High-level Grammars for Visualization and Visual Analytics

Gustavo Moreira*, Marcos Lage†, Nivan Ferreira‡, Fabio Miranda*
*University of Illinois Chicago, †Universidade Federal Fluminense, ‡Universidade Federal de Pernambuco
gmorei3@uic.edu, mlage@ic.uff.br, nivan@cin.ufpe.br, fabiom@uic.edu

***Abstract*—The increase in data availability and the popularization of data science brought a continuously growing need to use visualizations to explore complex data. Creating new visualizations, however, is a difficult task that often requires extensive programming expertise. High-level grammars for authoring visualizations provide a systematic and flexible way to specify visualizations in a declarative manner, lowering the entry barrier for data analysis. Such grammars effectively assist in the exploration of the visualization design space and of complex datasets. The goal of this tutorial paper is to then provide a summary of recently proposed grammars, so that a broad audience (e.g., students, researchers, practitioners) can better understand how they are designed and used in practice for visualization and visual analytics tasks.**

I. Introduction

The importance of data in decision-making in virtually all domains of human activity has fostered the popularization of data science and its related fields. This fact, alongside the advancement of data acquisition, data management, and data processing capabilities, has fostered applications in a wide range of domains, including urban informatics, finance, and science, to name a few. Data visualization techniques and systems are well known to be indispensable tools in this context, supporting both data exploration and communication. Given this popularization, we have witnessed in recent years the proposal of a plethora of tools for authoring visualizations and interactive dashboards [1]. These tools vary with respect to their expressiveness (*What can be built?*), accessibility (*What skills the user must have to use it?*), efficiency (*How long does it take to build?*) and, therefore, support different applications and users. At one end of a spectrum, one can find visualization programming libraries, such as D3 [2] and Processing [3], that provide high expressiveness (users can develop a large number of possible designs), while, in general, requiring specialized users with programming skills (low accessibility) and also usually demanding relatively long periods of development (low efficiency). On the other end, systems like Tableau and Power BI offer direct manipulation interfaces to compose customized visualizations and interactive dashboards. The main advantage of these systems is the fact that users with no programming background (high accessibility) can quickly develop interactive dashboards (high efficiency). However, these dashboards are limited to the data types, visualizations, and interactive mechanisms supported by the system (low expressiveness).

979-8-3503-3872-0/23/$31.00 ©2023 IEEE

An intermediate approach is the use of grammars for visualization end visual analytics. Grammars are a high-level declarative approach that abstracts important low-level concepts for the creation of visualizations and interactive dashboards. Examples in this category include grammars intended for general visualization tasks, such as Vega-Lite [4] as well as the ones intended for specific scenarios, such as DXR [5] for the creation of immersive analytics applications and the Urban Toolkit [6] used to create spatial visualizations for urban analytics. The grammar approach offers a compromise between the two ends of the spectrum. In this case, while some textual description is still used, no programming capabilities are required (thus providing high accessibility), while preserving expressiveness and achieving a compromise in efficiency when compared to the direct manipulation interfaces.

With the large number of visualization grammars proposed, and their trade-offs to support different users and use cases, a tutorial covering a range of grammars and discussing them in perspective can be valuable to a broad audience. Such material would be valuable to users interested in learning how to use grammars to design visualizations in different applications fields and also to more specialized users that want to learn about design choices, capabilities, and limitations. Given this scenario, this tutorial paper has three main objectives. First, present an overview of the foundations of grammars of graphics, and how it substantiates the many developments in the visualization field. Second, present an overview of the subsequent works on grammars for authoring visualizations, with a more in-depth (and practical) look at Vega-Lite [4], given its widespread adoption across visualization systems. And third, present our perspectives on the design of a domain-specific grammar used by the Urban Toolkit [6]. In doing so, we hope to not only provide an overview of a newly developed grammar but also encourage new research on the topic. By framing this paper around these three goals, we will cover both theoretical aspects of visualization grammars (and how they are grounded in Bertin's graphics semiology work [7]) as well as practical ones (through hands-on examples).

The rest of this document is organized as follows. Section II discusses prior works on grammars for visualization. Then, Section III and Section IV discuss Vega-Lite and the Urban Toolkit, respectively. These two examples illustrate the diversity and power of the grammars for authoring visualizations.

