

Semantic and Visual Encoding of Diagrams

Dartmouth Computer Science Technical Report TR2009-654¹

Version of August 1, 2009

Gabriel A. Weaver

Department of Computer Science/Dartmouth PKI Lab Dartmouth College
Hanover, New Hampshire USA

Abstract. Constructed geometric diagrams capture a *dynamic* relationship between text and image that played a central role in ancient science and mathematics. Euclid, Theodosius, Ptolemy, Archimedes and others constructed diagrams to geometrically model optics, astronomy, cartography, and hydrostatics. Each derived geometric properties from their models and interpreted their results with respect to the model's underlying semantics. Although diagram construction is a dynamic process, the media in which these works were published (manuscripts and books) forced scholars to either view a snapshot of that process (a static image) or *manually* perform the entire construction. Mainstream approaches to digitization represent constructed diagrams as they appear in print, as *static* images. Such representations fail to capture the *dynamic* nature of constructed diagrams and so we designed and implemented a *computational* framework for *dynamically* interacting with them. Our architecture for representing, retrieving, and interacting with diagrams has already been used to produce a publicly-available, archival-quality digital corpus of diagrams for the *Archimedes Palimpsest Project*, establishing our approach's viability in the real world. After using our system to study diagrams in Archimedes, we discuss the generality of our approach and its application to other domains including circuit design, software engineering, and patent databases.

1. Introduction

For centuries, mathematicians and scientists illustrated their claims by constructing a geometric diagram and arguing about its properties. Euclid's *Elements* relies upon diagram construction to illustrate the veracity of mathematical claims. Christopher Clavius, the primary architect of the Gregorian calendar, boasts of new illustrations in the title of his translation of Theodosius' *Spherics* [4]. Ptolemy's *Treatise on the Planisphere* layers several different geometric projections of the celestial sphere to demonstrate astronomical properties of the earth. Archimedes uses constructed geometric diagrams to prove claims about subjects ranging from pure mathematics to hydrostatics.

¹This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

The author's work was also supported by a non-residential fellowship from the Center of Hellenic Studies, Harvard University and a stipend from the Archimedes Palimpsest Project.

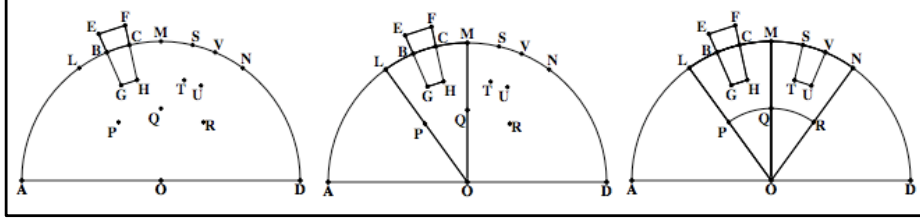


FIGURE 1. Archimedes' diagram construction for book I, proposition 3 of *On Floating Bodies*.

For example, in book 1, proposition 3 of *On Floating Bodies*², Archimedes demonstrates that if a solid is submerged into a fluid and their density is the same, then that solid will neither project above the surface nor sink lower. The proof by contradiction he describes within the text, references a diagram constructed using the following sequence of steps paraphrased here and illustrated in Figure 1.

1. We are given a fluid surface
2. If possible, immerse a solid EFGH into that fluid so that part of it (EBCF) projects above the surface. Let the solid have the same density as the fluid.
3. Draw a plane through O, the center of the earth, and the solid, cutting the surface at the curve ABCD.
4. Construct a pyramid around the solid whose base is a parallelogram at the surface and whose vertex is O.
5. Define a sphere PQR with center O so that it is contained within the fluid and below GH (the bottom of the solid).
6. Conceive another pyramid in the fluid with vertex O, contiguous, equal, and similar to the former pyramid.
7. Let STUV be a part of the fluid within the second pyramid, equal and similar to the part of the solid below the surface.

Properties of Constructed Diagrams The preceding construction illustrates four fundamental semantic and two fundamental visual properties of constructed diagrams. First, a diagram's semantics are not evident from looking at it. The concept modeled by the trapezoid EFGH is unknown until one reads the text and discovers it represents a solid, a non-geometric object. Secondly, the semantics of a geometric object may change over the sequence of construction events. For example, the geometric point labeled O symbolizes a center-of-earth, a vertex of a pyramid, and the center of a sphere in various stages of diagram construction. This phenomenon is referred to by Glasgow, Narayanan, and Chandrasekaran as *symbolic annotation* upon the diagram [8]. Third, parts of a diagram may be constructed from previously defined objects³. For example, the curve ABCD results

²Alternatively, we may also reference this passage using the following notation: *On Floating Bodies* 1.3.

³This is closely related to the notion of *emergent properties* [8], properties of a diagram which arise from combining existing entities.

from intersecting a fluid surface and a plane. Fourth, these diagrams are modular and leverage previous constructions. *Floating Bodies* 1.5 reuses the constructed diagram of proposition 3, changing only the density of solid EFGH.

