PRÉCIS

Data-Sharing Tools for Authors and Publishers

LTS's goal is to curate a suite of software tools which can help both authors and publishers produce documents that can take advantage of modern technological capabilities such as text-mining software, open-access data sets, and multi-media science applications. Whereas there are many innovative technologies that have been implemented over the past decade, they have not been widely adopted in the publishing industry. LTS offers to unify and extend some of these solutions into an integrated toolset covering many aspects of the publishing pipeline.

Publishers, in recent years, have begun to encourage authors to share their research data, and to incorporate "Data Availability" or "Supplemental Materials" sections as an integral part of their journal and book publications. Initiatives such **FAIRSHARING** or the Bill and Melinda Gates Foundation Guidelines for Authors (which is part of **FAIRSHARING**) provide recommendations to help authors produce data sets that are Findable, Accessible, Interoperable, and Reusable (**FAIR**). Such initiatives represent a step toward realizing the goal of engineering a broad ecosystem of open-access scientific data — an ecosystem that interconnects and interoperates with both scientific publications and scientific-computing software applications. However, this goal is hard to attain because, in reality, scientific applications, publishing software, and data-hosting technology remain largely siloed from one another. Whereas LTS's contributions build off of existing projects, such as **FAIRSHARING**, they add features which ensure that publications, data sets, and scientific software would become more tightly integrated than they are today.

The Bill and Melinda Gates Foundation Guidelines for Authors

The Bill and Melinda Gates Foundation (**BMGF**) makes open data-sharing a precondition for funding of research projects in most instances. In so doing, the Gates Foundation provides guidelines for authors to prepare, describe, and deposit their data on popular scientific data-hosting platforms (such as DataVerse, Dryad, or Open Science), when they publish their work on the Gates Open Research portal. This is a good example of how publishers and organizations which support scientific research in effect encourage authors to conscientiously curate and share research data.

The **BMGF** guidelines are published as part of **FAIRSHARING** — specifically, they are part of a collection of standards maintained by the **FAIRSHARING** Initiative for documenting research methods and protocols. These standards also include several dozen domain-specific guidelines collectively referred to as **MIBBI** (Minimum Information for Biological and Biomedical Investigations). The **MIBBI** specifications serve as a checklist for authors preparing data sets to ensure that they properly, and in sufficient detail, document their research and/or laboratory protocols. Similar research-documentation formats (to **MIBBI**) include *Springer Protocols* (part of Springer Nature) and **BIOCODER** (a code library developed by Microsoft Research India).

As these examples demonstrate, researchers have numerous options for how they may document their research data. However, none of these documentation format options are closely integrated

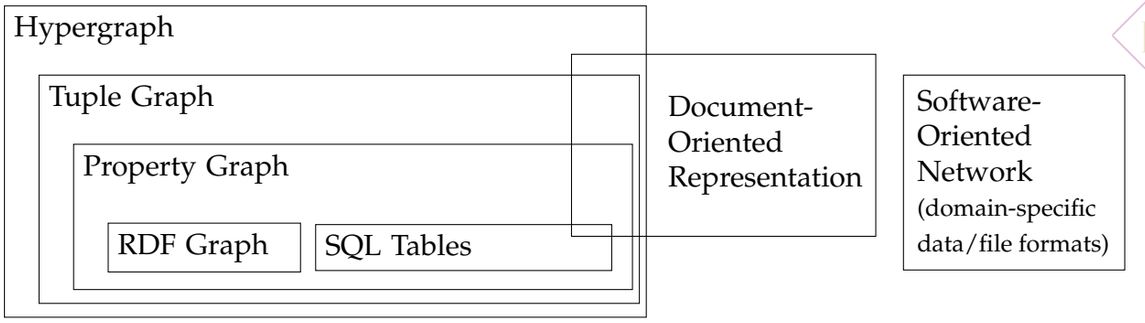


Figure 1: Relationships Between Different Kinds of Data Models

either with publishing software — in particular, with software used to compose books/articles for publication — nor with scientific software applications used to acquire, analyze, and visualize research data. Such extant lacunae point to the pressing need for a more integrated paradigm which unifies data curation, manuscript authoring, and application development into *one* single framework.