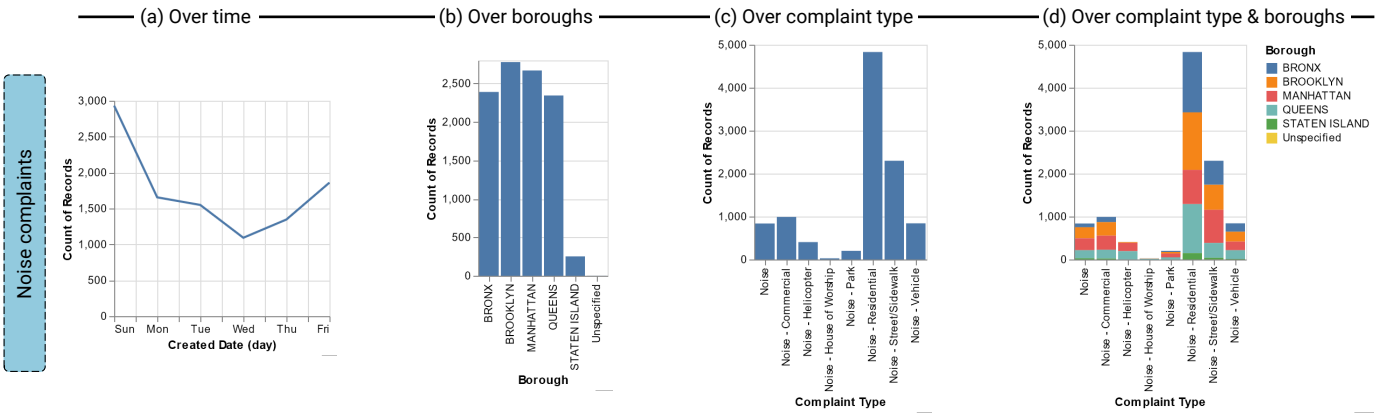

Fig. 1. Data exploration with Vega-Lite: (a) Distribution of noise complaints over time. (b) Distribution of noise complaints over boroughs. (c) Distribution of types of noise complaints. (d) Distribution of noise complaints over boroughs and noise types. Similar visualizations can be generated with fewer than 20 lines of JSON specification.

II. An Overview of Grammars for Visualization

In this section, we give a brief overview of previous grammars specifically designed with visualization and visual analytics goals in mind. Wilkinson first proposed a grammar for graphics [8], building on top of Bertin's semiology of graphics [7]. In his work, Wilkinson contributes a grammar to describe and construct a variety of statistical graphics. Using such grammar, one can map data items and attributes to visual marks and channels, as opposed to chart templates that map data attributes to aspects of a chart. Influenced by Wilkinson's grammar, several recent works have proposed software implementations of grammars for information visualization, scientific visualization, and visual analytics. These works offer the user the ability to declaratively specify a visualization using a high-level language, abstracting low-level details of *how* a visualization is implemented. Concretely, this means that users can make small changes in the specification (e.g., changing marks), without re-designing the entire visualization. Following this idea, visualization systems extensively leverage grammars to lower the barrier for authoring visualizations [9]. We refer the reader to McNutt's survey [10] for a complete overview of visualization grammars (and how they extend or wrap each other), and to Pu et al.' SIG [11] for a discussion on the design and evaluation questions around grammars.

While there are many dimensions that can be used to classify grammars, for the purpose of our work, we will focus on organizing them into two major groups: (1) general visualization grammars, designed to facilitate standard plotting tasks, and (2) domain-specific visualization grammars, designed to cater to more particular users, use cases, or tasks.

An important example of a visualization grammar aimed at general visualization tasks is Vega-Lite [4]. Created from Vega [12], Vega-Lite is a high-level grammar for interactive data visualizations. With it, users specify a mapping between data and the properties of graphical marks, as well as data and visual transformations, compositions, and interactions. When compared to D3 [2], a popular visualization library, Vega-Lite abstracts low-level aspects while maintaining enough expressivity to allow for chart customization. Such an approach has been proved useful, as highlighted by the number of interactive systems that leverage Vega-Lite [13], [14], its use in educational contexts [15], and visualization grammars inspired by it [5], [16].

