DOI: 10.1111/cgf.13899

COMPUTER GRAPHICS *forum*
Volume 39 (2020), number 6 pp. 5–18

ConceptGraph: A Formal Model for Interpretation and Reasoning During Visual Analysis

B. Karer, I. Scheler, H. Hagen and H. Leitte

TU Kaiserslautern, Germany
{karer, scheler}@rhrk.uni-kl.de, {hagen, leitte}@cs.uni-kl.de

Abstract

In order to discuss the kinds of reasoning a visualization supports and the conclusions that can be drawn within the analysis context, a theoretical framework is needed that enables a formal treatment of the reasoning process. Such a model needs to encompass three stages of the visualization pipeline: encoding, decoding and interpretation. The encoding details how data are transformed into a visualization and what can be seen in the visualization. The decoding explains how humans construct graphical contexts inside the depicted visualization and how they interpret them assigning meaning to displayed structures according to a formal reasoning strategy. In the presented model, we adapt and combine theories for the different steps into a unified formal framework such that the analysis process is modelled as an assignment of meaning to displayed structures according to a formal reasoning strategy. Additionally, we propose the ConceptGraph, a combined graph-based representation of the finite-state transducers resulting from the three stages, that can be used to formalize and understand the reasoning process. We apply the new model to several visualization types and investigate reasoning strategies for various tasks.

Keywords: information visualization, visualization, scientific visualization, visual analytics

ACM CCS: • **Human-centred computing** → **Visualization theory, concepts and paradigms**, • **Theory of computation** → Automata extensions, Transducers, Theory and algorithms for application domains

1. Introduction

Every insight to be obtained from a visualization is subject to the viewer's ability to perceive, recognize and interpret what is shown.

Consider the chart given in Figure 1 (left). Its interpretation will strongly depend on the observer's experience and background knowledge: A *five-year old child* may state that they see multiple lines starting from the same point and that one of the lines has beads on it. They may also recognize text that they cannot read yet. An *adult with chart literacy* may be able to tell that this is a line chart that tells how speedup changes with the number of processes. A *statistician* will comment that they observe linear dependence between the dependent and the independent variable and a *parallel programmer* will be delighted that they wrote an optimal parallel program. All observers get the syntactically same information. Nevertheless, they draw widely different conclusions from the presented figure, i.e. the semantic interpretation is very different. As stated above, this originates from the observers' varied backgrounds, experiences and analysis perspectives which we, as visualization designers, need to take into account. It is important to understand which parts of a visualization can be understood by means of the appropriate technical skills and provided information (e.g. a legend), and which aspects require domain-specific knowledge. The goal of this paper is to provide (1) a formal model that captures the information- and knowledge flow in the visual design- and analysis process and (2) a visual representation of this model that supports the analysis, understanding and communication of the reasoning process.

The first question that arises is how we can arrive at such a model that describes the interpretation and reasoning process about a visualization. First, we need to subdivide the process into the key stages. Mackinlay, Wilkinson and others [Mac86, WW10, TGOA15] explained the creation and reading of visualizations as an encoding–decoding mechanism, where the encoding maps data to its graphical representation and the decoding is the capability of viewers to properly understand what they see in the graphical display. In our model, the encoding is a visualization pipeline that maps data to a (potentially interactive) visualization. The decoding is the recognition and

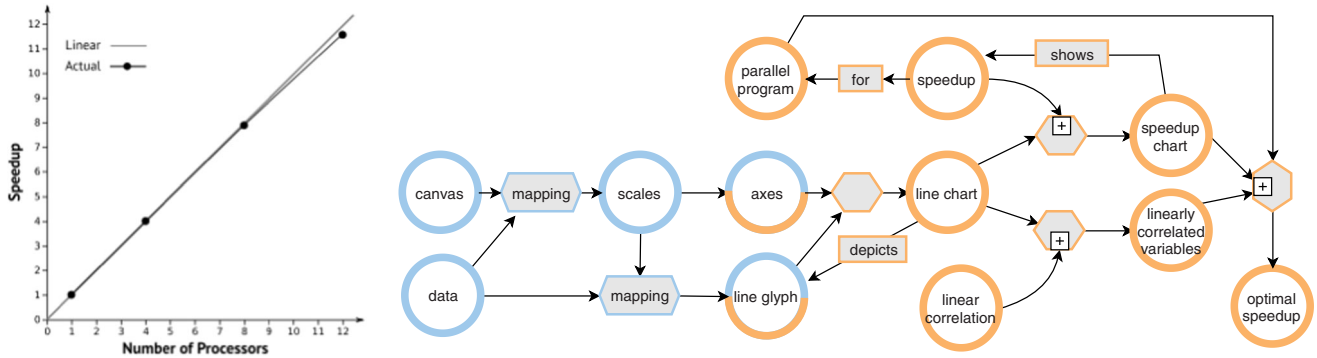

Figure 1: *Visualizing the reasoning process: (left) the interpretation of the speedup diagram strongly depends on the observer's background knowledge; (right) the ConceptGraph illustrates the information flow during visualization generation (blue nodes) and chart interpretation (orange nodes). The reasoning process is modelled using situation semantics and knowledge representations such as ontologies.*

interpretation of relevant structures within the visualization which then form the input for the reasoning strategy.

As we have seen in the introductory example of the speedup chart, a chart's interpretation is highly context sensitive. In general, this is highly difficult to model. Hence, we restrict our model for the reasoning strategy to pre-determined rules represented, for example, as an ontology or an inference network. Such knowledge for visualization interpretation is provided, for example, through the framework of data visualization literacy (DVL) [Bör16, BBG19, Wom15]. DVL provides us with a hierarchical typology of core concepts in data visualizations, their interpretation and an assessment strategy for users to quantify their DVL. In this sense, DVL provides us with a pre-determined reasoning strategy modelling inference and interpretation rules to be applied by the (literate) analyst. Such knowledge has already been integrated in several systems to design better visualization defaults and make recommendations. Examples are ShowMe [MHS07] and Draco [MWN*19]. Applying such knowledge to novel visualizations has been researched by Lee *et al.* [LKH*15], which makes our model extendable to new types of visualizations as well.

Aggregating these considerations, the formal model we propose requires three components:

1. **Encoding:** A description of the visualization's generation.
2. **Decoding:** A specification of the structures read from the visualization.
3. **Interpretation:** A model for the interpretation and deduction scheme specifying the reasoning strategy.

As we want the model to be actionable, all three processes must be executable. This means that the visualization generation model should be capable of processing actual data into a visualization and the reasoning strategy should somehow process the structures assumed to be read from the visualization in order to determine the meaning to be assigned to them in terms of semantic information. This is achieved by modelling the processing of data and its interpretation as a system of coupled automata such that the visualization, the structures being read from the display and the semantics assigned to them can be described in terms of formal languages.

Modelling the processes as the generation of a formal language, additionally allows us to assess the reasoning process' complexity and thus enables the user to identify potential optimization of the visualization such that a simplified reasoning strategy is supported.

In summary, we make the following contributions:

- We present a formal model for the reasoning process about a given data visualization.
- We detail its construction using concepts from grammar of graphics for encoding, DVL for decoding and situation semantics for the semantic analysis.
- We unify all concepts in a coherent framework and explain the modelling of the transitions between the three aspects of visual understanding.
- We present the ConceptGraph, a graph-based representation of the framework that eases its understanding, analysis and communication.

2. Related Work

As detailed in the 'Introduction', visualization interpretation is a multi-stage process: encoding, decoding and reasoning. The model we present covers the encoding–decoding interpretation aspect of this information communication problem, and many fields of research have been concerned with the various aspects of this problem. The most important ones for our work: grammars for graphics as representations of encoders (Section 2.1), low-level decoding through visual literacy and the respective knowledge representation (Section 2.2) and alternative models considering semantics in visualization (Section 2.3).

2.1. Grammars for graphics

As most visualizations nowadays are generated through automatized process or at least aided by appropriate languages that integrate hierarchical typologies and semantic interpretations of aggregate structures, we can more easily model the decoding phase. The earliest examples of such languages are R [IG96] and ggplot2 [Wic09] for statistical data analysis and chart generation that

integrate pre-designed charts that can be adjusted using parameters. A component-oriented approach was presented in the various grammars of graphics: the grammar of graphics [Wil12], a layered grammar of graphics [Wic10] and the stochastic grammar of images [ZM06]. Declarative languages such as D3 [BOH11], Vega [SRHH15] or Atom [PDFE17] extend these concepts integrating high-level concepts such as scales or axis and interactive manipulation of the visualization. Most of these languages target users with varying levels of expertise and use highly descriptive names for the concepts in order to indicate their respective semantics. Mei *et al.* [MMWC18] provide an overview of declarative systems and propose design spaces that summarize different design patterns in those languages which provide extracted knowledge for theoretical models such as ours.

