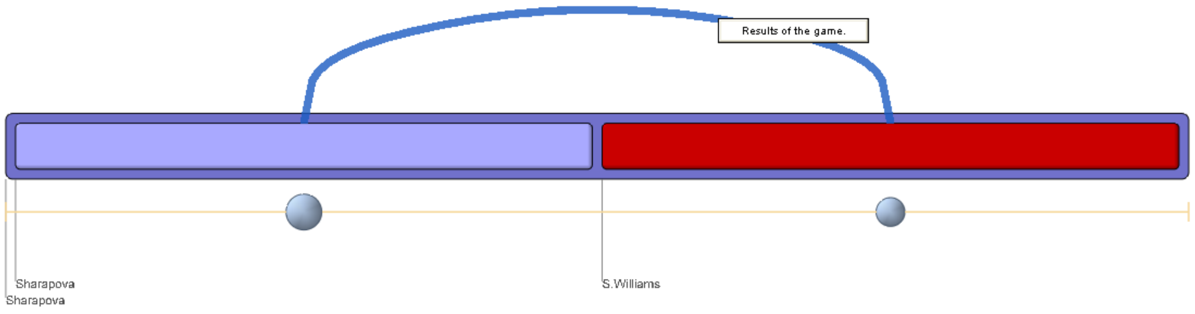
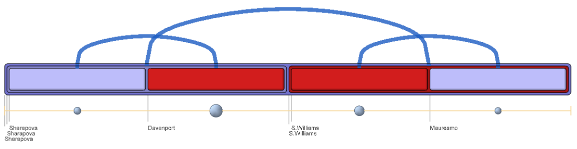
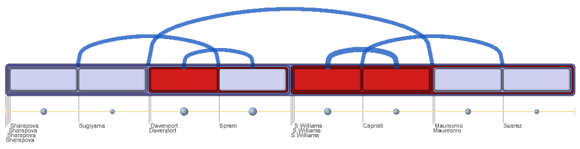


**Figure 5.16:** An overview of the Wimbledon 2004 final in the ladies' singles tournament. The red nodes stand for a US player. A tooltip could display additional information for relations.

Figure 5.17 provides a further overview of the last games in the Wimbledon ladies' tournament. As can be clearly seen both semi-finals had US-participation while only two of the quarter-finals were played by US players. One of the quarter finals, however, was played solely by US ladies which can be recognized from the wider arc and the two red-colored nodes.



(a) Semi-finals and final.



(b) Quarter-finals, semi-finals, and final.

**Figure 5.17:** Wimbledon 2004 ladies' singles tournament. Relations represent games with US participation. Red nodes stand for US players.

Wimbledon is one of the tournaments that attract a high number of top-ranked players each year. To allow for a large crowd of spectators to watch games with star-players the Wimbledon organizing committee ensures that:

> *"Every effort is made to have the potentially most attractive matches played on the courts with the most spectator accommodation but care has also to be taken to give any player likely to go far in the tournament his or her fair share of the show courts."* (AELTC and IBM Corp., 2004).

Another relation that can be visualized with the *arc tree visualization* involves two players who played against each other on the Center Court, one of Wimbledon's show courts. This type of relation $r$ is defined as: $\phi(r) = (u, v)$ with $r \in R$, $u, v \in N$, and without loss of generality $u$ played against $v$ on the Center Court during one round of the tournament. Figure 5.18 gives an overview of all games played on the Center Court at Wimbledon's 2004 ladies' singles tournament. A new labeling style is introduced. Labels are only shown for nodes connected by a relation. Although labels overlap it can be seen that number one seeded Serena Williams played most of her games on the Center Court. Counting the relations it can actually be discovered that she played in 6 of the 15 Center Court games.

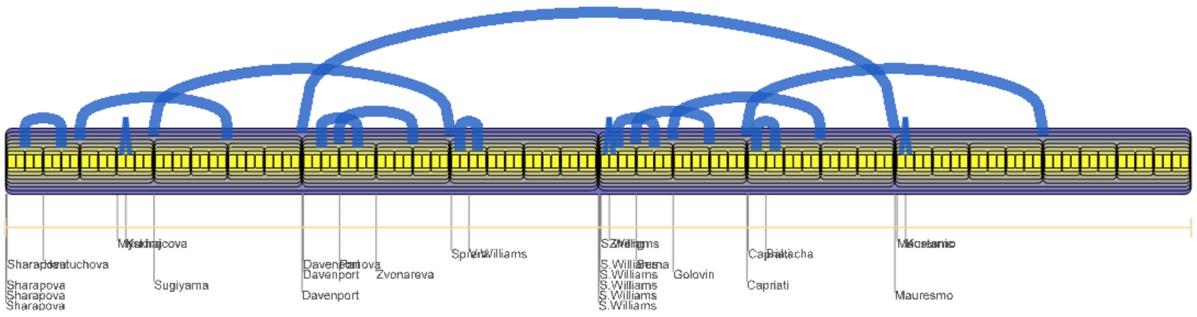


**Figure 5.18:** An overview of games played in Wimbledon's 2004 ladies' singles tournament. Relations stand for games played on the Center Court. Labels are only displayed for nodes connected by a relation.

Being the most important court at Wimbledon the Center Court hosts the most attractive games as can be seen in Figure 5.19. The user of the visualization zoomed into the four lowest levels of the tree to see the quarter-finals and all later games. As can be perceived the final and semi-finals were all played at the Center Court. In the quarter-finals only the top seeded game between Serena Williams and Jennifer Capriati was played on the Center Court and a game from the parallel subtree which would possibly put forth another of the later finalists: Maria Sharapova vs. Ai Sugiyama.

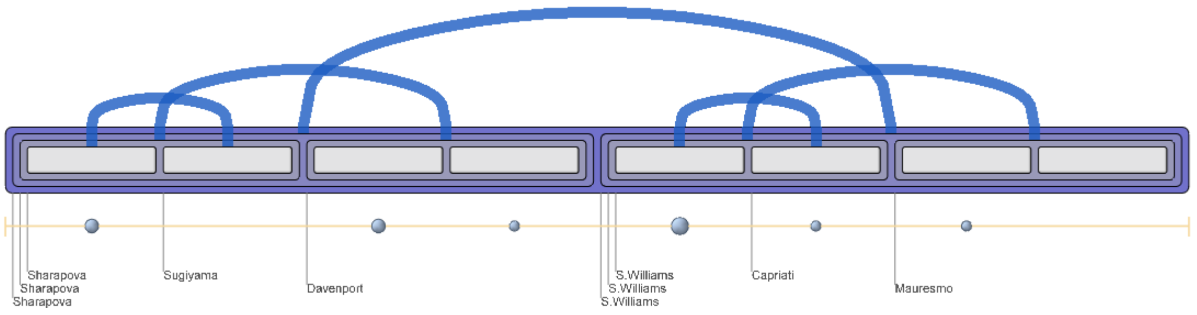


**Figure 5.19:** Quarter-finals and later games played in Wimbledon's 2004 ladies' singles tournament.

The *arc tree visualization* could be used as a visualization integrated as an interactive component in web-pages on sports tournaments. Such a visualization would augment typical tournament diagrams or tables with tournament results.

## 5.1.3 Calendar

Much effort in today's life is put into managing a large amount of appointments. Calendar applications typically focus on providing a good interface for organizing one's personal schedule. However, *visualizations* for calendar data are rather rare. In particular, an analysis of recurring events is difficult in common applications. The user of an electronic calendar might be particularly interested in the start and end times or possible exceptions to an otherwise regularly recurring appointment. Recurring events are typically encoded by an icon (as in MS OUTLOOK), by a highlighted border (as in the MOZILLA CALENDAR Project), or displayed with small dots in an overview window (as in the PALM DESKTOP Software). Getting an overview of recurring events is rather difficult with each of these visualizations. Start and end times or the type of recurrence (i.e. monthly, weekly, etc.) have to be found by going through the whole calendar or by calling detail on demand on one occurrence of the appointment.