In the other group, we have grammars designed to satisfy more specific use cases or domain requirements. For example, Li and Ma presented a series of grammars aimed at facilitating specific visual analytics tasks. With P4 [17], they presented a declarative approach for the specification of interaction visualization pipelines. They extended this notion with P5, aimed at facilitating the creation of progressive visualizations [18], and P6, for the integration of machine learning and visual analytics [19]. Shih et al. presented a grammar for the specification of volumetric visualization pipelines [20]. LYi et al. presented Gosling [21], specifically aiming the visualization of genomic data. The notion of grammars has also been applied to immersive visualizations. Lee et al. proposed Deimos [22], a grammar for morphs and transitions in immersive environments; Sicat et al. presented DXR [5], a grammar-based toolkit to facilitate the development of immersive analytics applications using Unity; and Butcher et al. presented VRIA [23] for web-based immersive visualizations. Park et al. [24] presented Atom for the description of unit visualizations to create novel chart forms. And Kim et al. [25] presented Cicero for responsive visualizations. Recently, we also proposed the Urban Toolkit [6], a grammar-based framework for urban visual analytics. While not exhaustive, the aforementioned works highlight the breadth and depth of visualization grammars and their applications.

III. The Vega-Lite Grammar

In this section, we provide a more in-depth overview of Vega-Lite, a high-level grammar for the creation of interactive visualizations. Vega-Lite stems from D3 [2] and Vega [26]. With the introduction of these different declarative approaches (D3 in 2011, Vega in 2015, and Vega-Lite in 2016), there was a

shift along the spectrum of expressivity: D3 and Vega are low-level solutions that offer fine-grained control for composing visualizations, while Vega-Lite is a higher-level grammar that abstracts many low-level implementation details. While Vega-Lite is not a replacement for D3 and Vega, it does facilitate the creation of visualizations through less verbose specifications. As an example of an abstraction leap, Vega-Lite does away with many of the events and data processing steps required by D3 and Vega, such as data joins and the need to explicitly specify visualization elements, such as axis and scales.

A. *Vega-Lite grammar*

A Vega-Lite visualization can be authored using a JSON file, with attribute-value pairs and arrays that specify different grammar elements. Next, we will cover the main elements of a Vega-Lite specification: data loading and transformation, marks and encodings.

Data loading and transformation. In the initial step, a `data` source must be specified. With Vega-Lite, data can be defined directly in the JSON file or specified through a URL. In our running example, we will use a dataset with noise complaints in New York City (NYC). As an example, we will use the following columns: **Created Date**, with the noise complaint creation date, **Complaint Type**, with the type of noise complaint, and **Borough**, with the NYC borough of the complaint. Loading the data can be done with:

```
{
  "data": {"url": "data/noise.csv"},
}
```

Marks. The data will then be visually encoded through `marks`, i.e., shapes that can have their properties (e.g., position, size, color) used to visually encode data. A `line` mark can be used for a line chart, or a `bar` mark for a bar chart:

```
{
  ...
  "mark": "line",
  ...
}
```

```
{
  ...
  "mark": "bar",
  ...
}
```

Encoding. To change their visual properties, users can use the `encoding` property, creating a mapping between data and encoding channels (e.g., x and y positions). For a line chart, two encodings need to be specified: `x` for time and `y` for aggregated counts:

```
{
  ...
  "x": {"timeUnit": "day", "field": "Created Date"},
  "y": {"aggregate": "count"}
  ...
}
```

In the previous example, given that the noise dataset has a column with time information (**Created Date**), we can also specify the temporal granularity through the `timeUnit` element. The result is shown in Figure 1 (a).

Similarly, for a bar chart, `x` can encode the position for categorical data and `y` their counts. To visualize a bar chart with the distribution of complaints per borough, we specify:

```
{
  ...
  "x": {"type": "nominal", "field": "Borough"},
  "y": {"aggregate": "count"}
  ...
}
```

The result is shown in Figure 1 (b). To visualize another dimension of the dataset, we can simply change the column mapped to a given channel:

```
{
  ...
  "x": {"type": "nominal", "field": "Complaint Type"},
  "y": {"aggregate": "count"}
  ...
}
```