We will use Vega [SRHH15] as reference graphic language in this paper to illustrate the concepts. Vega is a visualization grammar for interactive visualization designs. It is build on top of D3 and generates web-based views using the HTML canvas or SVG [Veg19]: We chose Vega as it is a widely accepted graphic language that includes many of the important concepts and gives a good level of abstraction. Examples are provided online and can be tested in the online editor [Veg19]. Limitations will be discussed later (We only look at charts so far, only things that can be shown in SVG, recomputation after each interaction, and so forth).

2.2. Semantic analysis of visualizations and influence of tasks

Petre and Green [PG93] report evidence that understanding visualization is unlikely a native ability of humans but can be learned and is supported by the DVL framework [Bör16, BBG19]. Findings like this indicate that cognitive load and other human factors might actually be the expressions of the process underlying the understanding of graphical displays and the reasoning about them. Hence, there must be a collection of mappings between the graphical display, how it is understood by a viewer, the viewer's tool set for reasoning about the presentation and the data and phenomenon being represented by the graphical display. Towards this direction, Vickers *et al.* [VFR13] propose a theoretical framework for the process of reasoning with visualizations based on category theory and semiotics. Explicitly taking into account perceptive and cognitive abilities as well as knowledge, they are capable of describing a number of effects commonly observed in visualization applications.

There is also an increasing corpus of literature on the understanding of diagrams. For example, Elzer *et al.* [EWCH04, ECD06] investigate models to capture the intended message of information graphics. Their approach determines an action plan the designer intends the viewer to follow when working with the visualization. Yet, their work is focused on perceptual aspects while we are more interested in the cognitive aspects combining several messages the graphical representation communicates into more complex conclusions. Other research concentrating on the extraction of higher level information from graphical depictions concentrates on the reading of structures like sets of points, chains of edges in node-link diagrams and the like [GWK93, MRGTBD08, TSK12]. In this work, we refer to such structures as graphical contexts. Although they play an important role in the model we propose, the focus of this work is not their identification and recognition but rather their interpretation with respect to the general investigative context the visualization is being viewed in. On the foundational level, Coppin discusses the highly important question what actually constitutes a graphical element and causes the viewer to recognize it as a carrier of meaning rather than simply another part of a picture [Cop12]. Again, such a fundamental discussion would exceed this paper's scope by far. Yet, the question what constitutes a graphical element should be kept in mind when following our discussion of how structures in visualizations are interpreted.

The second aspect that influences the semantic interpretation is the analysis task at hand. The assignment of meaning in human visual information processing is performed in short-term memory by mapping the observation to learned structures in long-term memory [Kos89]. In general, deriving global information such as trends and patterns from the data triggers different and more complex cognitive processes in the human brain and is harder than locating information that can simply be read off [GWK93]. This is already reflected in Bertin's distinction between presentation and layout and Rensink's triadic architecture for the description of visual data representations [Ber83, Ren00]. The three-level model is supported by experimental data gathered by Ratwani, Trafton and their colleagues in a series of experiments [TT01, RTBD04, TMMT02]. In a series of user studies on information visualization, they found three categories of insight to be obtained according to the difficulty users experienced in obtaining these insights. The first level is concerned with understanding the presentation itself. The second level reveals relations, patterns and trends depicted in the visualization.

The third category is the inference of additional information based on reasoning about the obtained information with the help of user knowledge external to the visualization.

From a task perspective, Nazemi and Kohlhammer associate the first category with search tasks, the second with exploration and the third with the actual analysis process [NKH*14]. A particularly interesting finding in this direction has been reported by Smuc *et al.* who found in a user study that while insight about the data could only be generated after the viewers obtained insight about the visualization, they only needed to understand those parts of the visualization they actually applied for their reasoning [SML*08].

2.3. Models for the reasoning strategy

Such structuring of visualization knowledge extraction has been an active field over several decades, and several models of visualization structure and content have been proposed in order to define the structures the reasoning is actually performed on and how the structures perceived in a visualization are interpreted.

Early research focuses on low-level processes that are close to the data. Typically, they apply algebraic, information theoretic or other formal constructions to describe visual encodings. Examples date back to Mackinlay's presentation tool [Mac86] and Wilkinson's grammar of graphics [Wil05]. More recent representatives of this category are the visual embedding of Demiralp *et al.* [DSK*14], the algebraic design process by Kindlmann and Sheidegger [KS14] and Tominski's event-based approach [Tom11]. The theory presented in this paper touches this direction by formalizing a model of information content and how this information is mapped to the graphical

elements composing the visualization. Otto and Schumann propose a model similar to the one presented in this paper, in that, it attempts to combine data wrapped into information objects [CG02]. Doleisch *et al.* develop a feature description language to interactively define features of high-dimensional data based on the user's interest [DGH03].

Modelling the reasoning strategy essentially as a collection of objects and their relations, our approach formalizes relevance by reachability in an ontology-like structure rather than by defining relevance functions. The second direction of theory is the development of general frameworks of visualization design. Typical representatives of this direction are high-level models that either aim to support the visualization expert directly by providing feedback or guidance (e.g. [AS13, SASS16, SSL*12]) or model the design process as a whole (e.g. [SSL*12, Mun09]). The theory proposed in this paper is more low level but can also be applied as a design tool. Examples for possible applications are the assessment whether a visualization supports the solution of a given task or finding possible optimizations towards more efficient reasoning.

3. Methodology

As stated earlier, the modelling of chart interpretations requires three steps: encoding, decoding and interpretation. Encoding and its visual representation are detailed in Section 3.1. Decoding and interpretation are closely coupled and are treated in Section 3.2. Section 3.3 presents the *ConceptGraph*, which is a graph-based representation of the information flow in these two processes.

3.1. Encoding: Information obtained from a visualization

Modern declarative graphics languages provide a high level of abstraction as they incorporate key concepts of visualization design. This enables us to derive the information contained in the visualization from its declarative description. Towards the incorporation of this knowledge, we need to understand the grammars of such declarative languages (Section 3.1.1), turn them into a formal automaton that has a graphical representation (Section 3.1.2) and think about an abstraction level for compound graphical objects (Section 3.1.3).

3.1.1. *Syntactic description of visualizations*

Consider for a running example Figure 2 which presents a line chart along with a compact representation of the Vega code from which it was created. We see that the Vega syntax already provides a lot of semantic information. It comprises components that define the data, scales, axes and marks. References between the entities are given in textual form, for example, marks are drawn for each data point and as multiple datasets may be specified, the interpreter needs to know which one is considered. The respective reference is provided by means of a textual entry in the 'from' field. To identify the relevant structural units in a chart and their interdependence, we need to formalize Vega's grammar. The grammar of Vega is provided through the API and we will summarize key concepts and relevant structures.

In this work, we consider a visual representation as a descriptive language, referred to as a visual language $\mathcal{L}_V$ that is commonly generated by a grammar $\mathcal{G}(\mathcal{L}_V) = (N, T_V, S_V, \overset{A}{\rightarrow})$, where N is a set of non-terminal symbols, T_V is a set of terminal symbols, S_V is a start symbol and $\overset{A}{\rightarrow}$ are production rules. The production rules for Vega in EBNF notation take the following form:

$$S_V ::= \{\text{“schema”} : \mathit{URL}, \langle \mathit{Canvas}\rangle, \langle \mathit{Data}\rangle, \langle \mathit{Scales}\rangle, \tag{1}$$

$$\langle \mathit{Projections}\rangle, \langle \mathit{Axes}\rangle, \langle \mathit{Legends}\rangle, \langle \mathit{Marks}\rangle\} \tag{2}$$

$$\mathit{Data} ::= \text{“data”} : [\|\mathit{Dataset}\|] \tag{3}$$

$$\mathit{Dataset} ::= \text{“name”} : \mathit{Name}, (\mathit{Source}|\mathit{Url}|\mathit{Values}) \tag{4}$$

$$\mathit{Marks} ::= \text{“mark”} : [\|\mathit{Mark}\|] \tag{5}$$

$$\mathit{Mark} ::= \{\mathit{Type}, \mathit{From}, \mathit{Encode}\} \tag{6}$$

$$\mathit{Type} ::= \text{“type”} : (\text{“arc”}|\text{“area”}|\text{“image”}|\text{“line”}|\ldots) \tag{7}$$

The first line (1) gives the general structure of a valid chart description. The description is enclosed by braces and starts with the specification of a schema. Optional components (marked by $\langle\cdot\rangle$) are canvas descriptions, definitions of data, scales and projections, as well as descriptions of axes, legends and marks. If the user wants to specify data (line 3), they have to start with the field name 'data:' and may provide multiple datasets (indicated by $\|\cdot\|$). Each dataset needs a name and exactly one specification—either source, url or values. A definition of a mark (line 6), in contrast, requires the specification of three properties (type, from and encode).