The *arc tree visualization* is not an ideal environment for managing one's personal calendar since it does not offer an interface for editing events but it might be a valuable visualization tool for recurring events as a modular component of a calendar application. In this case study one year's calendar serves as the hierarchical data structure. An exemplary tree structure for calendar data is presented in Figure 5.20. In this case study a node $u \in N$ stands for an *event* or a *period of time*, such as a year, month, or day, following the definitions given in Section 3.3. The incidence function defines the connection for a relation $r \in R$ to the tree: $\phi : R \to N \times R$; $\phi(r) = (u, v)$ with $r \in R$; $u, v \in N$ and without loss of generality $u$ and $v$ are nodes of the *event* type and $v$ is an immediate recurrence of $u$.

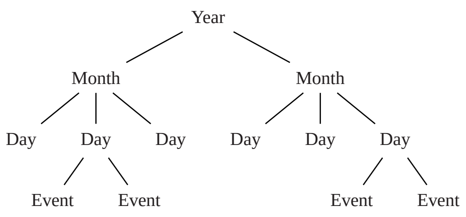


**Figure 5.20:** Example of a hierarchical calendar data structure.

An overview of the year 2004 and a sample set of events and their relations is given in the *arc tree visualization* in Figure 5.21. From this figure one or possibly more recurring events are noticed that last from April to August. From the different widths of the arcs and circular buttons a slight discontinuity of the recurrence can be inferred.

To get a closer look the user zooms into the months of interest as can be seen in Figure 5.22. From this view the user can get an insight into the type of recurrence of the presented events. One event occurs monthly while the other one seems to be a

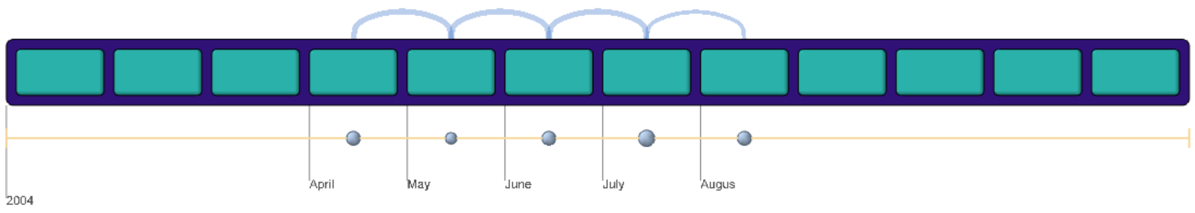


**Figure 5.21:** An *arc tree visualization* of the year 2004 and a sample set of events.

weekly appointment with one exception in the middle of May. The second event lasts until August while the first event ends in July. This is the reason why the last arc in Figure 5.21 is just slightly smaller than the previous arcs.

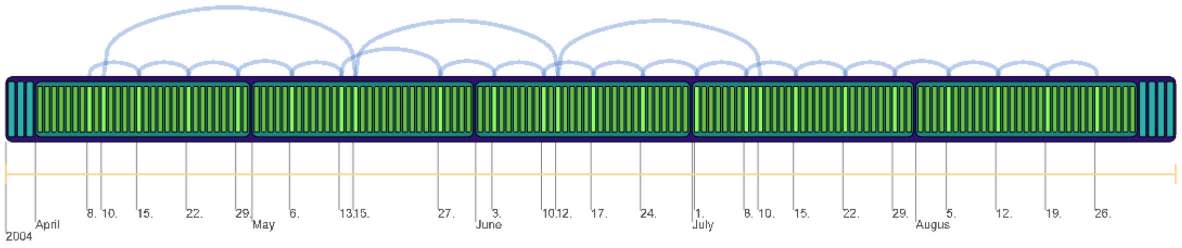


**Figure 5.22:** The user zoomed in on the months April to August.

For further insight into the weekly appointment and why it is not scheduled to take place during one week in May the user decides to place the focus on the month of May in Figure 5.23. He or she zooms in on the weekly appointment and calls detail on demand on the type of event. The user is familiar with German holidays and notices that May 20th is a holiday which is the reason that no appointment is scheduled.

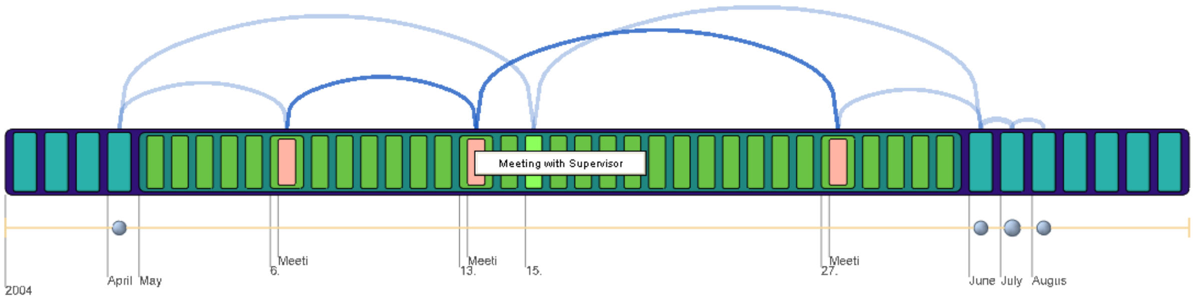


**Figure 5.23:** The user zoomed in on the months April to August. Detail on demand is called for one occurrence of the weekly appointment.

If many more recurring events are displayed it becomes more and more difficult to get a clear picture of a single stream of events. However, as can be seen in Figure 5.24, the *arc tree visualization* can be used as an analysis tool to identify busier and quieter weeks or months in one's personal schedule. From mid-April to mid-July the schedule is rather dense while the number of appointments starts to decline beginning in mid-July.

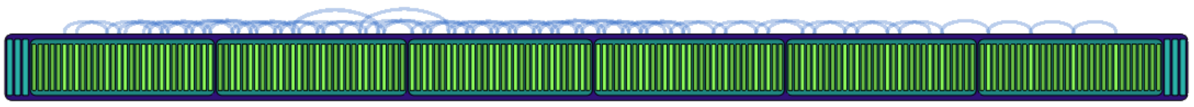


**Figure 5.24:** Display of a larger number of recurring event. Trends betweens busier and quieter months can be seen.

When few recurring events are displayed, the visualization provides a fast overview of start and end times and possible exceptions to the type of recurrence such as a rescheduled or a missed appointment due to a holiday. When a large number of recurring events is displayed at a time, some extensions to the visualization might be necessary. For example, all connected arcs for a recurring event could be colored in a special highlight color once the user selected one of the events. Instead of a special color a filter mechanism could be introduced that displays arcs for recurring events based on the choice of the user. A filter could be placed on the visualization based on the type of recurrence, the start, or end date, etc.

### 5.1.4 Study Program Organization

The following example involves the visualization of the composition of a study program. The task is to visualize that the chosen program of study involves lectures in many different research areas. As an example the composition of one student's lectures and seminars taken in Magdeburg's Computational Visualistics program before the completion of the final Diplom thesis has been chosen. In general the student has to take courses in the following areas:

- **Computer Science**: Courses are offered by the Faculty of Computer Science. Different departments such as the Department of Simulation and Graphics, Data and Business Information Systems, etc. offer courses in different areas. A number of different courses have to be chosen.
- **Mathematics:** Courses are offered by the Faculty of Mathematics including calculus, linear algebra, numerical analysis, and geometry. Special courses are required.
- **General Visualistics**: Courses are offered by different faculties. Subjects include psychology, philosophy, education, or design. Different subjects have to be chosen.

- ○ **Applied Subject**: Different subjects are offered by different faculties including medicine, image engineering, or material science. One subject has to be chosen.

