Reasonable Abstractions: Semantics for Dynamic Data Visualization

Joseph A. Cottam, and Andrew Lumsdaine

Abstract— Chi showed how to treat visualization programing models abstractly. This provided a firm theoretical basis for the data-state model of visualization. However, Chi’s models did not look deeper into fine-grained program properties, such as execution semantics. We present conditionally deterministic and resource bounded semantics for the data flow model of visualization based on E-FRP. These semantics are used in the Stencil system to move between data state and data flow execution, build task-based parallelism, and build complex analysis chains for dynamic data. This initial work also shows promise for other complex operators, compilation techniques to enable efficient use of time and space, and mixing task and data parallelism.

Index Terms—Dynamic data, Semantics.

1 INTRODUCTION

Formal models provide a basis for evaluating frameworks independent of implementation details and provide a strong basis for the evaluating and implementation of proposed features. As workloads and computational environments grow in complexity, such models will be essential for future visualization frameworks. Our work establishes a model with the following five capabilities: (1) dynamic data handling, (2) deterministic execution, (3) bounded resource consumption, (4) consistency in results and (5) interoperability. Informally, dynamic data handling means that data may change while analysis is being performed. Deterministic execution indicates that only the inputs values matter in determining the resulting visualization (not timing artifacts in the system). Bounded resource consumption implies that resource use is a reasonable function of the inputs. Consistency indicates that the visualizations produced represent all of the effects of the data presented. Finally, interoperability is the ability to use algorithm implementations with little Stencil-specific work. With these five properties, a wide variety of visualizations can be addressed. Our exploration has lead to a formalization of the data flow model of visualization in terms of Event-Driven Functional Reactive Programming (E-FRP). We have implemented the Stencil visualization framework in accordance with E-FRP semantics and demonstrate a performance advantage when the semantics are applied to find optimization opportunities.

2 RELATED WORK

The data-flow model of visualization as described by Chi [2] is an expression of the dependencies between data transformations. Chi showed that the data-flow model could be exchanged with a data-state model in a structured fashion. This enabled the prior experience with data-flow models to be applied to the relatively new data-state model and “information spreadsheets.” However, the models presented did not include order of operation information, making execution questions unapproachable.

Functional-Reactive Programming (FRP) is a formalism for constructing synchronous data-flow networks. Formal semantics for FRP show it is deterministic [5]. The core concepts of FRP are (1) behaviors (time-varying values), (2) events (values a specific time) and (3) combinators to construct new events and behaviors. Pure FRP makes two broad assumptions: (1) that the system will ‘react’ to dynamic data and (2) that operations are pure functions with time parameters taken from a continuous space. E-FRP is based on FRP, but replaces the second assumption with discrete notion of time and an assumption that behavior values only change when an event occurs. This enables a direct, resource-bounded implementation [6]. With E-FRP, stateful behaviors can include non-functions as long as state changes logically occur between events. This provides a direct means for interoperating with programs not specifically coded for E-FRP. E-FRP is deterministic and resource bounded insofar as the operations employed are as well. This satisfies four of the five desired properties; consistency is provided by treating rendering as response to an event in an E-FRP program.

There are four parts to an E-FRP visualization system, depicted in Figure 1. First, the Sequencer gathers external events and selects a ‘next’ event. Second, the Dispatcher accepts the ‘next’ event from the sequencer to process in the analysis engine. Third, the analysis engine produces a store full of name/value pairs that are processed by the dispatcher, sequencer and display system.

3 STENCIL: AN EFRP BASED VISUALIZATION SYSTEM

As a synchronous network language, E-FRP fits closely with the data-flow model of visualization. Informally, each link of a data-flow net-
work becomes a event type in the E-FRP program. Transformations operations correspond to behavior definitions. When an event occurs, behaviors execute according to the E-FRP semantics. E-FRP semantics indicate that independent behaviors may be evaluated in any order. Behaviors may see changes to behaviors earlier in the network only if there is no backward dependency. Otherwise, behaviors in mutually dependent groups can execute in any order, but results are still deterministic because state changes only occur between events. To force a specific order on loops, synthetic events are constructed and loop breaking is be performed. (This is not standard E-FRP, which uses a now/later tag to make updates visible at different times. The now/later tagging is (1) not sufficiently general to break all loops and (2) not know to be deterministic or resource bounded.) Loop-breaking constructions are shown in Figure 2. By constructing synthetic events, the order of updates is made explicit in the network.