3.1.2. *From grammars to graph-based representations*

A grammar can be used to verify that a given text is syntactically correct, i.e. in our case, we can check if a given specification is a valid description of a visualization in Vega or not. For context-free grammars, as occurred in most programming languages, this is commonly done by converting the input to a syntax tree. The root of this tree is the start symbol of the grammar. Inner nodes are the non-terminal symbols. Leaf nodes are the terminals and tokens. Edges are defined by the production rules. The syntax tree for the first level of the line chart code is given in Figure 3. The pink edges indicate that data from a source node is used by the target node, e.g. scales need to know about the canvas' width and height. To model our data flowchart, we remove all forward edges, visually these are shortcuts to nodes, e.g. the edge between start and marks and re-layout the graph (Figure 3, right).

Vega already uses the concept of a dataflow graph to represent data transformation and information flow during the chart generation process.[1] Vega's dataflow graph, however, strongly concentrates on the internal operations from which we want to abstract. Hence, we chose the syntax tree approach on the specification file.

One aspect that is not yet modelled in the graph representation of the visualization are data processing and transformation operations. Consider the definition of a scale which describes the mapping from

[1] https://observablehq.com/@vega/how-vega-works

Figure 2: *Line chart created using Vega: the Vega code (right) describes the relevant syntactic components of a line chart (left).*

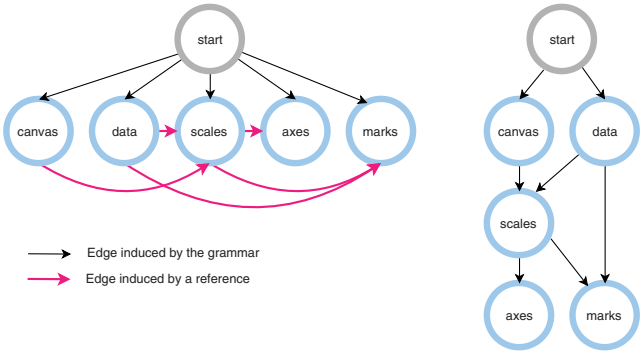

Figure 3: *Graph generation process: (left) the syntax tree for the first level of the code in Figure 2 including edges that are induced by references; (right) deleting forward edges gives the data flowchart.*

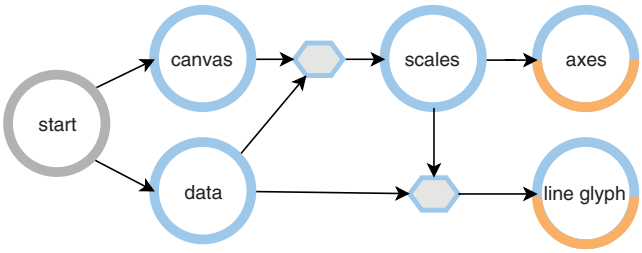

Figure 4: *Dataflow graph of the line chart: nodes represent states, arrows production rules. The start node is coloured in grey, classes and concepts in blue and blue-orange nodes encode visual elements.*

data units to image coordinates and that are used to position axes and marks in the visualization. We have already seen that the interpreter needs information from the canvas and the data in order to create the scales. From the canvas, it needs the width and/or height of the targeted visualization and from the data the ranges of the target variable. Using this information, a mapping function between the two variables (data value → image position) is computed. Similar considerations hold for marks where image point locations have to be computed. To communicate such complex computations that may be relevant for the interpretation and understanding process, we augment the flowgraph with additional glyphs for transformation and filter operations. The final dataflowgraph for the linechart

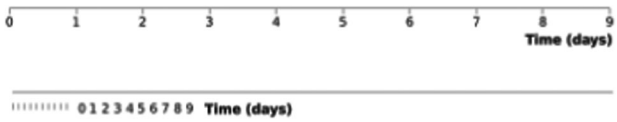

Figure 5: *Same graphical elements, different meaning: The same set of graphical elements is combined in two different ways. Top: structured spacing and annotation of tick marks indicates an axis; bottom: sorted arrangement of the same elements with no inherent meaning.*

example is given in Figure 4. A legend of structures in the flowgraph is provided in Figure 8.

3.1.3. *Graphical sentences and contexts*

The current design of the flowgraph using Vega as an input legend already provides a high level of abstraction and interpretation of the visualization. For example, we readily obtain a node for complex visual structures such as an axis or a pie chart. More difficult to handle are manually constructed aggregate structures like a boxplot that is not (yet) integrated in the language. To handle such aggregate structures or the combination of multiple views, we need to define the concepts of graphical sentences and contexts.

Aggregate structures are commonly constructed using simpler primitives such as points, lines and text. Often the combination of a set of graphical elements provides more meaning than the sum of its pieces. The meaning of a set of symbols, its semantics, has been learned in life and is often understood as common sense and no longer explained. Take, for example, the two sets of graphical primitives in Figure 5. While the upper configuration of graphical elements can be easily recognized as an axis, the lower one is simply an ordered collection of the same set of graphical primitives. In the lower case, users can recognize the sorting, but do not associate specific semantics with the set of graphical elements. In the upper case (axis), most users learned that a straight line with tick marks and associated numeric tick labels refers to a quantitative scale, and that the label close to this structure refers to the name of the encoded variable—here Time. Users can also deduce from experience that the text in braces '(days)' refers to the unit of the encoded variable. All this information is not explicitly given

Figure 6: *Traffic signs in Germany: Depending on the type of regulation and the possibility to represent it, the interpretation of the signs varies in difficulty.*

in the chart, but was learned and constitutes common chart literacy [BBG19].

In our flowgraph, we have already used such abstracted nodes that contain multiple graphical primitives and form a more complex object, namely, axes and line glyph in Figure 4. Formalizing such abstracted concepts and including them in the flowgraph is important as the goal is to concentrate on semantics and not pure data flow. In a visualization software, knowledge is explicitly encoded and commonly classes are named accordingly. This knowledge shall be represented and reused in the flowgraph.

We distinguish two types of aggregate structures that may occur: *graphical sentences* and *graphical contexts*. Graphical sentences originate from syntax, i.e. the systematic layout of the constituent pieces is defined by the visualization software and is fixed apart from aspects that can be manipulated through parameter changes. A graphical context emanates purely from the position of the graphical primitives after rendering and requires manual interpretation. Examples are clusters and outliers in scatterplots or periodic patterns in a temporal visualization. These features have not been stored in the visualization code, but need to be recognized by the viewer. Representations of graphical sentences and contexts in the flowgraph can be distinguished by colour. As the semantics of graphical sentences are included in the code, we colour the representing nodes blue. Graphical contexts require human reasoning and hence, are coloured orange.

3.2. Modelling the reasoning process

In the previous section, we discussed which information is encoded in a visualization and can be seen by an observer. The flowgraph also contains the dependency structure of the different graphical sentences that a user has to (at least in part) understand in order to understand a given visualization. In this section, we continue with the reasoning about graphical contexts and start with an example from daily life—interpretation of traffic signs in a given situation.

Figure 6 depicts multiple traffic signs that provide instructions or information to road users [Uni68]. The upper row depicts three triangular signs with a red boundary and different signs in the centre. The lower row depicts a circular sign, a give way sign and a town sign. Interpreting the signs poses varying levels of difficulty. If the observer is not familiar with traffic signs at all, they probably have a hard time interpreting them correctly. This is because some of the encoded information is literal, some is metaphoric, and some is conventional [Eng02]. Literal signs feature physical or structural similarity to the real object like the pedestrian. Metaphoric signs are based on an analogy between the sign and the real phenomenon like the windsock representing wind. Conventional signs often lack this clear connection and seem to be arbitrary like the cross for crossroad or the shape of the signs. From these interpretations, the observer may deduce that the signs are related to the respective hazards or situations (pedestrian, wind, crossroad). The precise interpretation requires another piece of information, namely, the conventions for sign shapes and colouring. According to the UN convention on Road Signs and Signals [Uni68], eight convention classes exist. Class A represents *danger warning signs* and is represented by an equilateral triangle with one corner at the top (top row of Figure 6). Class B represents *priority signs* and its representation is an upside-down triangle (give way sign, bottom row centre). *Prohibitory signs* are represented by circles (first sign in the bottom row). Combining these two pieces of information gives the observer a good access to the interpretation of road signs, even some that they may have never encountered before. For example, a triangular sign pointing upwards with a cow in the centre means 'potential hazard of cos on the road'. A cow in a circle means that one is not allowed to herd kettle on the road.