A tree structure can be built from this study system as seen in Figure 5.25. The first level of the tree is formed by the four main research areas a student has to cover. The second level represents all faculties that offer courses in the respective research area followed by departments and lectures or seminars a student can take. This hierarchical structure should show that a large number of faculties and topics are covered by students in the computational visualistics program.

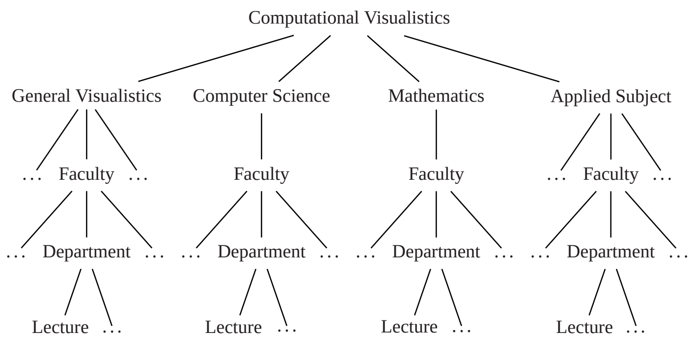


**Figure 5.25:** Basic tree structure of the Computational Visualistics program.

Relations can be introduced on this tree depending on the subjects a particular student chose in the course of his or her studies. The incidence function defines the connection for a relation $r \in R$ to the tree: $\phi : R \rightarrow N \times R$; $\phi(r) = (u, v)$ with $r \in R$; $u, v \in N$ and without loss of generality $u$ is a node on Level 1 representing one of the four research areas and $v$ is a course taken in the respective research area by the student.

Figure 5.26 gives an overview of the Computational Visualistics program. Relations have been loaded from one student's schedule. The size of the nodes is defined by a node metric dependent on the number of relations ending or starting at a node. From the size of the nodes and the buttons for hidden relations the user of the visualization can infer in which research areas the student took more lectures or seminars in comparison to the other areas.

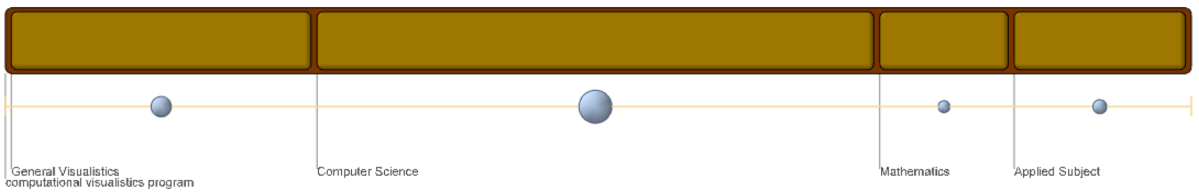


**Figure 5.26:** An overview of the Computational Visualistics program. The tree metric used for the display depends on the number of relations starting or ending at a node.

Zooming two levels into each research area results in a visualization as seen in Figure 5.27. The visualization shows departments that offer courses in the respective research area. In the “General Visualistics” area the user calls detail on demand on a larger node that seems to represent a department at which the student took several courses. The size of the relation in this case is a better indication to the number of courses a student took than the node size. Although the number of relations incident at a node has been chosen as a node metric the size of the nodes can only be compared to the sizes of their siblings. The explanation for this characteristic was given in Section 3.4.2. An example is given in Figure 5.27. Detail on demand is called for the Department of Political Science which is connected by the widest of the arcs in the “General Visualistics” area. Three faculties to the right another node has a similar size but is connected with a smaller arc. This is due to the fact that the Department of Political Science has many siblings and has to share the display space available from its parent. The other seemingly equally sized node has no siblings and can take up all available display space its parent node provides.

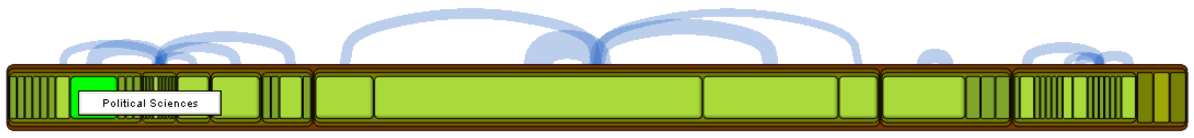


**Figure 5.27:** An overview of the Computational Visualistics program. Relations show courses taken in a specific area.

The user of the visualization is now interested in finding out more about the subjects taken in the “Computer Science” area. He or she chooses focus+context interaction and places a focus on the “Faculty of Computer Science” causing all other faculties to collapse. The node metric is changed to calculate node sizes from the number of visible descendants of a node to make nodes occupy more space if they have more visible descendants to display. The resulting presentation can be viewed in Figure 5.28. The

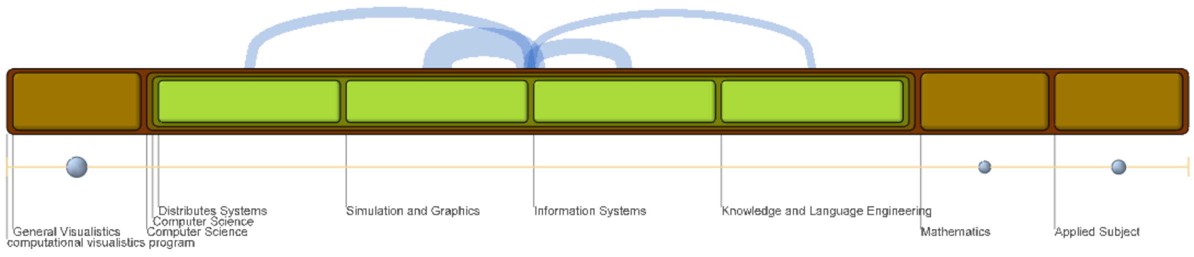


**Figure 5.28:** The focus is placed on the Computer Science Faculty causing all other faculties to collapse.

viewer realizes that the student took most courses at the Department of Simulation and Graphics. He or she chooses to find out more about the offered courses and selects the arc connecting to the department of interest. The visualization created after this interaction is shown in Figure 5.29. The user can now go through the opened lectures and seminars and call detail on demand to view the full titles.

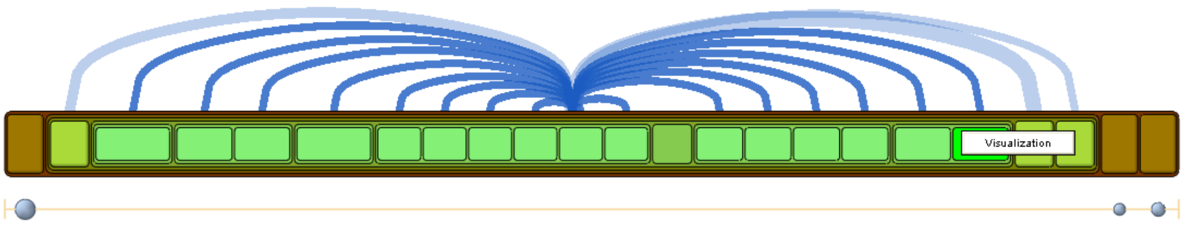


**Figure 5.29:** The user selected the arc connecting to the Department of Simulation an Graphics in the previous image.

The visualization of this data could be presented online for students to get a better overview of which courses they might take in the different research areas. It might also be interesting for the assessment of a student's academic background if a user is unfamiliar with the visualized study program. The visualization was created to show a seemingly different relational layout. All relations start at one of four lower level branches which leads to a fan-like layout.

The discussed case studies have been evaluated without questioning potential users of the visualization tool. How a formal evaluation of the *arc tree visualization* can be performed is the topic of the following section.

## 5.2 Evaluation

> *"Evaluate: to determine the significance, worth, or condition of — usually by careful appraisal and study."* (MERRIAM-WEBSTER, 2004)

Evaluating how well a system performs is a very common task in many areas of computer science. The areas providing most relevant information on an evaluation of the *arc tree visualization* are human-computer interaction and information visualization. A brief introduction to evaluation in both areas will be given in this section. These descriptions are followed by an assessment on how evaluation of the *arc tree visualization* can be performed.