The result is shown in Figure 1 (c). For a stacked bar chart, users can specify a `color` encoding. In our example, we can specify that the data mapped to the color channel will be the **Borough** column:

```
{
  ...
  "mark": "bar",
  "encoding": {
    "x": {"field": "a", "type": "nominal"},
    "y": {"field": "b", "type": "quantitative"},
    "color": {
      "field": "Borough",
      "type": "nominal"
    }
  }
  ...
}
```

The result is shown in Figure 1 (c). In Vega-Lite, user interactions, such as mouse clicks or drags, are handled through `select` elements. A selection will determine the event that triggers a selection and the resulting data subset. Importantly, a selection can be used to modify visual encodings. For example, use an encoding if the data is within the selection (e.g., on mouseover):

```
{
  ...
  "params": [
  {
    "name": "highlight",
    "select": {"type": "point", "on": "mouseover"}
  },
  {"name": "select", "select": "point"}
  ],
  "encoding": {
    ...
    "fillOpacity": {
      "condition": {"param": "select", "value": 1},
      "value": 0.3
    }
  }
  ...
}
```

Such elements can be seen as the building blocks to create not only interactive visualizations but also multi-view ones – i.e., a selection in one view will filter or highlight data in another view. In essence, Vega-Lite provides a simple and concise way to create maps between data and visual marks/channels. Time-consuming tasks, such as the creation of scales and axis, is abstracted by the use of smart defaults in the process that converts Vega-Lite to Vega specifications.

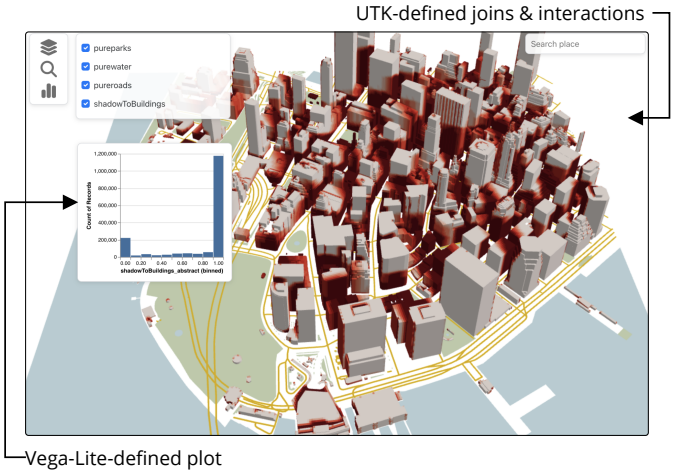

Fig. 2. Main components of a UTK visualization: UTK-defined data joins and interactions, and Vega-Lite-defined plots. UTK also supports different integration schemes between 2D-based plots and 3D-based maps (e.g., juxtaposed, embedded plots).

B. Extensions

Vega-Lite has been extended and wrapped by several grammars. For the purpose of our paper, four are particularly relevant: Gosling, DXR, VRIA, and Animated Vega. Gosling [21] is a grammar-based toolkit specifically built for genomics data visualization. It uses similar building blocks as Vega-Lite, such as `data`, `mark`, and encoding `channels`. Gosling is, however, designed to handle multi-scale data, common to genomics. As such, hierarchical structures are defined using `tracks` and `views`. Each `track` will specify how data is encoded; each `view` will be composed by a group of tracks and will specify their alignment and linking.

DXR [5] and VRIA [23] were specifically built for immersive data visualization and analytics. Again, similar to Vega-Lite, they make use of similar grammatical elements (e.g., `data`, `mark`, `encoding`), which will be interpreted into lower-level (and VR/AR appropriate) representations. In the case of DXR, specifications are interpreted into Unity prefabs, while in VRIA they are interpreted into a web-based application.

Lastly, Animated Vega-Lite [16] extended Vega-Lite to support animated visualizations. This is done by considering `time` as a new encoding channel, much like the aforementioned x, y, and color properties. On top of that, it makes use of selections (through the `select` element) to specify subsets of data points. While in Vega-Lite such selections are based on user input events, in Animated Vega-Lite selections react to timer events – i.e., a new selection can be specified based on the elapsed time of the animation.