3.2.1. *Situation semantics*

Now that we have a basic understanding of the conventions in traffic signs, the question arises how we can formalize this interpretation and transfer it to a wider field of applications in visualization that may not have such rigid and well-defined interpretations. We propose the application of situation semantics [Dev06] as a formalism to encode semantic interpretation of visual information. Situation semantics has originally been introduced in the context of natural language processing where the meaning of a spoken sentence strongly depends on the speaker's context, i.e. where they are, to whom they talk, what they know about the world [BP83, Bar86]. Similarly, a viewer's capability of understanding and applying visualization depends on personal factors such as educational background and expertise, as well as data-related factors like the domain of interest or a given analysis question to be answered [BBG19, LS10]. To demonstrate the fundamental concepts, we again start with the formulation of traffic sign interpretations using situation semantics and then move on to charts.

The smallest carriers of information in situation semantics are *infons*. Infons are of the form $\ll A, \vec{X}, \pi \gg$ where A denotes a type or relationship name, $\vec{X}$ is a number of subjects and objects whose states of affairs the infon expresses and the polarity $\pi \in \{0, 1\}$ indicates whether, in the currently observed context, the expression encoded by the infon holds or not. In this paper, we use a restricted form of situation semantics and allow only infons of the forms $\ll A, x, \pi \gg$, and $\ll r, x, y, \pi \gg$. The former type encodes the information that an entity x is of some type A while the latter encodes a relation r between two entities x and y.

In this way, we can define that if certain properties for a sign x hold, this is a danger warning sign:

$$\begin{aligned}(&\text{«upward equilateral triangle, } x, 1\text{»}\Lambda\\ &\text{«white ground, } x, 1\text{»}\Lambda\\ &\text{«red boundary, } x, 1\text{»}) \rightarrow \text{«danger warning sign, } x, 1\text{».}\end{aligned}$$

What we also see in this example is that infons can be used as literals in logical predicates and functions and can be used to define more complex objects using *constraints*. They are called *constraints* because they constrain the possible combinations of objects by implementing an implication of the form

$$a \Rightarrow_c b := (\text{«}A, a, 1\text{»}\Lambda\text{«}r, a, b, 1\text{»}) \rightarrow \text{«}B, b, 1\text{»},$$

which translates into: If a is of type A, and a and b are in relation r, then b is of type B, which may represent, for example, 'a is a warning sign' and 'b is the central glyph of a', hence, 'b is something traffic participants are warned of'. If required, constraints can also be subject to a condition involving further background information. Sometimes it is useful to define alias terms allowing to refer to complex structures by a more simple notation. In the above example, the implication establishes a relation determining that whenever some object x is a warning sign, it must be an upward-pointing equilateral triangle with a red boundary on a white background. Expressing this information as an abstract type

$$\begin{aligned}\text{WarningSign} := [x|S \models &\text{«upward equilateral triangle, } x, 1\text{»}\Lambda\\ &\text{«white ground, } x, 1\text{»}\Lambda\\ &\text{«red boundary, } x, 1\text{»}]\end{aligned}$$

allows to introduce danger warning signs into a context S by simply declaring $S \models$ «WarningSign, x, 1». In situation semantics, this process is called type abstraction. Types denote the regularities or commonalities shared by objects or situations [Dev99]. Thereby, it specifies an abstract collection of features that all instances of this type have in common—much like classes in object-oriented programming. We distinguish object types (obj) and situation types (sit):

$$\begin{aligned}obj &:= [x|S \models \text{properties of } x]\\ sit &:= [S|S \models \text{object instances and relations}].\end{aligned}$$

The restriction $S \models$ means that we bind our expressions to a certain situation S determining the context the current observation resides in. A viable situation for the traffic sign case is, for example, 'Someone takes part in traffic and sees a traffic sign'. In the context of a different situation, the meaning encoded by infons can be entirely different. As an example, consider a situation 'kids are drawing their homes and families', where the red triangle instead of a traffic sign represents the roof of a house. A situation expresses the context according to which the meaning of the objects, that are observable within the situation, is determined. Therefore, it describes the states of affairs between those objects. Like object types can be obtained from objects, situations can be abstracted into *situation types*. Other than for object types, the definition of situation types is self-referential. Therefore, while an instance of an object type always requires a containing situation, instances of situation types can exist on their own. Towards a better understanding of the idea underlying the definition of situation types, consider the situation of a person driving a car along a street. One interesting feature of situation semantics is that we do not need to know the situation completely. In order to draw the conclusions we are interested in, it is enough to know the part of the situation sufficient to draw these conclusions. Empirical studies show that this is also the case for visualization [SML*08]. In fact, situation semantics assumes to never know the situation completely. This so-called *partiality* is also the reason why every infon is bound to a situational context and why situation semantics react to changes in the context. Let us now specify the situation that the driver is passing a traffic sign from the driver's perspective:

$$\text{SignPassed} := [\text{Driver}|\text{Driver} \models \text{«Sign, } x, 1\text{»}].$$

We can now model the situation of a neutral observer watching drivers passing signs:

$$\begin{aligned}\text{Traffic} := [S|S \models &\text{«Driver, } d, 1\text{»}\Lambda\text{«Sign, } \dot{x}, 1\text{»}\Lambda\text{«passes, } d, \dot{x}, 1\text{»}\\ &\rightarrow \text{«SignPassed, } d, 1\text{»}].\end{aligned}$$

The dots denote variables as *parameters* which are essentially placeholders allowing the specification of properties and relations for multiple entities simultaneously. In the example, traffic is a situation where a single driver d and a number of signs $\dot{x}$ exist. This situation is further constrained by the fact that whenever the driver passes any sign, the driver also switches to state 'Sign passed'.

Let us further assume, the sign in the above example would be a warning sign. Depending on the type of potential hazard drivers identify on the sign, they may adjust their driving and find themselves in a new situation—general driving S versus driving under a particular danger SD. In this paper, we are concerned with the interpretation of the traffic sign and the required information the driver needs to make the transition from one situation to the other ($S \Rightarrow SD$). To interpret the traffic sign, they need to be familiar with the concept of danger-warning signs, glyphs used to encode potential dangers and the concept that a glyph inside the sign warns of this danger. Since we have already defined the general idea of a warning sign, this specification can be described as a form of reinterpretation, specifying the following equivalence relation on the sign object:

$$\begin{aligned}(&\text{«WarningSign, } x, 1\text{»}\,\Lambda\\ &\text{«contains, } x, g, 1\text{»}\Lambda\\ &\text{«PedestrianGlyph, } g, 1\text{»}) \leftrightarrow \text{«PedestrianWarning, } x, 1\text{».}\end{aligned}$$

Thereby, a warning sign containing a glyph showing a pedestrian is reinterpreted as a warning about pedestrians.

What we have not modelled thus far are the consequences drawn from the traffic sign's interpretation. Indeed, the interpretation of the sign motivates drivers to re-evaluate their own situation and change their behaviour according to the sign. For example, they may slow down due to the warning, changing their own situation from 'normal driving' to 'cautious driving'. Thus, reinterpretations do affect not only objects but also situations. Modelling this aspect more formally helps us to better understand the different sources of

knowledge involved in the interpretation of (visual) information and in the process of drawing conclusions from what is seen. To this end, we extend situation semantics by an (re-)interpretation operation $\iota : (X, R_X, B) \rightarrow \text{«}Y, y, 1\text{»}$. An interpretation depends mainly on two arguments: a set of objects $X = \text{«}X_i, x_i, 1\text{»}$ together with the relations R_X connecting them. Additional background information B can be provided, for example, to impose further conditions on the reinterpretation. The interpretation operation maps this information to an equivalent target type.

3.2.2. *Situation semantics in visualization*

Much like the driver in the traffic situation, an observer of a visualization changes their situation during the reasoning process. Once they recognize a particular type of chart, they understand how to read it and which types of features they may look for, e.g. read values of data points in a scatterplot or identify clusters. We can also model transitions between different tasks where users are differently primed to look out for particular features, or may switch between overview analysis and detail analysis. Before we look into these more complex situations, we first translate the concepts from situation semantics we have applied to traffic signs in the previous section to data visualization using the speedup chart example detailed in the 'Introduction' and represented in Figure 1.

The interpretation of a speedup chart by an observer strongly depends on the observer's background knowledge—whether they are a child, a visualization literate person, a statistician or a parallel programmer.