### 5.2.1 Evaluation in HCI

In the field of human-computer interaction (HCI) evaluation is defined as follows:

*"Evaluation is concerned with gathering information about the usability or potential usability of a system in order either to improve features within an interface and its supporting material or to assess a completed interface."* (PREECE, 1993)

The field of HCI is concerned with how people use computer systems and how systems can be improved to better meet the users' needs. Studies in HCI involve four components: the user, his or her task or job, the context of the job, and the computer system used (PREECE, 1993). One main aspect of research in HCI is the assessment of user interfaces through which users interact with a computer system to perform a certain tasks. PREECE et al. (1996) introduce four main reasons to perform evaluation in human computer interaction:

1. **Understanding the real world:** Learn how a system is used in a real work environment.
2. **Comparing designs:** Test design ideas in the creation of a system to integrate it in the final product.
3. **Engineering towards a target:** Ensure that a target that was set in the creation of a computer system is met (e. g., a system is at least as good as another one).
4. **Checking conformance to a standard:** Test a computer system against standards that were set in the creation of the product.

### 5.2.2 Evaluation in Information Visualization

In contrast to HCI, evaluation in information visualization is a research area that has just recently begun to evolve. Since information visualizations now find their way into an increasing number of commercial computer systems new evaluation methods specifically tailored to assessing the value of visualization are needed. PLAISANT (2004) identifies a special need to present further evidence of measurable benefits of visualizations to encourage their wide-spread adoption. According to CHEN and CZERWINSKI (2000) urgently needed are improved methods in areas such as task analysis, usability evaluation, and usage analysis.

PLAISANT (2004) introduces three reasons for performing evaluation in information visualization and their thematic areas:

1. **Comparing designs:** Design elements of one or more visualizations or two or more visualization tools are compared through *controlled experiments*. The latter case is the most common type of study in information visualization.

2. **Test usability:** A visualization tool is tested with potential users through a *usability evaluation*. The goal is to find potential problems of a tool and to identify possibilities for refinement.
3. **Understanding the real world:** Test a visualization tool in realistic settings through *case studies*. This is the least common study in information visualization.

A connection between evaluation in HCI and information visualization is introduced through the increasing integration of visualizations in common computer tools. According to MORSE et al. (2000), tests of visualizations are rarely undertaken except as a part of an overall usability testing of a system in which a visualization is integrated. This type of evaluation is an important research topic in HCI.

Before a visualization tool can be tested for its overall performance in a task, usability testing should be performed on visual elements and metaphors upon which the design of a visualization was built. Such studies can help to better understand the strengths and weaknesses of a tool or the tasks for which it is most appropriate. Furthermore, possible improvements might be discovered. If done early enough in the design process evaluation can point to aspects of a visualization where efforts should be focused. It might even point to new directions for interesting and useful research.

### 5.2.3 Evaluation of the Arc Tree Visualization

To assess possible evaluation methods for the *arc tree visualization* the stage of the visualization in the design life-cycle has to be considered. Information visualization designers are encouraged to design tools with a global scope that can be used with a variety of data types and applied in many application domains. According to PLAISANT (2004), this generality forms an impairment for a task-specific evaluation of information visualization. Possible adopters of a visualization are more interested in an evaluation of a system that meets their specific needs. The *arc tree visualization* is currently not restricted to a specific application domain, data, or task. It was also designed to be generic in nature. Therefore, experiments involving specific tasks or data sets are not the evaluation method of choice at this design stage. The evaluation of the *arc tree visualization* should rather be confined to testing the overall design choices made in the creation of the tool. Such an evaluation could test design choices, affordances, and interaction metaphors described in the previous chapters.

In particular, the following aspects of the *arc tree visualization* should be the subject of an evaluation:

- **Tree Layout:** The design of the nested tree structure of the visualization should be evaluated. Previous studies of tree layouts using containment as an encoding technique have already identified potential advantages and problems which have been discussed in Section 3.4.2. The tree layout was chosen deliberately due to the advantageous use of screen real estate to be able to use the *arc tree visualization* as a modular component of information displays. Also, the tree layout allows the display of relations with minimal edge-node intersections. An evaluation could test if these deliberate choices are justifiable, i. e., if another tree layout with a display of direct connections might be easier to understand. Such an evaluation could be performed by having users interpret and compare different designs. Features of the nested tree layout might also be tested. For instance, an appropriate offset size needs to be identified that enables a good display of the tree structure, large enough sizes for the display of nodes, and an adequate size for selecting nodes through a mouse click.

◦ **Relation Layout:** The basic relation layout is based on a previous study showing that connectedness can be a more powerful grouping principle than proximity, color, size, or shape (PALMER and ROCK, 1994). This finding should be confirmed by a user study. The display of relations could be compared to a display of relations in the same tree using color coding.

◦ **Visualization of Nodes:** Emphasis should be placed on assessing if the intended affordance in the visualization of nodes through the use of a 3D-depth cue is achieved. Users could be asked to identify expandable and non-expandable nodes in a visualization. Another possibility for testing the visualization of nodes could involve comparisons between different designs. Was the chosen node shape ideal or would a circular, oval, or rectangular shape be more appropriate?

◦ **Visualization of Relations:** The encoding of indirect and direct relations could be assessed to find out if the interpretation of transparent and opaque relations is straightforward or leads to interpretation errors. This encoding could be compared to another design choice using color to encode the two types of relations (e. g., gray for indirect relations and a color for direct relations). The same type of evaluation should be performed for the encoding of hidden relations. The questions to be answered are: Is the interpretation of the created circular buttons straightforward or do users interpret these buttons in a way previously unthought of? Does the display of these buttons create the intended affordance of being selectable?

◦ **Focus+Context Presentation:** The presented focus+context technique is novel in that it takes relations in the data into account. One of the main assumptions in the creation of the focus+context display could be tested: Are nodes connected from the focus area really "important" nodes? Or would a regular focus+context display be sufficient or even more appropriate?

- **Interaction:** One of the requirements for interaction in a visualization discussed in Section 3.2 is a low cognitive load on the user. Smooth transitions and no abrupt layout changes are necessary for a user to keep a mental model of the data. However, since the implementation of smooth transition between different views is a topic for future work on the *arc tree visualization* this evaluation should be delayed until smooth transitions are integrated.

## 5.3 Summary

This chapter introduced four case studies for the application of the *arc tree visualization* in real world applications and tasks. For each of the tasks the strengths and possible weaknesses of the *arc tree visualization* were described. The assessment of the visualization was extended by potential areas for further development of the tool. The shown case studies only include a small set of the possible application domains for the *arc tree visualization*. Other application domains have to be extrapolated from the shown examples. Further application areas for the *arc tree visualization* include project planning, especially the visualization of critical paths, visualization of relations in taxonomies and classifications, or in bibliographies. The last section assessed possibilities for a user-centered evaluation of the *arc tree visualization*. Areas were highlighted where an evaluation would be most appropriate at this stage of the development of the visualization.

CHAPTER 6

# Conclusion

Creating effective visualizations poses many problems including an appropriate representation of the data to be displayed, a meaningful encoding of attributes of the data, and the need for offering intuitive interaction techniques to explore the data. This chapter summarizes how this thesis attempts to solve these problems. The development process is briefly outlined followed by a short description of the main aspects considered in the creation of the visualization. In addition, possible directions for future work in the realm of the presented visualization are discussed in the last section.

## 6.1 Summary of Contributions

This thesis deals with the creation of an effective information display. It presents a novel visualization for the display of relations in hierarchical data structures. Many of the problems in the display of direct links on tree structures can be solved with the developed visualization technique. A particularly novel aspect is the consideration of relations in the development of focus+context navigation and presentation.