Unifying Many Data Formats into a Single Flexible Hypergraph Format for Data Sharing

FAIR data has two dimensions: first, how data sets are encoded and organized; and second, how data sets are connected to their corresponding scientific/academic publications. With respect to how scientific/research data is encoded and organized, several formats exist to encode both a data-set's metadata (information about the data set, such as its author, date, file types, size, folder organization, and software requirements) and its actual data. We propose to unify these various formats into a single flexible hypergraph-based format which is expressive enough to encompass all other commonly-used scientific data formats. This hypergraph model is based on the architecture of hypergraph database engines such as **HyperGraphDB**. The hypergraph model also extends property-graph databases such as **Neo4j**.

Data-sharing in the publishing context is often based on the Semantic Web. For example, "Research Objects" — a standard model for composing reusable data sets — relies on the canonical Semantic Web data format, **RDF** (Resource Description Framework), to encode research metadata. Semantic Web data is often criticized, however, for being "flat" — that is, **RDF** does not directly recognize any information structures which involve multiple levels or organization, or contexts: data collections (such as unordered sets or ordered lists), clusters of interrelated pieces of information, contexts where certain connections between information units have elevated significance, and so forth. To redress these limitations, multi-tier models such as Property Graphs and Hypergraphs have been proposed as alternative vehicles for Semantic Web data. While Property Graph and Hypergraph Database Engines are increasingly popular in business **IT**, these technologies have not been comparably adopted within the publishing industry.

The interrelationships between several different data-modeling paradigms — and hybrid models that seek to unify multiple paradigms — are sketched out in Figure **1** above. "Document-oriented" architecture in this context refers to formats such as **XML**, which allow arbitrarily nested content; these formats can model information with multiple levels of organization, but they are often difficult to work with in text-mining and data-mining contexts. Property graphs augment the flat-graph model by allowing sets of properties to be defined on edges and vertices. Property graphs, in turn, may be generalized to "tuple" models (where nodes could be collections of other nodes) and then subsequently to hypergraphs proper, which can be seen as tuple-graphs

augmented with extra details characterizing the information encompassed by individual nodes. As one proceeds from more restricted to more flexible data models, we achieve capabilities to represent larger classes of data structures, so that general-purpose data-sharing initiatives should in principle embrace broader frameworks (such as property or tuple graphs or hypergraphs proper) in lieu of narrower ones (such as **SQL** or **RDF**). However, it is a challenging problem to implement programming environments for the more complex graph models — in particular, encoding, querying, and validating complex graphs — which may be one reason why such technology has not been embraced by publishers, despite there being promising code libraries available that could serve as a foundation for technologies targeted to authors and publishers.

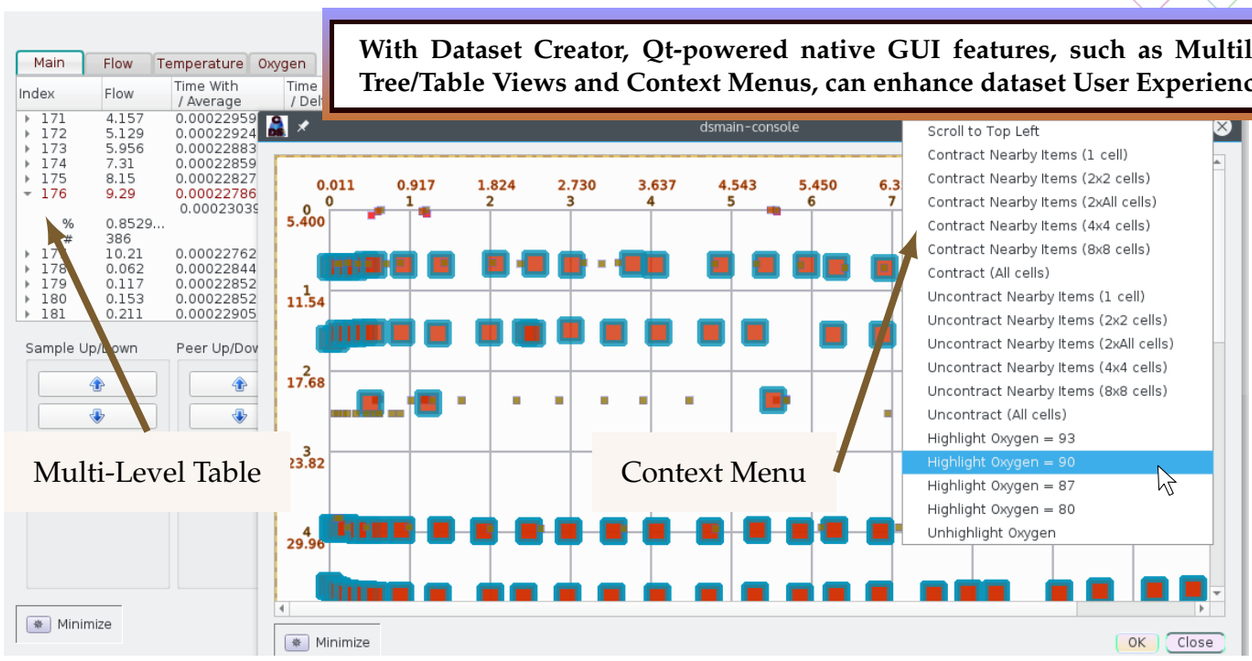
In particular, BitGraph (from the Johns Hopkins University Applied Physics Laboratory), a **C++** implementation of the "Gremlin Virtual Machine" — which provides a mechanism for querying property-graphs — is a good foundation for building a general-purpose engine for interoperating with open-access data sets. Several publishing-related technologies, such as **BioCoder** and **Pandore** (from GREYC lab, Caen, which is used for image processing) employ a "step-based" internal representation similar to Gremlin. This opens the possibility for an enhanced step-based virtual machine which could be used to model both research data and research methodology as well as to extract information from published data sets. The idea of a "step-based virtual machine" is a relatively new innovation in computer science, driven by the rise of property-graph models as an alternative to the traditional Semantic Web. However, its potential has not yet been fully realized in the publishing environment. To address this, LTS proposes to significantly expand BitGraph and codify a multi-purpose Step-Based Virtual Machine architecture which would *unify* several database and data-set related technologies into a single programming framework.