The constructed diagram for proposition 3 also reveals two important visual properties of constructed diagrams. First, geometric models of these semantics may display the logically impossible. In the second step of the construction, although solid EFGH has the same density as the fluid immersed in it, it is drawn projecting above the fluid surface as if it were lighter per unit volume than the fluid. Secondly, different editions of a work may present constructed diagrams differently. For example, Heath's edition [9] of proposition 3 may differ from other versions in terms of labeling, line styling, perspective, and geometric configuration.

Although print is the traditional medium of publication for this kind of diagram, none of its observed semantic and visual properties are readily apparent without a careful reading of the text. Static images are inadequate for understanding the meaning of a diagram which may change over the course of its construction. Furthermore images fail to capture the conceptual relations between components of diagrams and already-defined entities or previous propositions. Even though images capture presentational information, without text, they do not reveal whether a diagram illustrates the logically impossible. Moreover they do not provide any mechanism to indicate whether two geometric arrangements share the same construction process and are thereby, we claim, semantically equivalent.

We argue that static, two-dimensional images are insufficient for encouraging the active exploration and comprehension of constructed diagrams. Furthermore, we claim that digitizing diagrams offers scholars an opportunity to augment their claims with quantitative data about how concepts and ideas are *actually* visualized in real-world data.

Our Contributions to Constructed Diagrams We designed and implemented the *Semantic and Visual Encoding of Diagrams (SaVED)* language to directly address the shortcomings of static, two-dimensional images. We claim that SaVED diagrams facilitate traditional, manual approaches to the study of diagrams and introduce entirely new methods to understand their usage. Our contributions to digital diagrams focus on their identification, representation, and manipulation.

Identification Our approach identifies the geometric components of constructed diagrams as well as their underlying semantics. We developed *SaVED Uniform Resource Names (SaVED-URNs)*—machine-actionable, human-readable strings capable of identifying an entire corpus of diagrams, multiple editions of a single diagram, or even individual geometric or semantic components of a diagram. In the example above we can identify the trapezoid EFGH or the solid which it symbolizes with SaVED-URNs. Since text describes the semantics of constructed diagrams, we adapted our previous work in identifying arbitrary sections of text. Just as we used hierarchical, human-readable, machine-actionable reference strings called

Canonical Text Services Uniform Resource Names (CTS-URNs) to perform computations on Classical Greek texts [18, 22], so did we develop SaVED-URNs to identify arbitrary semantic or presentational sections of a constructed diagram.

Our identification scheme enables computers to process the semantic and visual aspects of diagrams; computationally speaking, static images are just blobs of data. Static images do not allow computers to readily reference individual geometric components of a diagram much less let them resolve those components to an underlying concept. SaVED-URNs let people and machines reference arbitrary sections of a diagram; this allows people and machines to choose when and how to display components of a diagram. For example, using SaVED-URNs, one could write an application to highlight all geometric primitives that represent a certain concept (perhaps highlighting all arcs and circles that represent spheres). We claim that machine-actionable references to semantic and visual components of a diagram enhances a scholar’s ability to interact with diagrams.

Representation SaVED diagrams represent the semantic and presentational aspects of a constructed diagram with two layers whose objects are associated by a common label. Euclid, Theodosius, Ptolemy, and Archimedes all used labels to associate semantic information in text with visual information in diagrams. Both layers of a SaVED diagram include several features of high-level programming languages including typecasting (for changing the semantics of an object) and parameterized subroutines (for using previous constructions in new diagrams).

SaVED’s double layers satisfy our expectations for digital diagrams. Diagram semantics are made *explicit* and are encoded in a high-level programming language. The presentational layer, which interfaces with a complete graphics pipeline, can accommodate the variety of presentational styles attested to within the historical record. Since these layers are loosely coupled via common identifier, scholars are free to actively explore alternate, semantics-preserving, presentations of diagrams. Compiling SaVED diagrams into *Scalable Vector Graphics (SVG)* ensures a high-quality, zoomable presentation whose parts are semantically identified with SaVED-URNs. The symbol tables generated by compilation introduce the possibility of querying diagrams by the meaning and presentation of their components. This opens up entirely new lines of research to gather statistics of how diagrams are *actually* used within a corpus.

Manipulation Computing on constructed diagrams provides scholars and the general public with access to more, higher-quality tools for interacting with constructed diagrams and exploring their usage. Algorithms processing a corpus of SaVED diagrams might index them for search, compile them into interactive SVG displays, generate PDFs suitable for printing, enable scholars to annotate their components, or count the many ways in which a concept is displayed. The output of these algorithms may be regenerated as necessary and may even be useful for producing quantitative data for augmenting traditional scholarly argument.

This Paper This paper is organized as follows. Section 2 describes the real-world, archival requirements of the *Archimedes’ Palimpsest Project* and argues

their relevance for digital collections of diagrams in general. Section 3 introduces this architecture and illustrates how it captures our six properties of constructed diagrams and meets real-world requirements. We apply our architecture to studying the diagrams in Archimedes' *Floating Bodies, Book I* in Section 4 and introduce new computational techniques for *quantifying* how diagrams were *actually used*. Section 5 reviews related work. Section 6 discusses future research, focusing on the generality of our approach and its application to other domains. Section 7 concludes.