It also depends on experience and analysis perspective, all of which needs to be taken into account when designing a visualization and reasoning about the observer's reasoning process. It is important to understand which parts of a visualization can be understood by means of the appropriate technical skills (e.g. data visualization literacy) and provided information (e.g. a legend), and which aspects require domain-specific knowledge. Situation semantics can help us formalize required knowledge. Combining this information with the syntactic information we obtained through the graphical language discussed in Section 3.1, we have all required means to investigate chart interpretation analysis which will follow in Section 3.3.

Many of the concepts we have seen in the previous section directly apply to visualization. We will work through them using again the line chart example from Figure 2. Object types that we can see in this chart are axis (two instances) and polyline (one instance). Denoting this in situation semantics notation for our general situation V gives:

$$V \models \text{«Axis}, x, 1\text{»}\Lambda\text{«Axis}, y, 1\text{»}\Lambda\text{«Polyline}, p, 1\text{»}. \quad (8)$$

Each of these objects feature properties that are modelled in the type definition, e.g. an axis contains a line, ticks and tick marks. For simplicity, we also accept implicit definitions, especially where the details are specified by the syntax structure or where they are not needed for modelling the interpretation and reasoning process. The aforementioned concept of partiality guarantees the formulation in situation semantics to remain sound even for such incomplete models.

From the syntactic description and chart design conventions, we know that axes commonly bound the line glyph. This is expressed by the following infons:

$$V \models \text{«bounds bottom}, x, p, 1\text{»}\Lambda\text{«bounds left}, y, p, 1\text{»}. \quad (9)$$

These pieces of information are the ones that the viewer can derive directly from the visualization and a child without chart literacy will stop at this interpretation. An interpretation $\iota : (X, R_X, B) \rightarrow \text{«}Y, y, 1\text{»}$ requires three inputs: a set of relevant objects X, relations between them R_X and background information B. The objects are the visible structures defined in (8), the relations the relative positions to each other (orthogonality of axes and bounding of polyline) defined in (9) and the background information that is required is DVL. If all this information is available, the viewer will interpret the given chart as a line chart:

$$\begin{aligned}&\text{«Axis}, x, 1\text{»}\Lambda\text{«Axis}, y, 1\text{»}\Lambda\text{«Polyline}, p, 1\text{»}\Lambda\\ &\text{«bounds bottom}, x, p, 1\text{»}\Lambda\text{«bounds left}, y, p, 1\text{»}\Lambda\\ &\text{«is orthogonal}, x, y, 1\text{»}\Lambda\text{«DVL}, v, 1\text{»} \leftrightarrow\\ &\text{«LineChart}, c, 1\text{»}. \end{aligned} \quad (10)$$

We can now combine the understanding of a line chart with domain knowledge in order to model an actual reasoning process. Figure 1 features a line chart to show the speedup of a parallel program depending on the number of processors used for the computation. A high performance computing engineer is interested in this kind of measurement to evaluate the performance of a program. Optimal speedup behaviour is close to linear increase. The engineer now extends the above recognition of the line chart by three further interpretation. First, introducing the knowledge that the analysis is conducted in a parallel programming context and that this chart is showing speedup (which is concluded from the y-axis' label), the engineer reinterprets the line chart to be a speedup chart:

$$\begin{aligned}&\text{«LineChart}, c, 1\text{»}\Lambda\text{«}shows, speedup, c, 1\text{»} \leftrightarrow\\ &\text{«SpeedupChart}, c, 1\text{»}, \end{aligned} \quad (11)$$

where a chain of constraints embed the speedup chart in its general context by declaring the particular chart c to be linked to some parallel program $\dot{p}$:

$$\begin{aligned}&\text{«SpeedupChart}, c, 1\Lambda\text{«}shows\text{», speedup}, c, 1\text{»}\Lambda\\ &\text{«}for\text{», speedup}, \dot{p}, 1\text{»}\Lambda\text{«ParalellProgram}, \dot{p}, 1\text{»»}. \end{aligned} \quad (12)$$

To judge the program's speedup behaviour, the engineer next evaluates the depicted speedup curve according to the interpretation rule:

$$\begin{aligned}&\text{«LineGlyph}, c, 1\text{»}\Lambda\text{«LinearCorrelation}, l, 1\text{»}\Lambda\\ &\text{«LineChart}, c, 1\text{»}\Lambda depicts, c, l, 1 \leftrightarrow\\ &\text{«LinarlyCorrelatedVariabes}, c, 1\text{»}. \end{aligned} \quad (13)$$

Note that if the viewer does not recognize the line glyph to show linear correlation, the observed situation would show $V \models$ «LinearCorrelation, l, 0» and the interpretation could thus not be made. However, in the example shown in Figure 1, the correlation

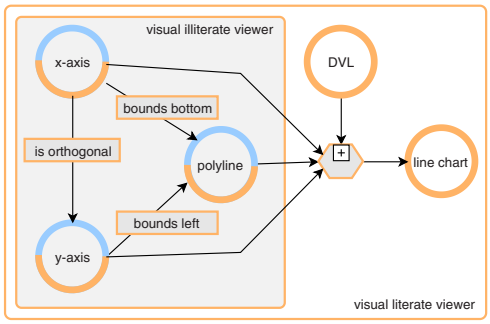

Figure 7: *Graph-based model of situation semantics: This model represents types (nodes), relations (edges) and interpretations (hexagon) performed in the reasoning process about the type of a given chart (line chart). Equation 10 shows the expression modelling this interpretation in situation semantics.*

fortunately is linear allowing the engineer to apply the last interpretation rule, concluding that a program $\dot{p}$ features optimal speedup if its speedup chart shows linearly correlated variables. For the modelling, we leverage the fact that interpretations implicitly specify situations by subsuming the objects and relations to the left of the equivalence relation under the object of the new type specified to its right. We can thus use type abstraction to combine (11) with (13) and the additional background information given in (12) into the following interpretation rule:

$$\begin{aligned} &\text{«SpeedupChart, } c\text{, 1}\Lambda\text{«LinarlyCorrelatedVariabes, } c\text{, 1»}\Lambda \\ &\qquad \text{«}\mathit{shows}\text{, speedup, } c\text{, 1»}\Lambda\text{«}\mathit{for}\text{, speedup, } \dot{p}\text{, 1»}\Lambda \\ &\qquad\qquad \text{«ParalellProgram, } \dot{p}\text{, 1»»} \leftrightarrow \\ &\qquad\qquad\qquad \text{«OptimalSpeedup, } \dot{p}\text{, 1».} \end{aligned}$$

Again note that $\dot{p}$ refers to a general program because no specific instance of a program is given in this example. The line chart instead is bound to a specific instance c of a chart. In general, objects that are not part of the visualization are never instantiated and thus modelled as parametric objects when used in the reasoning structure. Parameters allow to specify general rules meant to apply to any object of the given type once the particular object is introduced. In the example, a particular program could, for example, be introduced by adding a caption with the program's name to the chart.

3.2.3. *Graph-based visualization of situation semantics*

As we have seen in the two previous examples (traffic signs and line charts), the semantic description quickly becomes quite complex. To keep track of all components and their relations, we propose a graphical language for situation semantics in the context of visualization.

Similar to the flowchart for the syntax, we will also design a graph-based representation for situation semantics which primarily consists of types and relations. *Object types* and *concepts* are the nodes of the graph. Object types appear as abstract, purely semantic models just as they are defined in situation semantics. Where we are referring to actual instances of objects that are part of the visualization, the object becomes a hybrid of its syntactic representation in the graphical language and the abstract semantic information being anchored in the depicted instance of the object. To emphasize the duality of being a semantic object to reason about that at the same time is anchored to an actual instance and thus depends on the data and visualization technique, we refer to this kind of node as a *concept*. Other than type nodes which always remain abstract, concept nodes can be reasoned about in the abstract but can also appear as instances of graphical elements or sentences whose behaviour on the semantic level might be different due to the properties of each respective instance. For semantic analysis, this means concepts have to be evaluated by instance, and the polarity of several infons within the object definition might differ between different instances of the same concept. In the line chart example we have three concepts: x-axis, y-axis and polyline which are depicted in Figure 7 by labelled nodes with a blue–orange boundary. *Constraints* between objects are depicted as annotated edges, e.g. the x-axis is orthogonal to the y-axis («is orthogonal, x, y, 1»). *Interpretations* are represented by a hexagon with a plus sign. Relevant objects are directly connected to the hexagon. The background information is linked to the plus sign. The constraints involved are simply the ones to be found between any two nodes being connected to the interpretation. The resulting type is marked by the single outgoing edge from the hexagon.

We have also seen that situations play a critical role. As situations are sets of types and relating relations, they can be represented by bounding areas in the chart. In Figure 7, we distinguish between the visual literate and illiterate viewer that obtain different information from the same chart. The illiterate viewer can recognize objects and their spatial relationship, the literate viewer will understand the graphical context.