IV. UrbanTK

The Urban Toolkit (UTK)[1] is a grammar-based tool developed to support a spectrum of users interested in urban visual analytics [6]. We have created UTK based on our extensive work creating urban visual analytics tools across domains and datasets [27]–[32]. One of the main pillars of UTK is the democratization of access to urban data with a primary focus on urban experts. Through the abstraction of many low-level tasks of data management and visualization, UTK can support the workflow of highly experienced scientists as well as users with little knowledge of urban computing. UTK was developed with three main design goals in mind: extensibility, reproducibility, and flexibility. Maximizing the user's ability to create custom visualizations, having a grammar-based system, and minimizing friction with the final user are some of the ways these design goals are actualized. More concretely, a grammar specification is interpreted to generate an entire visual analytics system that assists the study of urban data provided by the user. In its current version, UTK is available as a Python package and the application itself runs as a web system. The tool is publicly available at http://urbantk.org.

[1]UTK is available at: http://urbantk.org.

A. Capabilities

UTK's contains a host of functionalities to support urban visual analytics tasks. The components defined in the grammar specification are used to provide windows of investigation into the data. Broadly speaking, these components represent the data in either a 2D or 3D form (Figure 2). The 2D representations are specified using Vega-Lite [4]. UTK extends the Vega-Lite grammar to create interactive plots that can effectively visually both 2D and 3D data. To do so, UTK supports juxtaposed and embedded plots (as discussed in Mota et al. [33]). The main view is composed of several layers, such as parks, roads, and buildings. The map is where most of the interactions will take place through zooming, panning, rotating, clicking and hovering the layers. The interactions are used to highlight elements or embed 2D plots. Like previous grammars, specifications using UTK's grammar use JSON files.

The data loaded into UTK can come from the user or directly generated by the system. In both situations, the loading functionalities are offered through a Python API. If the user wants to provide the data, several formats are supported (e.g., GeoJSON, Shapefile, CSV), and the files are processed to create UTK-ready files. UTK also supports the automatic loading and parsing of OpenStreetMap data, as well as sunlight access simulations. Finally, UTK also allows for the construction of what-if scenarios by supporting the definition of custom operations between layers.

B. Grammar overview

A valid UTK grammar specification needs at least one `map` component that will define how the components are organized on the screen. The components then leverage `knots` that define how thematic (e.g., simulations, surveys, sensing data) and physical (e.g., buildings, parks, roads) data layers are linked together (i.e., joined) to produce 2D and 3D visualizations. The `knots` are the data sources in the grammar, and they can also be used to specify arithimetic operations between them. For example, computing a *what-if* scenario with the impact on

sunlight access of proposed buildings can be modeled through the difference between two knots.

C. Example: Neighborhood signatures

UTK can be used to tackle a variety of use cases, its flexibility and grammar-like nature allow it to be tailored according to the user's needs. In order to illustrate that, we can use UTK to investigate the multidimensional features of a city. To properly capture and explore the dynamics of urban environments, one must realize the inherent complexity of the task. Cities are composed of ever-changing sets of events and relationships that directly influence one another. For example, identifying the unique characteristics of each neighborhood to support planning and resource allocation requires the analysis of datasets across scales. Using multiple datasets from NYC Open Data (crime and noise reports; restaurants, parks, and subway locations; sky exposure; school quality reports, and taxi pickups) we can build a visualization using UTK to support neighborhood characterization. The first step is using the UTK's backend to transform the downloaded CSV and shapefiles into UTK-ready thematic and physical layers, respectively:

```
utk.csv_to_thematic("sky_exposure.csv", ...)
utk.shapefile_to_physical("chicago_zip.shp", ...)
```

The files generated can now be used inside the grammar specification. Background layers (i.e., water, parks) are loaded using a `knot` containing only a physical layer, and each dataset is mapped to a `knot` aggregating the data by the ZIP code level. For example:

```
{
  "id": "sky_by_zip",
  "integration_scheme": [
    {
      "spatial_relation": "contains",
      "in": {"name": "sky", "level": "coordinates"},
      "out": {"name": "zip", "level": "objects"},
      "operation": "avg",
      "abstract": true
    }
  ]
  ...
}
```

Inside the map component, a parallel coordinate plot is defined using Vega-Lite. The plot has a `linked` arrangement indicating that it will be plotted juxtaposed to the map. The knots are used on the map, and a `hover` flag is added to indicate the type of interaction – i.e., the ZIP codes are highlighted when lines in the chart are selected.

```
...
  "plots": [
    {
      "plot": {...},
      "knots": ["sky_by_zip", "noise_by_zip", ...],
      "arrangement": "linked",
      "interaction": "hover"
    }
  ],
...
```