2. Addressing Real-World, Archival Needs

Recognizing the potential value of semantically computing on multiple editions of diagrams or their individual components, the *Archimedes Palimpsest Project* contacted us to encode the diagrams of the first book of *On Floating Bodies*. The project was finishing a 10 year effort to recover text from a palimpsested⁴ manuscript of Archimedes, employing state-of-the-art imaging techniques ranging from X-ray fluorescence, to ultraviolet photography to narrow-band imaging. Working with experts around the world including RIT, Boeing, the Stanford Linear Accelerator Center (SLAC), and Harvard's Center for Hellenic Studies (CHS), the project's primary objective was to deliver an archival-quality dataset of manuscript images along with digital corpora of the text and diagrams contained therein. As a secondary goal, the project also desired a platform which would allow end users to actively explore the properties of the diagrams, eventually in the context of the other datasets.

We originally developed SaVED as an encoding format for constructed diagrams, the analogue of the *Text Encoding Initiative's (TEI) Guidelines* [3] for diagrams rather than text. Over the course of five years, we had experimented with various syntaxes of diagram markup languages for Euclid, Theodosius, and Ptolemy⁵. Addressing the requirements of the *Archimedes Palimpsest Project* transformed SaVED from a classroom exercise into a mature programming language for creating digital corpora of archival-quality diagrams.

The *Archimedes Palimpsest Project* addressed the long-term utility and maintainability of their dataset by requiring *open formats*. Open formats require that the mode of presentation of data is transparent and or the specification of the data format is publicly available [15]. Data in an open format can either be encoded transparently, and thereby readable in any text editor, or encoded in binary, with instructions on how to decode supplied in a specification. Open formats ensured that the diagram and text corpora would be independent of a particular application or platform and that their syntax could be reliably decoded in a documented manner. Open formats also ensure that the data can be transformed into alternative formats that leverage new technologies. For example, the Perseus Project, a

⁴A palimpsest is a manuscript whose original writing has been rubbed clean and new text written in its place.

⁵This experience is another reason the *Archimedes Palimpsest Project* contacted us.

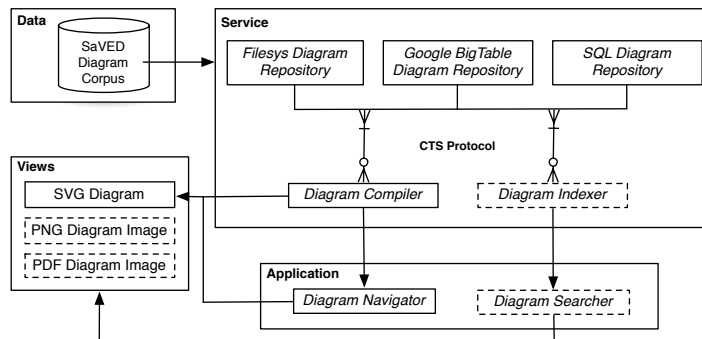


FIGURE 2. Our architecture for computing with constructed diagrams. (Dashed modules not yet implemented)

digital library for the Classics encoded numerous Greek and Latin texts in SGML in the 1980s and 1990s. Since SGML was an open format, they could migrate their data to XML and process it using XML-based tools, adding value to their dataset.

3. Our Architecture for Computing with Constructed Diagrams

We designed, implemented, and delivered an architecture for computing with constructed diagrams which captures the six fundamental properties of constructed diagrams while satisfying the real-world needs of the *Archimedes Palimpsest Project*. Figure 2 illustrates the design which we will now consider.

Our architecture consists of three primary layers: data, service and application. For the data layer, we constructed a SaVED-encoded corpus of diagrams for Archimedes’ first book of *On Floating Bodies*. Using the service layer, we retrieve SaVED diagram encodings using the *Diagram Repository* and render them using our *Diagram Compiler*. Our *Diagram Navigator* allows people to actively navigate the step-by-step construction of diagrams, providing a dynamic experience similar to that of Figure 1. Currently, we render all diagrams into an SVG format, allowing us to generate high-quality presentations whose components are semantically identified using SaVED-URNs.

Building a Diagram Corpus Our SaVED-encoded diagram corpus for Archimedes’ *On Floating Bodies, Book I* captures the six fundamental properties of constructed diagrams within an open format to increase their long-term utility and maintainability.

SaVED’s high-level programming language features—a type system, operators, and subroutines—enable scholars to explicitly encode the four semantic properties of constructed diagrams. Consider the third diagram whose construction we discussed in Section 1. That diagram visually encodes relations between many different types of objects including the *fluid surface*, *solid*, *pyramid*, and *point*.

Some of these objects were defined by Euclid; in *The Elements, Book I*, Euclid defines the point for example. Other objects, like the *fluid surface* are introduced by Archimedes. Furthermore, these technical terms may possess slightly different meanings depending upon the author. Euclid's circle is a plane figure *contained* by a line whereas others may use circle to mean only its circumference. Given that the sense of a technical term may vary depending upon its context, SaVED qualifies its semantic types like `fluid_surface`, `solid`, `pyramid`, and `point` with namespaces indicating its source definition or attestation. For example, we encode the *center-of-earth O* in the third proposition as:

```
<tlg0552.tlg008:center_of_earth id='o' />
```

where `tlg0552` is the *Canonical Text Services (CTS)* identifier for Archimedes in the *Center for Hellenic Studies (CHS)* namespace for Greek literature [20]. Similarly, `tlg008` is the CTS identifier for *On Floating Bodies*. SaVED explicitly encodes the semantics of each referenced component of the diagram and thereby captures the first semantic property of constructed diagrams.