As detailed before, situation semantics is partial, i.e. we model only the aspects that are relevant to our analysis at the level of detail that is appropriate. As we can see from this simple example of a line chart, the graph is already very complex. Hence, we will summarize structures into single nodes where appropriate to reduce visual complexity and clutter.

The reasoning process can now be modelled in terms of reachability via constraints and interpretations. Thus, to determine the possible conclusions to be drawn about an object (type/concept) or a situation, one simply has to follow the constraints and interpretations through the graph. For example, the reasoning process discussed for the line chart example above is completely encoded in the graph to the right of Figure 1. Thereby, the graph modelling the reasoning implicitly defines a formal automaton allowing to compute the language of possible interpretations and conclusions to be drawn from an type/concept or situation as the collection of all reachable types/concepts and the constraints and interpretations between them. This can be implemented algorithmically in terms of an altered breadth-first search on the graph that takes into account the conditions that have to apply before an interpretation can be executed. In this sense, the computation resembles a Petri net up to the point that more complex conditions than the mere presence of a token can be required for an interpretation to be triggered.

3.3. The ConceptGraph: combining syntax and semantics

In the previous two sections, we described two formal approaches to turn syntactic and semantic information obtained from

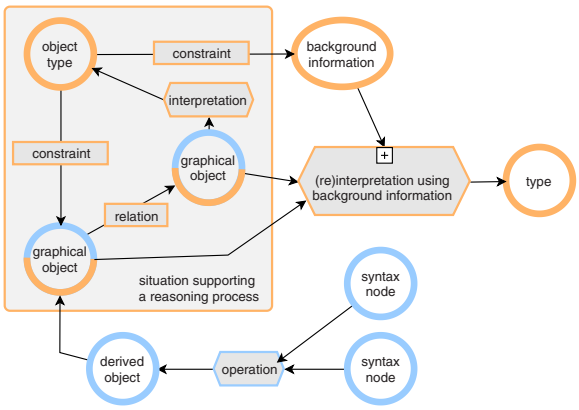

Figure 8: *Overview over the ConceptGraph's notation: The concept nodes (both colours) establish the mapping between the visualization, its generation and the mental model. The blue subgraph is the automaton modelling the visualization process. The graphical sentences and contexts (blue–orange nodes) that can be read as structures in the result of running this automaton serve as the input to be processed by the reasoning strategy represented by the orange subgraph.*

visualizations into a graph-based representation. As our goal is a unified analysis of these two aspects that constitute the visual reasoning process, we need to combine the two models. The coupling of the two automata directly originates from the fact that the output of the first is the input to the second, i.e. the grammar of graphics defines an automaton which produces graphical elements and sentences. The reasoning process as modelled above can also be interpreted as a finite state transducer whose input language is the reading language defined as graphical contexts over the visualization automaton's output. Thereby, the two automata share a set of nodes. These shared nodes are exactly the concept nodes specified above as nodes that at the same time feature abstract semantics but can also be instantiated by the visualization. In the depiction, we overlap these nodes in the joint visualization and indicate the sharing by mixed colour. As introduced before, we will continue to colour syntactic aspects of the interpretation in blue and semantic aspects in orange. Shared nodes are coloured in blue and orange. The combination of the two graphs is called a *ConceptGraph*.

Figure 1 (right) depicts the ConceptGraph of the speedup chart. The blue–orange parts are the visible items. Blue nodes and operations (hexagons) represent syntactic information which was derived from the visualization Vega specification. The orange parts represent the reasoning process derived from situation semantics. What this graph nicely illustrates is the different insights that may be obtained from people with varying backgrounds. The child will only be able to recognize mixed colour nodes which are directly visible without additional semantic information. A chart-literate person understands the graphical context and interprets it as line chart. It is also likely that they understand about scales and how they affect axes and the line placement. The statistician will extend this knowledge by the concept of linear correlation and hence, probably be able to apply this to the chart and evaluate the linear dependence of the two presented variables. The parallel programmer understands the application background and can make sense of what is shown, i.e. that the chart is a speedup chart. To understand the concept of an optimal parallel program, they also need to understand linear correlation.

A summary of the graphical encoding used in the ConceptGraph is given in Figure 8. The ConceptGraphs were manually drawn using Draw.io.[2]

4. Applications

In the following two scenarios, we will apply the proposed model to two additional use cases. In the first one, we investigate linked views and the resulting reasoning process and in the second one, we investigate varying complexities of tasks.

4.1. Conditional views

Figure 9 presents a scatterplot of the MPG dataset [Qui93]. The variables horsepower and miles per gallon are encoded as position, and origin as colour. The legend is interactive and selecting either of the regions fades out the others. The example is available in the Vega website.[3]

The respective ConceptGraph (Figure 9, right) contains information the user may obtain when browsing this task. Visible concepts (bi-colour nodes) are the axes, point glyphs, and the legend. The user may recognize that there are points of different colour and focus on them individually (→ filter operation). Thus, they are able to recognize gray and coloured points. Reasoning about the entire plot (left side of ConceptGraph), they may interpret this chart as a scatterplot (→ scatterplot node) and applying this knowledge back on the individual parts, they understand that each point in the chart represents a car (→ car node) and each axis an attribute of the cars (→ attributes node + relationship). Combining these two pieces of information (point information + global chart information) and applying the knowledge they obtained from the knowledge, the observer can conclude that the green points are European cars and the grey cars originate either from the United States or Japan.

From the ConceptGraph, we see that it takes several interpretations to understand the presented chart and that this type is already much more complex than the line chart we saw earlier. What we also learn is which concepts need to be explained to a novice user that is trying to understand the given visualization.

To a literate user, we may pose the task to compare European and non-European cars with respect to their horsepower. They would have to understand the aspects we have discussed so far and make additional interpretations and operations (top part of ConceptGraph). Assuming they did interpret the data points correctly, they also need to identify the correct axis and compare the grey and green points with respect to this axis. In this way, we can discover potential error sources in the reasoning process and reason about the difficulty of the various steps.

[2] https://www.draw.io

[3] https://vega.github.io/vega/examples/interactive-legend/

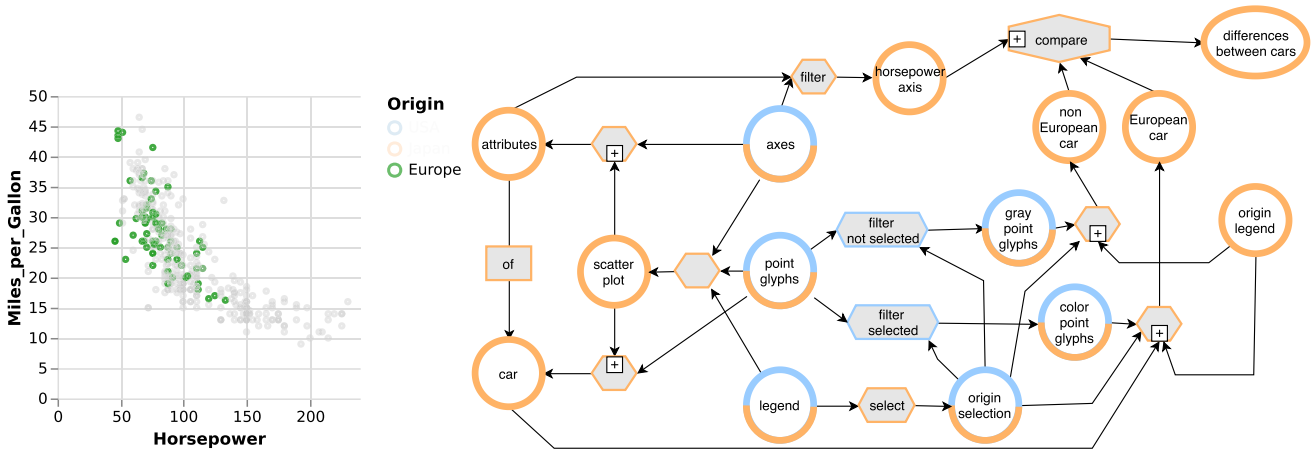

Figure 9: *ConceptGraph of a linking+brushing example: (left) An interactive scatterplot of the mpg–dataset encoding horsepower, miles per gallon and origin of cars. The legend is interactive and European cars are highlighted. Cars from the United States and Japan are coloured in grey. (right) The concept graph for chart browsing and comparative analysis of European and non-European cars. (Data: auto-mpg [Qui93])*

4.2. Task-based reasoning complexity

In the second example, we discuss the application of the ConceptGraph for different analysis tasks. Figure 10 shows a force-directed layout of the co-occurrence network data of the Les Miserable dataset. Communities are provided in the data and colour-coded in the visualization. The ConceptGraph in the same figure represents the information flow in the algorithmic part. Starting from this initial concept graphs, we will look into several tasks and the necessary reasoning processes.