The discussion of visualization techniques (in particular graphical variables, Gestalt laws, and the visual information seeking mantra) and the analysis of related work in Chapter 2 forms the basis for the development of the visualization in Chapter 3. In the design of the visualization, encodings for the hierarchical tree structure and relations between data items were of primary concern. Chapter 4 enriched the developed visualization with methods for an interactive exploration of the data. The created visualization was subsequently analyzed with respect to possible application domains in Chapter 5 which also introduced possible areas for evaluation of the presented visualization tool.

In the development of the visualization the following points were emphasized:

- The display of **relations** using direct links since connection has been shown to be a strong grouping principle,
- Visualization of a hierarchical structure for **ordered trees**,
- Effective use of **screen real estate**,
- Providing possibilities for a **global overview** of the data,
- **Interaction mechanisms** for the exploration of the data set including options for focus+context navigation, and
- Providing **affordances** and **encodings** for attributes of the data to make the interface more intuitive to use and understand.

For the visualization of tree structures a nested tree layout was chosen to be able to restrict the visualization in space. The display of relations was solved using arc-shaped glyphs which can be adapted in height and shape. An effective use of screen real estate is achieved through these layout choices. Interaction possibilities include zoom and filter mechanisms for tree content. Automation for interaction on the tree is offered through user-centered focus+context navigation. Affordances and encodings for attributes of the data were carefully chosen. How and if these choices are successful should be further tested with user studies. Several other extension possibilities for the *arc tree visualization* will be discussed in the following.

## 6.2 Future Work

This section describes possibilities for future work on the *arc tree visualization*. These have not been implemented in the current version of the program but will provide valuable contributions once integrated.

### 6.2.1 Visualization

The visualization of relations can be extended by providing information on the *direction* of a relation. In some application domains the direction of a relation might be important to know. Encoding possibilities include placing an icon indicating direction on top of the arc or showing the direction through shading as seen in Figure 6.1. Integrating this encoding in the *arc tree visualization* would make its display of relations more powerful than most other commonly used approaches. The use of color, texture, shape, etc. cannot visualize directions of binary relations.

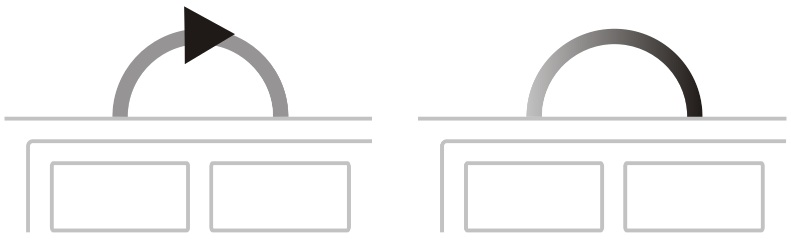

(a) Encoding through an icon. (b) Encoding through shading.

**Figure 6.1:** Encoding possibilities for the direction of a relation.

Another extension for the visualization of relations involves the integration of several types of relations in one display. As seen in the calendar example presented in Chapter 5.1, if a larger set of different relations is presented it might be advantageous to distinguish them by color.

The presented case studies also showed the need for a more sophisticated label layout. Labels can easily overlap if they become too large or too numerous. An improved label layout includes the display of abbreviated labels if nodes become too small, or the integration of focus+context interaction and degree of interest values with label layout. Another possible label layout involves the display of vertical labels which would also lead to less occlusion in the display.

Filter mechanisms for relations could also augment the visualization. Relations could be filtered depending on choices made by the user (e. g., display relations which connect to one certain node only, or those which start or end at a specific level of the tree).

### 6.2.2 Interaction

The implementation of smooth transitions is an important point for future work on the *arc tree visualization* but was not of primary concern in the creation of interaction and visualization techniques. It can, however, be integrated in the visualization with a fair amount of programming effort.

Finally, detail-on-demand could not only be included for nodes but also for relations if such information is made available. As for nodes, tooltips could present additional information for relations if called for. A prototypical implementation was presented in Figure 5.16.

### 6.2.3 Evaluation

Before the visualization is further developed in the context of a special application domain, user studies as suggested in Section 5.2 should be carried out. These might give insight into possible problems of the current visualization and point to areas which need further development. If the visualization is to be used for a special application domain further user studies should identify the exact tasks users need to carry out with the visualization. Then, an adjustment of the visualization should be performed to make the *arc tree visualization* an ideal information display for a specific task in a certain application domain.

# Bibliography

All England Lawn Tennis Club AELTC and IBM Corp. The championships, wimbledon 2004 - grand slam tennis - official site by ibm - information. Website: `http://www.wimbledon.org/en_GB/about/schedule.html`, 2004.

Keith Andrews and Helmut Heidegger. Information Slices: Visualizing and Exploring Large Hierarchies using Cascading, Semi-Circular Discs. In Graham Wills and John Dill, editors, *Proceedings of the IEEE Symposium on Information Visualization 1998 (INFOVIS'98)*, pages 9–12, Los Alamitos, CA, USA, 1998. IEEE Press.

Ragnar Bade, Stefan Schlechtweg, and Silvia Miksch. Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI'04*, pages 105–112, New York, NY, USA, 2004. ACM Press.

Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ , USA, 1999.

Luc Beaudoin, Marc-Antoine Parent, and Louis C. Vroomen. Cheops: A Compact Explorer For Complex Hierarchies. In *Proceedings of the 7th Conference on Visualization '96*, pages 87–92, Los Alamitos, CA, USA, 1996. IEEE Press.

Brent Berlin and Paul Kay. *Basic Color Terms: Their Universality in Evolution*. University of California Press, Berkeley, USA, 1969.

Jacques Bertin. *Semiology of Graphics: Diagrams Networks Maps*. The University of Wisconsin Press, Madison, WI, USA, 1983. Translation of: Sémiologie graphique, 1918.

Mark Bruls, Kees Huizing, and Jarke van Wijk. Squarified Treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization (TCVG 2000)*, pages 33–42, Los Alamitos, CA, USA, 2000. IEEE Press.

STUART K. CARD, JOCK D. MACKINLAY, and BEN SHNEIDERMAN. *Readings in Information Visualization: Using Vision to Think*. The Morgan Kaufmann series in interactive technologies. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999.

STUART K. CARD and DAVID NATION. Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. In MARIA DE MARSICO, STEFANO LEVIALDI, and EMANUELE PANIZZI, editors, *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 231–245, New York, NY, USA, 2002. ACM Press.

M. SHEELAGH T. CARPENDALE, DAVID J. COWPERTHWAITE, and F. DAVID FRACCHIA. Making Distortions Comprehensible. In *IEEE Symposium on Visual Languages*, pages 42–51, Los Alamitos, CA, USA, 1997. IEEE Press.

M. SHEELAGH T. CARPENDALE and CATHERINE MONTAGNESE. A Framework for Unifying Presentation Space. In *Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pages 61–70, New York, NY, USA, 2001. ACM Press.

GAVIN MAC CARTHY. Images of Greek Gods and Goddesses–Family Tree of Deities. Web-page `http://www.holycross.edu/departments/classics/jhamilton/mythology/images.html`, 1997. Web-page visited on 04/20/2004, last updated in December 1997.

GEORGE M. CHAIKIN. An Algorithm for High Speed Curve Generation. *Computer Graphics and Image Processing*, 3(12):346–349, 1974.

CHAOMEI CHEN and MARY P. CZERWINSKI. Empirical Evaluation of Information Visualizations: An Introduction. *International Journal on Human-Computer-Studies*, 53 (5):631–635, 2000.

RICHARD CHIMERA. Value Bars: An Information Visualization and Navigation Tool for Multi-Attribute Listings. In PENNY BAUERSFELD, JOHN BENNETT, and GENE LYNCH, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'92*, pages 293–294, New York, NY, USA, 1992. ACM Press.