The SaVED type system also implements typecasting, in which a declared variable is associated with another type. Continuing our example from above, Archimedes uses *center-of-earth O* as the center of a sphere, PQR. Since O is already declared, we can associate it with a new type using the `ref` attribute instead of the `id` attribute.

```
<tlg0552.tlg008:center_of_sphere ref='o' />
```

Our typecasting syntax allows us to easily associate multiple semantic types with the same identifier⁶. This gives SaVED a straightforward mechanism for encoding the changing semantics of a diagram component over a construction sequence, the second semantic property of constructed diagrams.

In addition to types, SaVED also provides a set of operators for producing objects from already-declared variables. SaVED's semantic or presentational operators take zero or more variables as input and produce zero or more variables as output. For example, Archimedes defines the curve ABCD in terms of a fluid surface and a plane. SaVED explicitly captures this dependency when encoding ABCD.

Notice that since Archimedes never labels the fluid surface or plane surface being intersected, we create a label name, using the convention of prefixing our name with `u1` to indicate an object used in the construction but unlabeled in the

⁶More generally typecasting allows us to retrieve and operate on different representations of the same object. As observed by D. Neel Smith, a Professor of Classics at Holy Cross, this mechanism may prove important to scholars in the humanities who already use a system of common identifiers for their objects but study and debate the various properties of those objects

```

<tlg1799.tlg001:intersection>
  <dialog:emergent_primitives>
    <tlg0552.tlg008:curve id='abcd' />
  </dialog:emergent_primitives>
  <tlg0552.tlg008:fluid_surface ref='ul_abcd' />
  <tlg1799.tlg001:plane_surface ref='ul_abcd' />
</tlg1799.tlg001:intersection>

```

text. SaVED's semantic and presentational operators explicitly encode how objects are constructed from previous objects, satisfying the third semantic property of these diagrams.

SaVED also implements parameterized subroutines, given zero or more parameters, subroutines return a result. We argue that diagram constructions may be viewed as a kind of subroutine: given zero or more objects, perform a sequence of operations on them to produce a new object. In fact, we argue that *citation is the operation by which mathematical argument becomes possible*. The growth of knowledge depends upon the ability to resolve references to prior information and apply it to new contexts. As mentioned earlier, in the 5th proposition, Archimedes reuses the construction from the 3rd proposition, changing the density of one of the solids diagrammed. He also references the solid as EGHF rather than as EFGH as he did in proposition 3. SaVED captures the relationship between these two propositions as parameters to a subroutine instantiating the third construction. In the semantic encoding, we declare an instance of the third construction, `c3`. We then associate the density of the fluid `ul_abcd` from proposition 3 with the density of the current fluid and derive the density of the solid `eghf`.

```

<dialog:construction_instance id='c3'
  ref='urn:saved:tlg0552.tlg008.logic01:3' />
<tlg0552.tlg008:density id='ul_abcd_den'
  ref='c3:ul_abcd_den' />...
<tlg1799.tlg001:less_than>
  <dialog:emergent_primitives>
    <tlg0552.tlg008:density id='ul_eghf_den' />
  </dialog:emergent_primitives>
  <tlg0552.tlg008:density ref='ul_abcd_den' />
</tlg1799.tlg001:less_than>

```

Since our SaVED encoding represents the density of the solid as a percentage of the fluid surface's density, we simply associate a new value with the presentational information for the density `ul_abcd_den` and use it to instantiate the presentational information for the third construction.

```

<epix7:scalar id='ul_efgh_den'
  diapre:param_id='3'>0.7</epix:scalar>
<diapre:diagram_instance id='c3'
  ref='urn:saved:tlg0552.tlg008.heath:3'>
  ...
  <epix:scalar ref='ul_efgh_den'
    diapre:param_ref='3'/>
  ...
</diapre:diagram_instance>

```

SaVED models the modular nature of constructed diagrams using subroutines, explicitly encoding their inter-construction dependencies and capturing the fourth fundamental property of diagrams.

While SaVED has several features in common with traditional programming languages that allow it to represent the semantic properties of diagrams, SaVED also has unique features for capturing the visual properties of constructed diagrams. As mentioned before, diagrams may display the logically impossible. The solid EFGH in proposition 3 has the same density as the fluid in which it is immersed and yet it appears projecting above the water. SaVED represents this contradiction with a two-layered approach. Although the semantic layer declares solid `efgh` with the same density as the fluid, `ul_abcd_den`, the presentational layer defines the midpoint of trapezoid `efgh` as a function of density so that trapezoid `efgh` always projects above the curve `abcd`. In this manner SaVED diagrams encode diagrams that display the logically impossible, the first observation on visual properties of constructed diagrams.