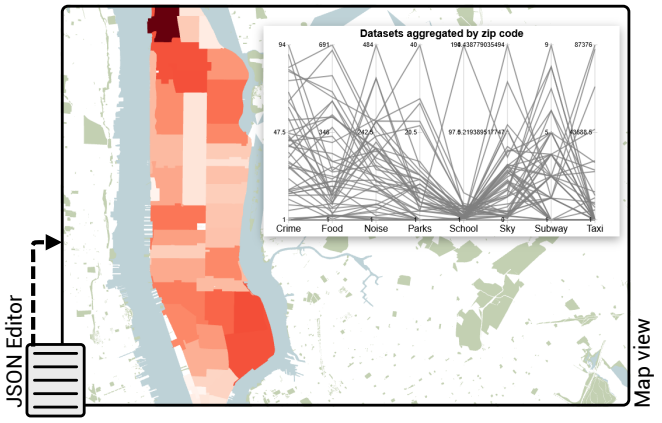

Fig. 3. Visualizing neighborhood signatures in Manhattan (NYC). The grammar is interpreted to generate a visualization where the distribution of noise complaints is displayed on the map and features from other datasets are displayed on the parallel coordinate plot.

The defined grammar is then interpreted to generate the visualization shown in Figure 3. Even such a relatively complex visualization can be specified in a very succinct manner using UTK's grammar. In this example, the JSON specification has fewer than 200 lines.

D. Limitations

Despite the applicability of the tool, UTK has some limitations that need to be considered. Large and high-resolution data can slow down join operations between physical and thematic layers, due to the use of the Geopandas library, which is not optimized for interactivity. Another limitation is regarding the scalability of temporal data: each time range must have its own knot, making it verbose and difficult to scale.

V. Conclusions

Since Vega-Lite, there has been an increase in the number of new visualization grammars being proposed by the visualization community. In the past three years alone, more than 20 new grammars have been proposed, tackling different visualization questions and domains [10]. Such a large number reflects the diversity of complex data tasks and use cases that are being considered in the development of these higher-level abstractions. There are, however, important research directions that must be considered in future work [10], [11]. In the context of map-based visualizations, Hogräfer et al. [34] and Ziegler and Chasins [35] make the case for the creation of grammars specifically designed for spatial tasks and data. Such developments would follow a trend to develop grammars to support more specific data tasks, such as in genomics [21] and machine learning [19]. The increase in the number of domain-specific grammars also leads to questions related to their usability. Establishing proper evaluation methodologies, especially considering domain experts' workflows, is an important step to establish the validity of previous approaches and have a more grounded understanding of domain-specific gaps and needs.

ACKNOWLEDGMENTS

This collaboration was partly funded by the University of Illinois' Discovery Partners Institute (DPI), CNPq (316963/2021-6), and FAPERJ (E-26/202.915/2019, E-26/211.134/2019).

REFERENCES

[1] H. Mei, Y. Ma, Y. Wei, and W. Chen, "The design space of construction tools for information visualization: A survey," *Journal of Visual Languages & Computing*, vol. 44, pp. 120–132, 2018.

[2] M. Bostock, V. Ogievetsky, and J. Heer, "D³ data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[3] C. Reas and B. Fry, "Processing.org: A networked context for learning computer programming," in *ACM SIGGRAPH 2005 Web Program*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 14–es.

[4] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-lite: A grammar of interactive graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017.

[5] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister, "DXR: A toolkit for building immersive data visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 715–725, 2019.

[6] G. Moreira, M. Hosseini, M. N. A. Nipu, M. Lage, N. Ferreira, and F. Miranda, "The Urban Toolkit: A grammar-based framework for urban visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, 2024, accepted, to appear.

[7] J. Bertin, *Semiology of Graphics*. University of Wisconsin Press, 1983.

[8] L. Wilkinson, *The Grammar of Graphics*. Springer, 1999.

[9] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu, "Critical reflections on visualization authoring systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 461–471, 2020.

[10] A. M. McNutt, "No grammar to rule them all: A survey of json-style dsls for visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 160–170, 2023.

[11] X. Pu, M. Kay, S. M. Drucker, J. Heer, D. Moritz, and A. Satyanarayan, "Special interest group on visualization grammars," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery, 2021.