The first question is as follows: How many communities does the network contain? The literate user will understand that they can obtain this information quickly from the legend by counting the number of entries. Alternatively, the user has to do the computational work manually and identify all different colours in graph visualization. We do not depict this concept graph as it is similar to the one in the next task. In the various task classification frameworks (see related work), this task belongs to the lowest level which is considered to be easy. This agrees with our observation.

The second task is: What is the size of the largest community? Here, we assume that the user iterates over all colours they see, count the respective nodes and keep track of the maximal value. We observe that multiple pieces of information have to be held in working memory and the more operations are required than in the previous task. Count, sort and compare tasks are commonly considered to be of medium difficulty which is true from the perspective of an intellectual challenge. Figures 11 and 12 show concept graphs modeling reasoning for counting finding the largest community.

The most difficult types of tasks are the ones that require analytical reasoning using background knowledge, like: Does the olive-coloured community feature a clique structure? or Why does the pink node form its own community? While the first question can still be answered if the user is familiar with the concept of a clique, the second one requires in-depth knowledge of the algorithm and probably additional information about the data like the edge weights which are not depicted in the presented chart.

5. Discussion and Future Work

In the above discussion, we introduced a theoretical framework describing how viewers reason with and about visualizations. The model is based on a solid theoretical foundation formalizing the processes of visualization generation and interpretation in terms of formal languages and logic. Because the formalism is represented in terms of formal automata, it scales well with increasingly large and complex visualizations displaying large amounts of data.

What sets our framework apart from applying ontologies to visualization is the idea of computing the semantics rather than just assigning it. The focus of the presented model is on processing and relating interpretations in order to obtain more information. One major design goal was to obtain an actionable model, in which the semantics associated with depicted graphical structures can be computed with respect to a pre-determined model for the reasoning process. We achieved this by modelling the visualization generation and interpretation as a two-step process. First, a formal automaton models the visualization generation. In our formalization, it is based on Vega but may as well use other existing grammars for visualization specification. The reasoning process is formalized in terms of situation semantics extended by an additional interpretation relation. The possibility of an interpretation is thereby determined by its reachability in a network of consecutively evaluated implications and equivalence relations.

Perhaps the most appealing feature of the presented model is its graphical annotation in terms of the ConceptGraph. By the ConceptGraph, the complexity of modelling the reasoning strategy is reduced to specifying a sophisticated mind map rather than having to directly define all semantic types and their relations explicitly. This renders the theoretical framework applicable to a wide range of possible users.

There are, however, some limitations to the framework. At this point of the development, the exact translation between the graphical language and a language of the exact structures being recognized and investigated by the viewer is left open. Being concerned only

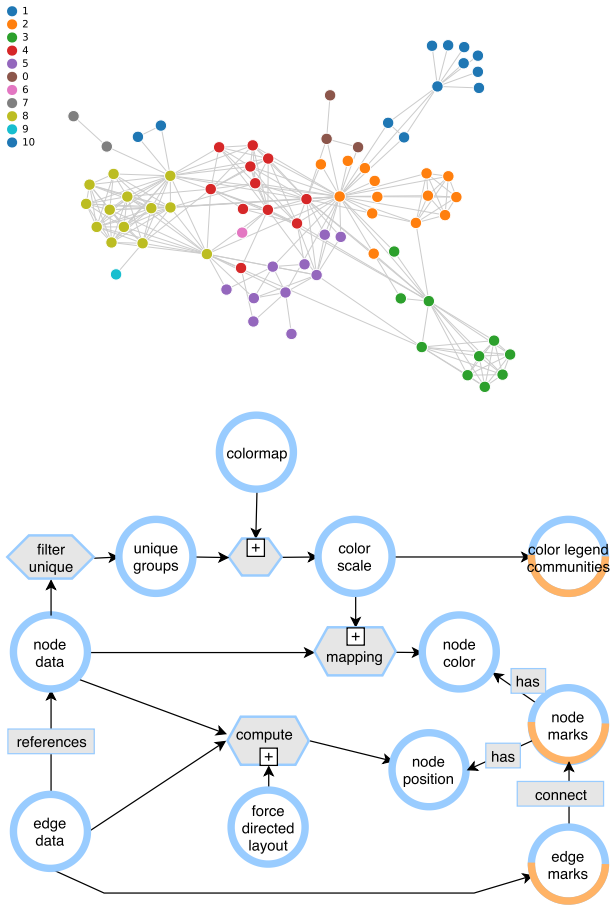

Figure 10: *Reasoning process for a network: (top) co-occurrence network of the LesMiserable datas, (bottom) ConceptGraph for the syntactic aspects.*

Figure 11: *ConceptGraph for the counting task.*

with the cognitive processing of interpretations and not with perceptual issues, closing this gap is beyond the scope of the presented model. However, there is a wide range of existing theory concerned with this problem. For example, information theory has been shown to be a useful model for the transport of information from the visualization to the viewer, rendering it an interesting candidate for modelling the translation between the graphical and the reading language [CJ10]. Another approach could be to model the transport of information in terms of lossy channel systems. This kind of system allows to drop portions of information before it is being received and therefore is an interesting candidate for considerations on the robustness of a reasoning strategy subject to missing components in the graphical sentences and contexts. Indeed, it will be interesting to apply the new model to analyse this robustness and to thereby find

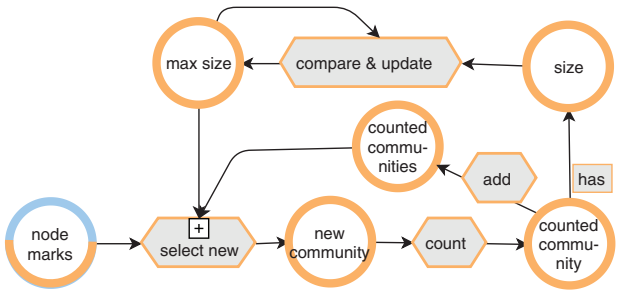

Figure 12: *ConceptGraph for the find largest community task.*

mechanisms determining information critical for solving a given task and to identify patterns in the ConceptGraph that help designing visualizations effectively preventing this information from not being recognized by the viewer.

An inherent problem of the reasoning process itself is its subjectivity. Sometimes, it is not clear when exactly to apply a certain interpretation or there are interpretations for which the exact criteria when they might be applied are not known. An example for such a situation is a continuous scale like a heat map. Every viewer will interpret the colours slightly differently. Yet, it can still be expected that they will attempt to interpret the data into similar classes. Therefore, the presented model considers the applicable semantics following all possible interpretations simultaneously. The actual semantics determined by the viewer's interpretation can still be computed by specifying which of the applicable interpretations should actually be applied. Note, however, that this does by no means advocate for modelling huge ConceptGraphs attempting to capture as many different interpretations as possible as it is sometimes attempted for ontologies. Other than being sound, the reasoning strategy still should attempt to match the actual reasoning applied by domain experts as best as possible. After all, the answers obtained by evaluating applicable interpretations in the presented model can only be as good as the reasoning strategy's fitting to the domain of investigation.

6. Conclusion

The theoretical framework introduced in this paper allows to explain how viewers reason about structures displayed in a visualization. Due to its solid mathematical underpinning based on the theories of formal automata and situation semantics, the framework allows to specify formal models of the reasoning processes to be applied in order to draw conclusions from the structures depicted in a visualization. Its concise graphical annotation, the ConceptGraph, renders it simple to apply for a wide audience.

The model along with the ConceptGraph is therefore not only of theoretical interest but also bears interesting potential for visualization design.

Acknowledgements

The authors would like to thank Alina Freund, Julio Palacios and Christoph Garth for the productive and fruitful discussions helping us to shape our formalism. We would also like to thank the

anonymous reviewers for their patience, their insightful and constructive remarks and their overall support during the process of authoring this paper.

References

[AS13] Angelini M., Santucci G.: Modeling incremental visualizations. In *EuroVis Workshop on Visual Analytics* (2013), M. Pohl, and H. Schumann (Eds.), The Eurographics Association.

[Bar86] Barwise J.: The situation in logic. In *Logic, Methodology and Philosophy of Science VII Proceedings of the Seventh International Congress of Logic, Methodology and Philosophy of Science*. G. J. D. Ruth Barcan Marcus, and P. Weingartner (Eds.), vol. 114 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1986, pp. 183–203.

[BBG19] Börner K., Bueckle A., Ginda M.: Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proceedings of the National Academy of Sciences of the United States of America 116*, 6 (2019), pp. 1857–1864.

[Ber83] Bertin J.: *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison, 1983.