Using *SaVED styles*, diagram components may be rendered with a variety of presentational attributes. For example, we might change how a label is displayed or alter the color of a geometric primitive. The labels of diagrams within the palimpsest edition of *Floating Bodies* use Ancient Greek letters whereas those in Heath's English translation use Roman letters. SaVED diagrams can represent both of these possibilities. For example, to display the point `c` as a Greek letter γ , one would simply associate the Unicode (UTF-8) representation of γ with the text label for point `c`. Since the letter γ appears in ancient and modern Greek, we disambiguate its usage with a *language-encoding triplet* [19] which encodes the label's language, writing system, and digital representation. The triplet `grc-Grek-x-utf8` means that the label γ comes from the Ancient Greek (`grc`) alphabet, is written using literary Greek orthography (`Grek`), and is digitally represented using UTF-8. In the example below, we color the point `c` red and display its label as the Ancient Greek letter γ .

In addition to *SaVED styles*, we implemented a graphics pipeline to accommodate the variety of perspectives attested to by the historical record. The constructed diagrams in *Floating Bodies, Book I* represent, at most, a *single* two-dimensional projection of three-dimensional objects. We witness this in the third

```

<diapre:style ref='c'>
  <diapre:label_text>  $\gamma$  </diapre:label_text>
  <diapre:label_size>8pt</diapre:label_size>
  <diapre:label_triplet>grc-Grek-x-utf8</diapre:label_triplet>
  <diapre:color>red</diapre:color>
</diapre:style>

```

proposition where Archimedes informs us that the curve ABCD represents a slice of a three-dimensional fluid surface. In such cases, a graphics pipeline is overkill as the construction occurs on a single projection drawn on the plane of the paper. However, we designed SaVED so that it would eventually handle the diagrams of Ptolemy’s *Planisphere* where diagrams consist of multiple two-dimensional projections of the same three-dimensional object stacked on top of one another. Nonetheless, even in the context of Archimedes, we believe the ability to change the “camera” angle from which a diagram is viewed a useful feature for actively exploring diagrams.

We designed SaVED so that we could encode and explore *semantically-equivalent* diagrams, diagrams sharing the same construction but which have different geometric arrangements, regardless of labeling and styling. The third proposition of Archimedes’ *Floating Bodies* states that the sphere PQR must be contained in the fluid surface and must be below the base of solid EFGH. However, the static nature of printed diagrams means that only one configuration of PQR may be displayed although there are multiple, semantically-valid ways in which PQR could be drawn. Similarly, some editions of Archimedes’ *Sphere and Cylinder* illustrate sides of a polygon as straight lines, while other editions, like those from Archimedes’ palimpsest, present them as circular arcs [14]. For this reason, we designed SaVED so that diagrams resulting from the same construction always used the same semantic layer, differing only in the presentational layer.

The presentation may vary in terms of *style*, or it may differ in configuration, either way, SaVED allows us to express the variation and allows us to transform diagrams’ presentations without violating the constraints of the construction process.

We claim that SaVED captures the six fundamental properties of constructed diagrams in an open format, increasing the long term utility and maintainability of encoded diagrams. The semantic and presentational languages are expressed in an XML format whose written specification and grammar are publicly available⁸. As such, SaVED is an open, non-binary format for encoding constructed diagrams. Furthermore, the entire diagram corpus for Archimedes’ *On Floating Bodies* is publicly available and licensed under the GNU GPLv3.

Developing Diagram Services Although specifying the SaVED language and completing the diagram corpus for Archimedes’ *Floating Bodies*, *Book I* met the

⁸In fact the SaVED specification and schema, diagram corpus, *Diagram Compiler*, and *Diagram Navigator* are all publicly available through *The Episteme Project* at <http://episteme.sourceforge.net/>.

primary objective of the *Archimedes' Palimpsest Project*, we still wanted to satisfy their second objective to make our dataset useful in *actual practice*. We wanted to develop supporting tools so that people could actively explore the properties of the diagrams, and eventually cross-reference the constructed diagrams with the manuscript images and TEI texts. Rather than just building an end-user application, we developed a service layer with two services: the *Diagram Repository* for retrieving SaVED diagram encodings and the *Diagram Compiler* for rendering them.

Diagram Repository We built the *Diagram Repository* to resolve semantic and presentational references to SaVED-encoded diagrams. Citation is the operation by which mathematical argument becomes possible. The growth of knowledge depends upon the ability to resolve references to prior information and so we consider the *Diagram Repository* a fundamental service for mathematical digital libraries serving constructed diagrams. Retrieving referenced diagrams traditionally involves turning printed pages or scrolling through a PDF. Even then, entire diagrams are displayed, a far cry from the ability to reference arbitrary diagram components provided by SaVED-URNs.

For digital editions of constructed diagrams, we claim that page numbers are an unnecessary artifact of print. Digitization allows one to navigate diagrams (and texts) by semantic reference rather than by page number. The *Diagram Repository* is a *Canonical Text Services (CTS)* library loaded with SaVED-encoded diagrams. The CTS library [21] retrieves encoded diagrams for an entire construction. Individual components of a diagram are then referenced using SaVED-URNs. The *Diagram Repository* currently retrieves SaVED diagrams from a local filesystem. However, light refactoring of the code would enable a Google AppEngine or Groovy/Java implementation to also be used.