MEI CHUAH. Dynamic Aggregation with Circular Visual Designs. In GRAHAM WILLS and JOHN DILL, editors, *Proceedings of the IEEE Symposium on Information Visualization 1998 (INFOVIS'98)*, pages 35–43, Los Alamitos, CA, USA, 1998. IEEE Press.

WILLIAM S. CLEVELAND and ROBERT MCGILL. Graphical Perception: Theory, Experimentation, and Application to the development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

JAMES W. COOPER. *The Design Patterns Java Companion*. Addison-Wesley Design Patterns Series. Addison-Wesley, electronic version edition, 1998.

FRANCISCO DE SALINAS. *De musica libri septem*. Mathias Gastius, Salamanca, Spain, 1577. Reprint M.S. Kastner (ed.), Documenta Musicologica I no. 13, Bärenreiter, Kassel, 1958, Found online at `http://www.xs4all.nl/~huygensf/salinas.html`.

STEPHEN G. EICK, JOSEPH L. STEFFEN, and ERIC E. SUMNER. Seesoft – A Tool For Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, 1992.

JEAN-DANIEL FEKETE, DAVID WANG, NIEM DANG, ALEKS ARIS, and CATHERINE PLAISANT. Overlaying Graph Links on Treemaps. In *Proceedings of the IEEE Symposium on Information Visualization 2003 (INFOVIS'03) Poster Compendium*, pages 82–83, Los Alamitos, CA, USA, 2003. IEEE.

JAMES D. FOLEY, ANDRIES VAN DAM, STEVEN K. FEINER, and JOHN F. HUGHES. *Computer Graphics: Principles and Practice*. Addison-Wesley Systems Programming Series. Addison Wesley Publishing Company, Inc., Boston, San Francisco, New York, USA, 18th printing edition, 1990.

ERIC FREEMAN and SCOTT FERTIG. Lifestreams: Organizing your Electronic Life. In *AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval,* pages 38–44, Menlo Park, CA, USA, 1995. AAAI Press.

GEORGE W. FURNAS. The FISHEYE View: A New Look at Structured Files. Technical Memorandum 81-11221-9, Bell Laboratories, Murray Hill, NJ, USA, 1981. Published in Readings in Information Visualization: Using Vision to Think by Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman.

GEORGE W. FURNAS. Generalized Fisheye Views. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI'86*, pages 16–23, New York, NY, USA, 1986. ACM Press.

CHARLES WATERHOUSE GOODYEAR. *Bogalusa Story*. Wm. J. Keller Inc., Buffalo, NY, USA, 1950. found online at `http://freepages.genealogy.rootsweb.com/~mcclendon/Bogalusa/Bogalusa%20Story/`.

JONATHAN L. GROSS and JAY YELLEN. *Graph Theory and its Applications*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, London, New York, Washington D.C., 1999.

CARL GUTWIN and CHRIS FEDAK. A Comparison of Fisheye Lenses for Interactive Layout Tasks. In WOLFGANG HEIDRICH and RAVIN BALAKRISHNAN, editors, *Proceedings of Graphics Interface,* pages 213–220, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

ERNST HAECKEL. *Natürliche Schöpfungsgeschichte*. G. Reimer, Berlin, Germany, 1868. found online at: `http://caliban.mpiz-koeln.mpg.de/~stueber/haeckel/natuerliche/index.html`.

DAVID HAREL and YEHUDA KOREN. Drawing Graphs with Non-Uniform Vertices. In *Proceedings of the Working Conference on Advances Visual Interfaces (AVI'2002)*, pages 157–166, New York, NY, USA, 2002. ACM Press.

KNUT HARTMANN, KAMRAN ALI, and THOMAS STROTHOTTE. Floating Labels: Applying Dynamic Potential Fields for Label Layout. In ANDREAS BUTZ, ANTONIO KRÜGER, and PATRICK OLIVIER, editors, *International Symposium on Smart Graphics*, pages 101–103, Berlin, Germany, 2004. Springer Verlag.

CHRISTOPHER G. HEALEY. Choosing Effective Colours for Data Visualization. In *Proceedings of the IEEE Conference on Visualization*, pages 263–270, Los Alamitos, CA, USA, 1996. IEEE Press.

CHRISTOPHER G. HEALEY, KELLOGG S. BOOTH, and JAMES T. ENNS. High-Speed Visual Estimation Using Preattentive Processing. *ACM Transactions on Computer-Human Interaction*, 3(2):107–135, 1996.

MARTI A. HEARST. Tilebars: Visualization of Term Distribution Information in Full Text Information Access. In IRVIN R. KATZ, ROBERT MACK, and LINN MARKS, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'95*, pages 59–66, New York, NY, USA, 1995. ACM Press.

IVAN HERMAN, GUY MELANÇON, and M. SCOTT MARSHALL. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

MAO LIN HUANG. Information Visualization of Attributed Relational Data. In *Australian Symposium on Information Visualization*, pages 143–149, Canberra, Australia, 2001. Australian Computer Society, Inc.

DEAN F. JERDING and JOHN T. STASKO. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):257–271, 1998.

BRIAN JOHNSON and BEN SHNEIDERMAN. Treemaps: A Space-filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of IEEE Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Press.

BERNARD J. KERR. THREAD ARCS: An Email Thread Visualization. Technical Report RC22850, IBM Research, Cambridge, MA, USA, 2003.

DONALD ERVIN KNUTH. *The Art of Computer Programming*, volume 1 - Fundamental Algorithms. Addison-Wesley, Reading, MA, USA and others, 3rd edition, 1997.

YING K. LEUNG and MARK D. APPERLEY. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, pages 126–160, New York, NY, USA, 1994. ACM Press.

HAIM LEVKOWITZ and GABOR T. HERMAN. Color Scales for Image Data. *IEEE Computer Graphics and Applications*, 12(1):72–80, 1992.

JOCK D. MACKINLAY. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

JOCK D. MACKLINAY, GEORGE G. ROBERTSON, and STUART K. CARD. The Perspective Wall: Detail and Context Smoothly Integrated. In SCOTT P. ROBERTSON, GARY M. OLSON, and JUDITH S. OLSON, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'91*, pages 173–179, New York, NY, USA, 1991. ACM Press.

INC MERRIAM-WEBSTER. Dictionary Search. Website, 2004. `http://www.m-w.com`.

EMILE MORSE, MICHAEL LEWIS, and KAI A. OLSEN. Evaluating Visualizations: Using a Taxonomic Guide. *International Journal of Human-Computer Studies*, 53(5):637–662, 2000.

TAMARA MUNZNER. H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space. In *Proceedings of the IEEE Symposium on Information Visualization 1997 (INFOVIS'97)*, pages 2–10, Los Alamitos, CA, USA, 1997. IEEE Press.

TAMARA MUNZNER, FRANÇOIS GUIMBRETIÈRE, SERDAR TASIRAN, LI ZHANG, and YUNHONG ZHOU. TreeJuxtaposer: Scalable Tree Comparison Using Focus+Context with Guaranteed Visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003.

DAVID A. NATION. WebTOC: A Tool to Visualize and Quantify Web Sites using a Hierarchical Table of Contents Browser. In CLARE-MARIE KARAT, ARNOLD LUND, JOËLLE COUTAZ, and JOHN KARAT, editors, *Conference Summary on Human Factors in Computing Systems, CHI'98*, pages 185–186, New York, NY, USA, 1998. ACM Press.

EMANUEL G. NOIK. A Space of Presentation Emphasis Techniques for Visualizing Graphs. In WAYNE A. DAVIS and BARYY JOE, editors, *Proceedings of Graphics Interface'94*, pages 225–234, Summerland, BC, Canada, 1994. Canadian Human-Computer Communications Society.

DONALD A. NORMAN. *The Psychology of Everyday Things*. Basic Books, New York, NY, USA, 1988.