[BOH11] Bostock M., Ogievetsky V., Heer J.: D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309.

[Bör16] Börner K.: Data Visualization Literacy. *Proceedings of the 27th ACM Conference on Hypertext and Social Media* (2016), 1.

[BP83] Barwise J., Perry J.: *Situations and Attitudes*. MIT Press, Cambridge, MA, 1983.

[CG02] Celentano A., Gaggi O.: Schema modelling for automatic generation of multimedia presentations. *27* (2002), 593–600. https://doi.org/10.1145/568760.568864.

[CJ10] Chen M., Jaenicke H.: An information-theoretic framework for visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov 2010), 1206–1215.

[Cop12] Coppin P. W.: Pictures are visually processed; symbols are also recognized. In *Diagrammatic Representation and Inference*. P. Cox, B. Plimmer and P. Rodgers (Eds.). Springer, Berlin, Heidelberg (2012), pp. 334–336.

[Dev99] Devlin K. J.: *Infosense: Turning Information Into Knowledge*. WH Freeman & Co., New York, 1999.

[Dev06] Devlin K.: Situation theory and situation semantics. In *Handbook of the History of Logic*. J. Woods and D. M. Gabbay (Eds.), vol. 7. Elsevier, Amsterdam, 2006, pp. 601–664.

[DGH03] Doleisch H., Gasser M., Hauser H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the Symposium on Data Visualisation 2003* (Aire-la-Ville, Switzerland, Switzerland, 2003), VISSYM '03, Eurographics Association, pp. 239–248.

[DSK*14] Demiralp C., Scheidegger C. E., Kindlmann G. L., Laidlaw D. H., Heer J.: Visual embedding: A model for visualization. *IEEE Computer Graphics and Applications 34*, 1 (Jan 2014), 10–15.

[ECD06] Elzer S., Carberry S., Demir S.: Communicative signals as the key to automated understanding of simple bar charts. *Proceedings of the 4th International Conference on Diagrammatic Representation and Inference* (Springer-Verlag, Stanford, CA, 2006) pp. 25–39. https://doi.org/10.1007/11783183_5.

[Eng02] Engelhardt Y.: *The Language of Graphics: A Framework for the Analysis of Syntax and Meaning in Maps, Charts and Diagrams*. PhD thesis, (Universiteit van Amsterdam, Amsterdam, The Netherlands, 2002).

[EWCH04] Elzer S., White M., Carberry S., Hoffman J.: Incorporating perceptual task effort into the recognition of intention in information graphics. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science) 2980*, 255–268.

[GWK93] Guthrie J. T., Weber S., Kimmerly N.: Searching documents: Cognitive processes and deficits in understanding graphs, tables, and illustrations. *Contemporary Educational Psychology 18*, 2 (1993), 186–221.

[IG96] Ihaka R., Gentleman R.: R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics 5*, 3 (1996), 299–314.

[Kos89] Kosslyn S. M.: Understanding charts and graphs. *Applied Cognitive Psychology 3*, 3 (1989), 185–225.

[KS14] Kindlmann G., Scheidegger C.: An algebraic process for visualization design. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec 2014), 2181–2190.

[LKH*15] Lee S., Kim S.-H., Hung Y.-H., Lam H., Kang Y.-a., Yi J. S.: How do people make sense of unfamiliar visualizations?: A grounded model of novice's information visualization sensemaking. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2015), 499–508.

[LS10] Liu Z., Stasko J.: Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 999–1008.

[Mac86] Mackinlay J.: Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics 5*, 2 (Apr. 1986), 110–141.

[MHS07] Mackinlay J. D., Hanrahan P., Stolte C.: Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics 13*, *6* (2007), 1137–1144.

[MMWC18] Mei H., Ma Y., Wei Y., Chen W.: The design space of construction tools for information visualization: A survey.

Journal of Visual Languages & Computing 44, (2018), 120–132. https://doi.org/10.1016/j.jvlc.2017.10.001.

[MRGTBD08] M Ratwani R., Gregory Trafton J., Boehm-Davis D.: Thinking graphically: Connecting vision and cognition during graph comprehension. *Journal of Experimental Psychology. Applied 14* (04 2008), 36–49.

[Mun09] Munzner T.: A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov. 2009), 921–928.

[MWN*19] Moritz D., Wang C., Nelson G. L., Lin H., Smith A. M., Howe B., Heer J.: Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 438–448.

[NKH*14] Nazemi K., Kuijper A., Hutter M., Kohlhammer J., Fellner D. W.: Measuring context relevance for adaptive semantics visualizations. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business* (New York, NY, USA, 2014), i-KNOW '14, ACM, pp. 14:1–14:8.

[PDFE17] Park D., Drucker S. M., Fernandez R., Elmqvist N.: Atom: A grammar for unit visualizations. *IEEE Transactions on Visualization and Computer Graphics 24*, 12 (2017), 3032–3043.

[PG93] Petre M., Green T.: Learning to read graphics: Some evidence that 'seeing' an information display is an acquired skill. *Journal of Visual Languages & Computing 4*, 1 (1993), 55–70.

[Qui93] Quinlan R.: Combining instance-based and model-based learning. In *Proceedings on the Tenth International Conference of Machine Learning* (University of Massachusetts, Amherst, Morgan Kaufmann, 1993).

[Ren00] Rensink R. A.: The dynamic representation of scenes. *Visual Cognition 7*, *1–3* (2000), 17–42.

[RTBD04] Ratwani R. M., Trafton J. G., Boehm-Davis D. A.: From specific information extraction to inferences: A hierarchical framework of graph comprehension. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 48*, 16 (2004), 1808–1812.

[SASS16] Schulz H. J., Angelini M., Santucci G., Schumann H.: An enhanced visualization process model for incremental visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 7 (July 2016), 1830–1842.

[SML*08] Smuc M., Mayr E., Lammarsch T., Bertone A., Aigner W., Risku H., Miksch S.: Visualizations at first sight: Do insights require training? In *HCI and Usability for Education and Work*. A. Holzinger (Ed.), Springer, Berlin, Heidelberg (2008), pp. 261–280.

[SRHH15] Satyanarayan A., Russell R., Hoffswell J., Heer J.: Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2015), 659–668.

[SSL*12] Streit M., Schulz H.-J., Lex A., Schmalstieg D., Schumann H.: Model-driven design for the visual analysis of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (June 2012), 998–1010.

[TGOA15] Thellmann K., Galkin M., Orlandi F., Auer S.: *LinkDaViz—Automatic Binding of Linked Data to Visualizations*. (Springer-Verlag, Berlin, Heidelberg, 2015).

[TMMT02] Trafton J. G., Marshall S., Mintz F., Trickett S. B.: Extracting explicit and implicit information from complex visualizations. In *International Conference on Theory and Application of Diagrams* (2002), Springer, pp. 206–220.

[Tom11] Tominski C.: Event-based concepts for user-driven visualization. *Information Visualization 10*, 1 (2011), 65–81.

[TSK12] Takemura R., Shimojima A., Katagiri Y.: A logical investigation on global reading of diagrams. In *Diagrammatic Representation and Inference*. P. Cox, B. Plimmer, and P. Rodgers (Eds.). Springer, Berlin, Heidelberg (2012), pp. 330–333.

[TT01] Trafton J. G., Trickett S. B.: *A New Model of Graph and Visualization Usage*. Tech. rep., Naval Research Lab Washington, DC, 2001.

[Uni68] United Nations, General Assembly: Convention on road signs and signals, 8 November 1968.

[Veg19] Vega website: https://vega.github.io, accessed August 2019.

[VFR13] Vickers P., Faith J., Rossiter B. N.: Understanding visualization: A formal approach using category theory and semiotics. *CoRR 19*, (2013), 1048–1061.

[Wic09] Wickham H.: *Elegant Graphics for Data Analysis*. *Media* (Springer-Verlag, New York, 2009).

[Wic10] Wickham H.: A Layered grammar of graphics. *Journal of Computational and Graphical Statistics 19*, 1 (2010), 3–28.

[Wil05] Wilkinson L.: *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[Wil12] Wilkinson L.: The grammar of graphics. In *Handbook of Computational Statistics*. J. E. Gentle, W. K. Härdle and Y. Mori (Eds.). Springer, Berlin, Heidelberg (2012), pp. 375–414.

[Wom15] Womack R.: Data Visualization and Information Literacy. *IASSIST Quarterly*, *38*, 1 (2015), 12.

[WW10] Wills G., Wilkinson L.: Autovis: Automatic visualization. *Information Visualization 9*, 1 (2010), 47–69.

[ZM06] Zhu S. C., Mumford D.: A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, *2*, 4 (2006), 259–362.