Diagram Compiler In addition to the *Diagram Repository*, we built the *Diagram Compiler* to serve SVG diagrams annotated with semantic information. The *Diagram Compiler* implements the SaVED language. The general process of compiling a diagram consists of providing two CTS-URNs pointing to the semantic and presentational markup for the same construction. Note that we use CTS-URNs since at this stage, we are choosing to interpret the SaVED code as text. The compiler then parses this code into an abstract syntax tree, constructs symbol tables relating the semantic and presentational objects, and emits an SVG diagram annotated with semantic information about each of the diagram components. The SaVED compiler, written in Python [23], is based upon the xcom compiler written in MATLAB [1].

Creating Diagram-Driven Applications With an extensible platform for retrieving and rendering SaVED code, we wanted to create a lightweight application so that end users could explore some of the properties of constructed diagrams that were not obvious in static images. We decided to start by highlighting the dynamic nature of constructed diagrams with our *Diagram Navigator*.

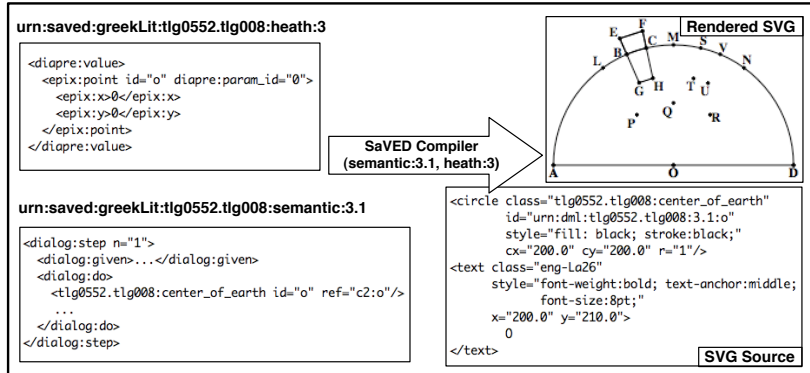


FIGURE 3. Given references to semantic and visual markup, the SaVED compiler generates a digital diagram in SVG format.

Diagram Navigator Our *Diagram Navigator* is a simple JavaScript application leveraging the generated SVG markup that allows one to interactively walk through a diagram construction. By simply clicking and holding a green bar to the right of the diagram, the construction progresses from beginning to end similar to the sequence seen in Figure 1.

Traditional diagram production requires one to read through the construction step-by-step, drawing entities upon the page until complete. For the construction process to be meaningful, one has to systematically remember the purpose of a presentational entity at a given step, increasing the time to create the diagram. Alternatively, one can quickly draw a diagram, but risk losing semantic context.

The current digital representation of diagrams greatly reduces the time required to produce diagrams and understand their meaning. Because navigation of associations between diagram entities and arbitrary information is now possible by linking from the SVG image, the logical context of an entity is now just a mouse click away.

4. Computing with the SaVED Architecture: Exploring the Diagrams of Archimedes

The SaVED architecture provides a system for encoding and interacting with diagrammatic content. Machine-actionable diagrammatic information opens the door to entirely new classes of research questions which could be explored through active experimentation upon these models. We designed, built, and delivered a framework for referencing diagrammatic content, computationally finding patterns therein, and quantitatively expressing those patterns. This section of the paper describes two observations discovered while producing the SaVED dataset for Archimedes' *On Floating Bodies, Book I* [16]. These observations motivate new computational

techniques for systematically studying how diagrams were *actually used* to present mathematical and scientific concepts.

Observations In *Floating Bodies, Book I*, Propositions 6 and 7, Archimedes primarily reasons about semantic entities with no intuitive geometric representation. Unlike a Euclidean circle or even a fluid surface that can be presented as a geometric circle or curve, Archimedes introduces concepts like weight and volume. In Heath’s edition, weight and volume are presented as rectangles. The heavier the weight of a solid, the larger its “weight rectangle” becomes and the size of the corresponding solid increases as well. Propositions 6 and 7 demonstrate that Archimedes used geometric representations of non-geometric concepts in the reasoning process. What conventions did Archimedes follow for presenting logical entities lacking an intuitive geometric form? The conventions observed in the Palimpsest edition of Archimedes emphasize the logical properties required by the proof.

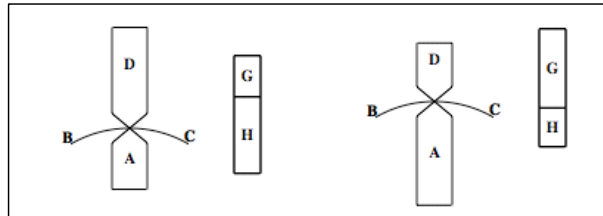


FIGURE 4. Adjusting the Weight of a Solid: Book I, Proposition 6, of *Floating Bodies*

As discussed earlier, Propositions 3 and 5 reveal that Archimedes actively uses previous constructions, changing them slightly in semantics and presentation. Does Archimedes refer to some constructions more than others? Does he ever refer to a subset of steps within a construction or does he always use them in their entirety? Perhaps the dependencies determined by these patterns of reference affected the order in which Archimedes presented propositions.

Quantifying the Observed Usage of Diagrams With an understanding of how Archimedes used diagrams within *On Floating Bodies*, we now propose computational techniques for gathering quantitative evidence to examine the questions raised above.