STEPHEN PALMER and IRVIN ROCK. Rethinking Perceptual Organization: The Role of Uniform Connectedness. *Psychonomic Bulletin & Review*, 1(1):29–55, 1994.

CATHERINE PLAISANT. The Challenge of Information Visualization Evaluation. In MARIA FRANCESCA COSTABILE, editor, *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 109–116, New York, NY, USA, 2004. ACM Press.

Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proceedings of the IEEE Symposium on Information Visualization 2002 (INFOVIS'02)*, pages 57–64, Los Alamitos, CA, USA, 2002. IEEE Press.

Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. LifeLines: Visualizing Personal Histories. In Michael J. Tauber, editor, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'96*, pages 221–227, New York, NY, USA, 1996. ACM Press.

Jenny Preece, editor. *A Guide to Usability*. Addison-Wesley, Wokingham, England, Reading, MA, USA, and others, 1993.

Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. *Human-Computer Interaction*. Addison-Wesley, Harlow, England, Reading, MA, USA and others, reprinted edition, 1996.

Bernhard Preim. *Entwicklung interaktiver Systeme*. Springer-Verlag, Berlin, Heidelberg, Germany, 1999.

Matthias Puhle, editor. *Otto der Grosse – Magdeburg und Europa, Band I Essays*, Mainz am Rhein, Germany, 2001. Verlag Philipp von Zabern.

Helen C. Purchase. Which Aesthetic has the Greatest Effect on Human Understanding? In *Proceedings of the Symposium on Graph Drawing GD'97*, volume 1353 of Lecture Notes in Computer Science, pages 248–261, Berlin, Germany, 1998. Springer-Verlag.

Ramana Rao and Stuart K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. In Beth Adelson, Susan Dumais, and Judith Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'94*, pages 318–322, New York, NY, USA, 1994. ACM Press.

Edward M. Reingold and John S. Tilford. Tidier Drawing of Trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981.

Richard F. Riesenfeld. On Chaikin's Algorithm. *IEEE Computer Graphics and Applications*, 4(3):304–310, 1975.

George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In Scott P. Robertson, Gary M. Olson, and Judith S. Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'91*, pages 189–194, New York, NY, USA, 1991. ACM Press.

MANOJIT SARKAR and MARC H. BROWN. Graphical Fisheye Views of Graphs. In PENNY BAUERSFELD, JOHN BENNETT, and GENE LYNCH, editors, *Proceedings of the Conference on Human Factors in Computing Systems, CHI'92*, pages 83–91, New York, NY, USA, 1992. ACM Press.

STEFAN SCHLECHTWEG, PETRA SCHULZE-WOLLGAST, and HEIDRUN SCHUMANN. Interactive Treemaps With Detail on Demand to Support Information Search in Documents. In OLIVER DEUSSEN, CHARLES HANSEN, DANIEL A. KEIM, and DIETMAR SAUPE, editors, *Data Visualization 2004. Eurographics/IEEE TCVG Visualization Symposium Proceedings*, pages 121–128, Air-la-Ville, Switzerland, 2004. Eurographics Association.

HEIDRUN SCHUMANN. Informationsvisualisierung & Visuelles Data Mining. Lecture Notes, 2003.

HEIDRUN SCHUMANN and WOLFGANG MÜLLER. *Visualisierung - Grundlagen und allgemeine Methode*. Springer Verlag, Berlin, Heidelberg, Germany, 2000.

BEN SHNEIDERMAN. Direct Manipulation - A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

BEN SHNEIDERMAN. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

BEN SHNEIDERMAN. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, Los Alamitos, CA, USA, 1996. IEEE Press.

ROBERT SPENCE. *Information Visualization*. Addison-Wesley Publishing Company, Boston, MA, USA, 2000.

JOHN T. STASKO and EUGENE ZHANG. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *Proceedings of the IEEE Symposium on Information Visualization 2000 (INFOVIS'00)*, pages 57–65, Los Alamitos, CA, USA, 2000. IEEE Press.

STYLE MANUAL COMMITTEE THE COUNCIL OF BIOLOGY EDITORS. *Scientific Style and Format: The CBE Manual for Authors, Editors, and Publishers*. Cambridge University Press, New York, NY, USA, 6th edition, 1994.

DAVID TURO and BRIAN JOHNSON. Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation. In *Proceedings of the IEEE Conference in Visualization*, pages 124–131, Boston, MA, USA, 1992.

JARKE VAN WIJK and HUUB VAN DE WETERING. Cushion Treemaps: Visualization of Hierarchical Information. In *Proceedings of the IEEE Symposium on Information Visualization 1999 (INFOVIS'99)*, page 73, Los Alamitos, CA, USA, 1999. IEEE Press.

COLIN WARE. *Information Visualization – Perception for Design*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2000.

COLIN WARE, HELEN PURCHASE, LINDA COLPOYS, and MATTHEW MCGILL. Cognitive Measurements of Graph Aesthetics. *Information Visualization*, 1(2):103–110, 2002.

MARTIN WATTENBERG. The shape of song. Website `http://www.turbulence.org/Works/song/mono.html`, 2001. Website launched 2001, visited May 2004.

MARTIN WATTENBERG. Arc Diagrams: Visualizing Structure in Strings. Technical Report 02-11, IBM Research, Cambridge, MA, USA, 2002.

YED. yEd - JavaTM Graph Editor, yWorks GmbH. Program available at: `http://www.yworks.com/en/products_yed_about.htm`, 2004. Version used: 2.2.1.

# List of Figures

# List of Tables

APPENDIX A

# Appendix

## A.1 Tree Representations

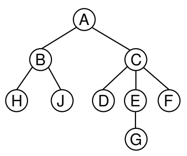
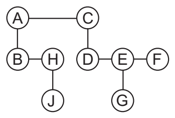
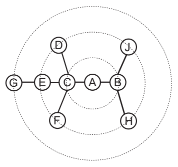
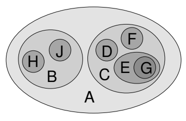
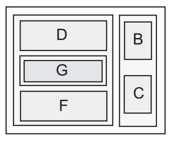
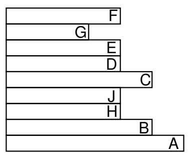
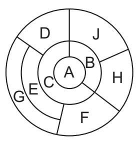
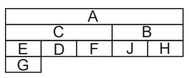
| Different Tree Representations | | |
|---|---|---|
|  Conventional Diagram |  Preorder Representation |  Radial Tree |
|  Venn Diagram | (A(B(H)(J))(C(D)(E(G))(F))) Nested Parentheses |  Nested Treemap |
|  Indentation |  Tree Ring |  Icicle Plot |

**Table A.1:** Conventional Tree Representations. Many examples taken from (KNUTH, 1997).

## A.2 Graph Theory Definitions

This section will introduce a selection of basic terminology used in graph theory. The given definitions follow those introduced by (Gross and Yellen, 1999).

**Definition A.1** *A* graph $G = (V, E)$ *is a mathematical structure consisting of a finite, non-empty set* $V$*, a finite set* $E$*, and an incidence function* $\varphi : E \rightarrow V \times V$*;* $\varphi(e) = (u, v)$ *with* $e \in E$ *and* $u, v \in V$*. The elements of* $V$ *are called vertices or* nodes*, and the elements of* $E$ *are called* edges*. Each edge has a set of one or two nodes associated to it that are called its* endpoints.

**Definition A.2** *A* directed edge *is an edge with one endpoint designated as the* tail *and the other endpoint designated as the* head.