[12] A. Satyanarayan, K. Wongsuphasawat, and J. Heer, "Declarative interaction design for data visualization," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 669–678.

[13] A. Satyanarayan and J. Heer, "Lyra: An interactive visualization design environment," *Computer Graphics Forum*, vol. 33, no. 3, pp. 351–360, 2014.

[14] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager 2: Augmenting visual analysis with partial view specifications," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2648–2659.

[15] B. Naimipour, M. Guzdial, and T. Shreiner, "Engaging pre-service teachers in front-end design: Developing technology for a social studies classroom," in *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1–9.

[16] J. Zong, J. Pollock, D. Wootton, and A. Satyanarayan, "Animated VegaLite: Unifying animation with a grammar of interactive graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 149–159, 2023.

[17] J. K. Li and K.-L. Ma, "P4: Portable parallel processing pipelines for interactive information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 3, pp. 1548–1561, 2020.

[18] ——, "P5: Portable progressive parallel processing pipelines for interactive data analysis and visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1151–1160, 2020.

[19] ——, "P6: A declarative language for integrating machine learning in visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 380–389, 2021.

[20] M. Shih, C. Rozhon, and K.-L. Ma, "A declarative grammar of flexible volume visualization pipelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 1050–1059, 2019.

[21] S. LYi, Q. Wang, F. Lekschas, and N. Gehlenborg, "Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 140–150, 2022.

[22] B. Lee, A. Satyanarayan, M. Cordeil, A. Prouzeau, B. Jenny, and T. Dwyer, "Deimos: A grammar of dynamic embodied immersive visualisation morphs and transitions," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023.

[23] P. W. S. Butcher, N. W. John, and P. D. Ritsos, "Vria: A web-based framework for creating immersive analytics experiences," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3213–3225, 2021.

[24] D. Park, S. M. Drucker, R. Fernandez, and N. Elmqvist, "Atom: A grammar for unit visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 12, pp. 3032–3043, 2018.

[25] H. Kim, R. Rossi, F. Du, E. Koh, S. Guo, J. Hullman, and J. Hoffswell, "Cicero: A declarative grammar for responsive visualization," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022.

[26] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive Vega: A streaming dataflow architecture for declarative interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 659–668, 2016.

[27] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, "Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2149–2158, 2013.

[28] F. Miranda, H. Doraiswamy, M. Lage, K. Zhao, B. Gonçalves, L. Wilson, M. Hsieh, and C. T. Silva, "Urban Pulse: Capturing the rhythm of cities," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 791–800, 2017.

[29] H. Doraiswamy, E. Tzirita Zacharatou, F. Miranda, M. Lage, A. Ailamaki, C. T. Silva, and J. Freire, "Interactive visual exploration of spatio-temporal urban data sets using urbane," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1693–1696.

[30] H. Doraiswamy, J. Freire, M. Lage, F. Miranda, and C. Silva, "Spatio-temporal urban data analysis: A visual analytics perspective," *IEEE Computer Graphics and Applications*, vol. 38, no. 5, pp. 26–35, 2018.

[31] F. Miranda, H. Doraiswamy, M. Lage, L. Wilson, M. Hsieh, and C. T. Silva, "Shadow Accrual Maps: Efficient accumulation of city-scale shadows over time," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 3, pp. 1559–1574, 2019.

[32] S. Chen, F. Miranda, N. Ferreira, M. Lage, H. Doraiswamy, C. Brenner, C. Defanti, M. Koutsoubis, L. Wilson, K. Perlin, and C. Silva, "UrbanRama: Navigating cities in virtual reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4685–4699, 2022.

[33] R. Mota, N. Ferreira, J. D. Silva, M. Horga, M. Lage, L. Ceferino, U. Alim, E. Sharlin, and F. Miranda, "A comparison of spatiotemporal visualizations for 3D urban analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1277–1287, 2023.

[34] M. Hogräfer, M. Heitzler, and H.-J. Schulz, "The state of the art in map-like visualization," *Computer Graphics Forum*, vol. 39, no. 3, pp. 647–674, 2020.

[35] P. Ziegler and S. E. Chasins, "A need-finding study with users of geospatial data," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023.