In the Stencil implementation, interoperators with existing algorithms is facilitated by wrapping the existing algorithm in a new interface. The Stencil system has operators based on JUNG and UMD TreeMap implementations. Each requires about 100 lines of wrapper code around the 10K+ code bases. The wrapper code primarily buffer state changes so they logically occur between events.

Working with dynamic data makes timing rendering a significant issue. A consistent visualization is one where the rendered results represents either all of the effects of an event or none of them. Simply waiting for all data to be processed may not be possible as the streams with no known bound (i.e., mouse input or network traffic). Visualization frameworks typically provide consistency by literally making analysis and rendering mutually exclusive phases [3, 4]. E-FRP enables this discipline to be logically implemented by making rendering a response to an event, without requiring the runtime to be so literal. Rather than mutual exclusion, the Stencil system uses persistent data structures to render concurrently with analysis. This increases the the number of data points that can be processed per unit time.

With E-FRP semantics and a simple effect system (analysis operations are labeled as either ‘function’, ‘reader’ or ‘writer’), Stencil can efficiently handle dynamic data analysis. Particularly, the semantics and type system indicate when batch operation groups are possible and how to concurrently perform render and analysis. We compared Stencil performance to Prefuse performance on a simple quadrant-filling layout. Prefuse was selected because it is a mature framework implemented in Java and uses a column store. Stencil is also implemented in Java and uses a column store. Each new data-point requires all other points to move, introducing a worst-case scenario for computation before rendering. A quadrant filling layout was used so all data points would remain on the screen, but no zoom/pan calculations are required. To mediate framework differences, rendering was triggered Prefuse on a timer that matched the frame-rates seen in the Stencil. Average runtimes over five executions are presented in Figure 3. For all but trivial data sizes, Stencil enjoys sizable a performance advantage, despite being untyped and interpreted while Prefuse is typed and compiled. The advantage is achieved through Stencil’s application of E-FRP semantics and abstract encoding of consistency.

Fig. 2: Interleaved cycle broken by introducing synthetic events that are consumed and ordered by the dispatcher. If there are no cycles, such as between C and D, the dispatcher can be be bypassed.

Fig. 3: Performance of Prefuse and Stencil on a quadrant-filling layout with dynamically updated data.

4 OPEN QUESTIONS AND CONCLUSIONS

The presented semantics enable program analysis at a more detailed level than Chi’s treatment. However, they become more useful as additional program meta-data is brought into play. For example, the performance results presented rely on a simple effect system. We are currently investigating other program meta-data to apply to create efficient visualization systems. Information about input stream structure (e.g., that values are sorted or unique) may enable more efficient memory use. More detailed information about analysis operators properties also has obvious applications (i.e., it is commutative, or should race conditions be considered benign). Operator groups may enable specification specific implementation selection.

E-FRP semantics have revealed useful to establish properties of the Stencil system (like consistency and resource bounds). However, they are not necessarily the “best” semantics for a visualizations system. E-FRP semantics assume that strong memory consistency is desired and thus limit data parallelism. Other semantic models (such as a process calculus) may provide other insights and be more suitable to other types of workloads, or novel hardware configurations.

Even retaining E-FRP semantics, other visualization abstractions may be formalized and then optimized. Cross-tab/pivot tables for summarization, efficiently scheduling recalculations (for dynamic data), axis/legend creation and data-based parallelization are all being investigated. Each presents a non-trivial problem for Visualization and the E-FRP semantics are guiding an efficient implementation that preserves the properties enjoyed by the basic framework.

The semantic framework of Stencil enables it to reason about the visualization programs presented in valuable ways. As visualization workloads become more demanding and compute environments, framework semantics will become more important. E-FRP provides a basis for deterministic, resource bounded computation resulting in internally consistent visualizations.

REFERENCES