**Definition A.3** *A* self-loop *is an edge that joins a single endpoint to itself.*

**Definition A.4** *A* multi-edge *is a collection of two or more edges having identical endpoints.*

**Definition A.5** *A* directed graph *(or* digraph*) is a graph with all its edges being directed.*

**Definition A.6** *The underlying graph of a directed graph G is the graph that results from removing all edge directions of G.*

**Definition A.7** *A graph or digraph is* simple *if it has neither self-loops nor multi-edges.*

**Definition A.8** *In a graph, a* walk *from vertex* $v_0$ *to vertex* $v_n$ *is an alternating sequence* $W = < v_0, e_1, v_1, e_2, \dots, v_{n-1}, e_n, v_n >$ *of vertices and edges such that* $\varphi(e_i) = (v_{i-1}, v_i)$ *for* $i = 1, \dots, n$.

**Definition A.9** *A graph is* connected *if for every pair of vertices* $u$ *and* $v$ *there is a walk from* $u$ *to* $v$.

**Definition A.10** *A digraph is* connected *if its underlying graph is connected.*

**Definition A.11** *A* trail *is a walk with no repeated edges.*

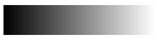
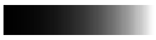
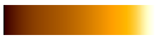
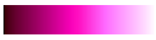
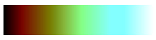
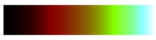
**Definition A.12** *A* path *is a trail with no repeated vertices (except possibly the initial and final vertices).*

**Definition A.13** *A walk, path, or trail is* trivial *if it has only one vertex and no edges.*

**Definition A.14** *A nontrivial closed path is called a* cycle.

## A.3 Color Scales

Color scales are commonly used to encode numerical values where each value is assigned its own color. This section introduces the color scales available in the *arc tree visualization* and their respective advantages and disadvantages.

| Color Scales for Visualization | | |
|---|---|---|
| Gray Scale |  | This scale is not considered to be a color scheme but is often used. It is simple and has a natural sense of order (from dark to bright). The disadvantage of the gray scale is its limited dynamic range (only 60 to 90 Just Noticeable Differences). |
| Linearized Gray Scale |  | This scale is perceptually linearized to make perceived distances between adjacent gray values as uniform as possible. The disadvantage is that a large amount of values is mapped to dark gray values. |
| Heated Object Scale |  | The intensity of the three primary colors rises monotonously and with the same amount of magnitude. The basis for this color scale is the fact that the human visual system is most sensitive to luminance changes in the orange-yellow hue. |
| Magenta Scale |  | The intensity of the three primary colors rises monotonously and with the same amount of magnitude. The basis for this color scale is the fact that the human visual system is most sensitive to hue changes for the magenta hue. |
| Optimal Color Scale |  | This color scale was introduced by LEVKOWITZ and HERMAN (1992). It maximizes the number of distinctly perceived colors and maintains a natural order among the colors. |
| Linearized Optimal Color Scale |  | This color scale resembles the Optimal Color Scale but is also perceptually linearized. |

*continued on next page*

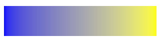
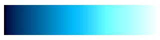
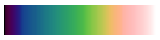
| Color Scales for Visualization | | |
|---|---|---|
| Blue to Yellow Scale |  | This scale provides a clear differentiation between negative blue values and positive yellow values. This scale has a higher perceived dynamic range (measured in Just Noticeable Differences) than the gray scale, thus providing better perceptual resolution. |
| Blue to Cyan Scale |  | The blue to cyan scale belongs to a class of scales each spanning 60 degrees out of the 360 degrees of the hue circle. This scale does not include red or green colors. This it avoids problems with the most common color deficiencies. |
| Rainbow Scale |  | This scale traverses colors along a path from black to white through all the colors of the rainbow. The colors are traversed at different lightness. |

**Table A.2:** Color scale options for the additional encoding of level information.

## A.4 Design Study for the Visualization of Hidden Relations

For the visualization of relations hidden in the subtree below a collapsed node several encoding techniques were considered. Figures A.1–A.4 give an overview of the techniques thought of.

Hidden relations are visualized by drawing a self-loop below the collapsed node in Figure A.1. A self-loop has the advantage that it corresponds to graph drawing conventions and its function might, therefore, be easier recognized by a first-time user. This encoding technique was not chosen since self-loops could not be easily recognized and selected when the node has a very small width.

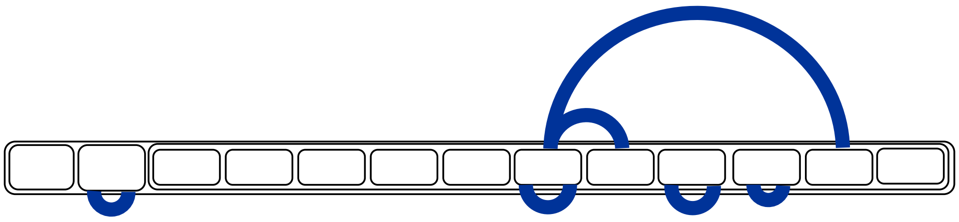


**Figure A.1:** Hidden relations visualized through a self-loop.

Hidden relations are indicated with an arc shaped glyph in Figure A.2. The arc-shaped glyph has the advantage that it resembles the already used glyph for representing relations.

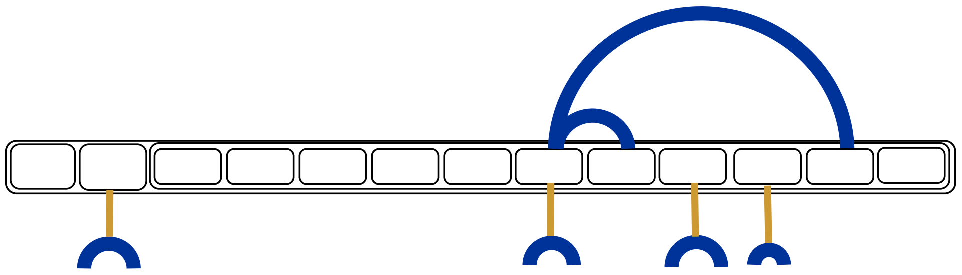


**Figure A.2:** Hidden relations are indicated with an arc shaped glyph.

Hidden relations are indicated by circular buttons in Figure A.3. This encoding technique was chosen and is further explained in Section 4.2.2.

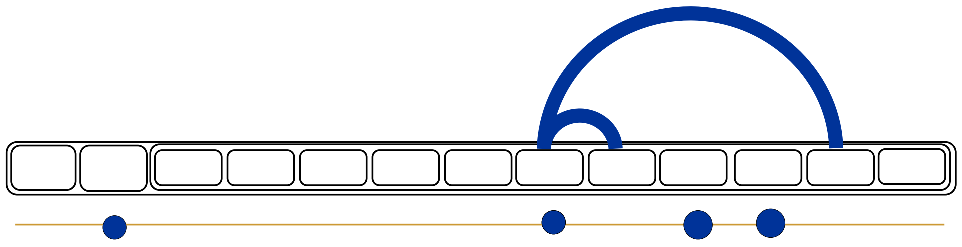


**Figure A.3:** Hidden relations are indicated by circular buttons.

Hidden relations are indicated by an arc shaped icon on the surface of the node. This encoding technique was not chosen since these icons cannot be recognized if the width of the node becomes too small.

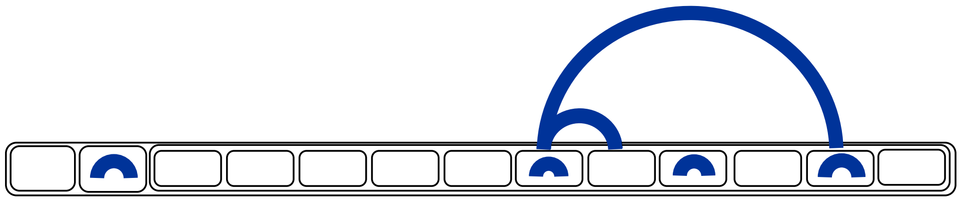


**Figure A.4:** Relations indicated by an arc shaped icon on the surface of the node.

# Independence Statement

Herewith I declare that I have completed this work solely and with only the help of the mentioned references.

Magdeburg, September 6, 2004

Petra Neumann