We first observed that Archimedes used geometric representations of non-geometric concepts in his reasoning. SaVED recognizes that diagrams may use geometry to model non-geometric concepts, keying concept to geometric representation via label. Questions such as whether Archimedes had diagramming conventions for presenting different concepts can be answered by first extracting the presentational types for semantic types like `tlg0552.tlg008:weight` and

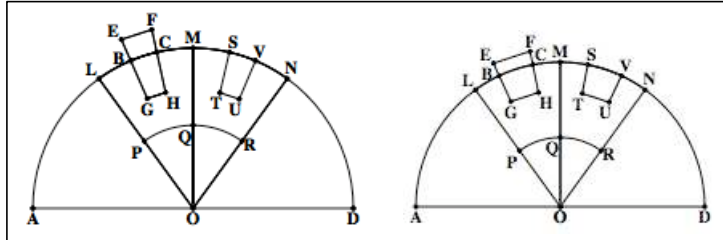


FIGURE 5. Propositions 3 (left) and 5 (right) from Book I of *Floating Bodies*

tlg0552.tlg008:volume and then doing a simple histogram where bins are presentational types. The results of such an analysis would quantify how Archimedes chose to represent concepts geometrically.

It was observed that Archimedes actively uses previous constructions, changing them slightly in meaning and presentation. Therefore, SaVED-URNs provide a mechanism for referring to previous diagrammatic content. Whether Archimedes refers to some constructions more than others and the level at which reference occurs can be quantitatively explored by extracting a reference graph from the encoding and looking at the number of incoming edges for each reference node. Proximity of reference could be explored by looking at the distance between the references within the structure of the text. Looking at different measures of this notion of proximity of reference could provide insight into the order in which Archimedes presented propositions and how Archimedes expected scientists to navigate his work.

5. Related Work

Barwise and Etchemendy, at the intersection of logic and computer science, explored heterogeneous reasoning—reasoning involving multiple modes of representation such as diagrams and sentences—in their *Hyperproof* software [2]. Others, including Shin [17], consider the logical status of reasoning with diagrams. Indeed, work has been done in formalizing the diagram proofs in Euclidean geometry, both in Philosophy with the development of Mumma’s *EU* [12, 13] and in Computer Science with Miller’s *FG* [11].

We reference and compute upon diagrams to drive tools that allow one to *actively explore* diagrams and *quantify* how they were *actually used* to visualize concepts. Our work builds upon established standards and mature technologies. SaVED, 5 years in the making, is based upon TEI P5 [3] which represents 15 years of research in encoding texts with XML. The CTS Protocol [21] has also been in development for 5 years and is based upon over 20 years of experience [6]

in computing with a variety of digitized texts and exposing those computations to users as a digital library.

6. Future Work

In future work, we plan to migrate the *Diagram Repository* from a filesystem-based CTS library to the HTTP-based CTS protocol, implementing this service on Google AppEngine. Secondly, building a *Diagram Indexer* service to parse SaVED diagrams and build up queryable tables (much like the symbol tables generated by the *Diagram Compiler*) will allow us to further experiment with semantically querying diagrams. Finally, the diagram corpora we develop could be used as training data for recognizing components of scanned manuscripts. The challenge here will be to build up a big enough data set relative to the work being scanned as semantics change across works. For example, a geometric circle may most likely mean a *fluid surface* in Archimedes but *Euclidean circle* in Euclid.

In general our approach should work on any constructed diagram that models relations between concepts geometrically. Two useful properties of digital diagrams emerged from this work. The first, associating semantic types with the geometric components of diagrams (whether constructed or not) will enable semantic search. Secondly, ordering the components of a diagram, even when there is not a natural construction order, may be beneficial. Both of these properties are useful in the context of circuit design, software engineering, and patent databases.

Consider applying these two properties to the circuit diagrams for computer hardware. Annotating sections of the diagram with semantic types would allow users to search a corpus of circuit diagrams at any level of abstraction desired. A student of architecture could query one or more CPU schematics for occurrences of flip-flops or search at a higher level for things like the register file. Matching diagram components could be highlighted and cross-referenced with hardware specifications. Additionally, if we imposed an order on the elements in the CPU diagram that corresponded to the datapath of a machine instruction, we could visualize instructions running on the CPU. Such a visualization would be of benefit to students of computer architecture.

Software designers regularly employ diagrams to communicate data models and protocols. Entity Relationship Diagrams (ERDs) are often used to show relations between classes. By annotating these diagrams with semantic information, one could search for all instances of a package, class, or variable and resolve the results to actual code sitting in an online code repository. Protocol design often involves producing a diagram showing how two or more entities communicate. By imposing the protocol order on the elements in the diagram, designers could step through the protocol. Furthermore, because SaVED-URNs let us reference arbitrary components of the diagram at any given step, multiple parties could annotate the diagram just as one might make comments on a draft of a text.

Finally, semantic annotation and search of diagrams might prove especially useful in the context of patent databases. Patents regularly employ diagrams to clarify concepts described within its text. Associated with the patent are a series of claims which the invention satisfies. A SaVED representation of patent diagrams would allow patent lawyers and inventors to search patent diagrams in terms of their meaning. By cross-referencing the descriptions or claims to relevant diagram components, inventors could quickly see how claimed behavior directly relates to the underlying mechanism illustrated within the diagram.

7. Conclusion

Reasoning about mathematical and scientific knowledge graphically is a technique used throughout history. Fundamental arguments foundational to mathematics and science have been expressed diagrammatically and transmitted alongside textual and tabular content from the manuscript, to the book, and now to the computer. The arrival of books greatly increased the ability to navigate and reference content. Innovations such as pages, tables of contents, and indices allowed readers to find chapters, sections, and other logical units of reference much more efficiently than traversing a rolled up manuscript. Digitization of textual content has led to entirely new modes of interacting with text such as search. Digitization of tabular content has led to the modern relational database [5], enabling the reuse, recombination, and easy querying of tabular data. Similarly digitization enables entirely new modes of interaction with diagrammatic content [8, 7].

With a mature representation of mathematical diagrams in hand, it now becomes possible to model a series of traditional operations for diagrams, namely navigation and production, as computations. While sketching computational solutions to Classical questions about the use of diagrams in Archimedes, a mechanism for the uniquely-digital operation of querying diagrams has emerged. More generally, we have seen how encoding diagram semantics and imposing an order upon diagram components may prove useful in applications ranging from hardware and software design to patent databases. Developing a larger corpus of encoded diagrams will provide scholars, scientists, and mathematicians with more data for quantifying how diagrams are *actually* used to present concepts in their respective domains.

References

- [1] B. McKeeman. xcom. Retrieved September 15, 2008 from <http://www.mathworks.com/matlabcentral/fileexchange/20149>.
- [2] J. Barwise, J. Etchemendy, G. Allwein, and A. Bush. *Hyperproof*. CSLI Publications, 1994.
- [3] Lou Burnard and Syd Bauman. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. 5th edition, 2007.

- [4] Christopher Clavius. *Theodosii Tripolitae Sphaericorum libri III*. ex typographica D. Basae, Romae, 1586.
- [5] EF Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13:377–387, 1970.
- [6] Gregory Crane. The Perseus Digital Library. Retrieved May 29, 2009 from <http://www.perseus.tufts.edu/hopper/>.
- [7] D. E. Joyce. Euclid’s Elements, 2008. Retrieved November 1, 2008 from <http://aleph0.clarku.edu/~djoyce/java/elements/elements.html>.
- [8] J. Glasgow, N.H. Narayanan, and B. Chandrasekaran. *Diagrammatic reasoning: Cognitive and computational perspectives*. MIT Press Cambridge, MA, USA, 1995.
- [9] Thomas L. Heath. *Great Books of the Western World: The Works of Archimedes Including the Method*. The University of Chicago, 1952.
- [10] A. Hwang. ePiX, 2008. Retrieved September 15, 2008 from <http://math.holycross.edu/~ahwang/current/ePiX.html>.
- [11] N. Miller. *Euclid and His Twentieth Century Rivals: Diagrams in the Logic of Euclidean Geometry*. CSLI Publications, 2007.
- [12] J. Mumma. *Intuition Formalized: Ancient and Modern Methods of Proof in Elementary Euclidean Geometry*. PhD thesis, PhD dissertation. Pittsburgh, Penn.: Carnegie Mellon University. Online at www.andrew.cmu.edu/user/jmumma, 2006.
- [13] J. Mumma. Ensuring Generality in Euclid’s Diagrammatic Arguments. In *Proceedings of the 5th international conference on Diagrammatic Representation and Inference*, pages 222–235. Springer, 2008.
- [14] R. Netz and W. Noel. *The Archimedes Codex*. Da Capo Press, 2007.
- [15] Open vs. proprietary formats. Retrieved July 5, 2009 from <http://www.openformats.org/>.
- [16] The Archimedes Palimpsest: Data. Retrieved November 1, 2008 from <http://www.archimedespalimpsest.net/>.
- [17] S.J. Shin. *The Logical Status of Diagrams*. Cambridge Univ Pr, 1994.
- [18] D. Neel Smith. CTS-URNs: Overview, December 2008. Retrieved May 29, 2009 from <http://chs75.harvard.edu/projects/diginc/techpub/cts-urn-overview>.
- [19] D. Neel Smith. Language/script encoding standards, December 2008. Retrieved July 5, 2009 from <http://chs75.harvard.edu/projects/diginc/techpub/language-script>.
- [20] D. Neel Smith. Citation in Classical Studies. *Digital Humanities Quarterly*, 3(1), 2009. Retrieved July 5, 2009 from <http://www.digitalhumanities.org/dhq/vol/3/1/000028.html>.
- [21] D. Neel Smith and G. Weaver. Canonical Text Services (CTS). Retrieved May 29, 2009 from <http://cts3.sourceforge.net/>.
- [22] D. Neel Smith and G. Weaver. Applying Domain Knowledge from Structured Citation Formats to Text and Data Mining: Examples Using the CITE Architecture. In *Text Mining Services*, page 129, 2009.
- [23] G. Weaver. The Episteme Project. Retrieved November 1, 2009 from <http://episteme.sourceforge.net/>.