Specific Tools for Authors and Publishers

LTS's proposed "Scientific Data Repository Framework" (**SdrF**) includes tools for authors and code libraries for publishers. For authors, one component within **SdrF** is "Dataset Creator (dsC)," which is a development environment assisting authors to construct published data sets based on the **Qt C++** application-development framework. The dsC resources include code libraries, project files, build scripts, and LaTeX packages targeted toward **Qt** Creator (a **C++** Integrated Development Environment). Authors and data-curators would then use **Qt** Creator as a platform for building components such as the **TAGML** parsers discussed below. In addition, dsC provides templates so that authors may supplement these components with their own data and code, producing shareable/reusable data sets with their own coding environment. If desired, authors can also supply their own **Qt**-based **GUI** components, giving them great flexibility to choose how their data should be visualized and manipulated. In short, with dsC authors can share data via customized standalone desktop-style applications, which are distributed in source-code fashion alongside their raw data.

Figure 2 shows an example of a data set published using a custom-designed **GUI** front-end in lieu of a generic application, such as a spreadsheet. By coding the **GUI** logic directly, the data-curator can add features to how the data is presented — for instance, as seen in the figure, employing a multi-level table view instead of a flat spreadsheet, which enables the **GUI** to visually distinguish input parameters acquired directly from the experimental apparatus from intermediate values obtained via calculations. Moreover, the **GUI** front-end can offer a more sophisticated interactive User Experience than a generic (e.g., spreadsheet) application. Context menus and secondary

Figure 2: Customizing Data-Set GUIs



windows/dialog boxes, for instance, may be used to modify the data-visualization settings or to present additional information about different features of the data set in relation to the overarching research project being presented.

For publishers, **SDRF** includes a "Scientific Data Repository Model" (**SDRM**) which serves as a template that may be instantiated with specific information about individual publishers' portals and technologies (a detailed summary of **SDRM** is included in our overview of **SDRF** at https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/sdrf/SDRF.pdf). Each concrete **SDRM** would provide code and information governing how software tools may interact with publishers' technologies, including: **API**s; protocols for obtaining documents and other assets; information about manuscript encoding; publisher-specific LaTeX packages or **XML DTD**s; representations of publishing workflows; checklists/guidelines for authors, editors, data-curators, and other individuals who play a role in the publishing process; and code libraries which automate certain tasks associated with these roles. For example, the **TAGML** parsers discussed below could be individually extended for each publisher's environment, providing publisher-specific commands whose implementations would address a specific publishing task (e.g., annotating bibliographic references to articles hosted on the publisher's portal).

Augmenting Publishing Technology via TAGML

In addition to describing/encoding data sets, it is also important to compose publications via technologies which can document the connections between publications and data sets. Popular languages for encoding academic documents, such as **JATS** (Journal Article Tag Suite) are inadequate for notating how documents and data sets are interrelated. For example, **JATS** has a "data title" element which documents an article reference that is also a data source (a file or a data set which is deposited somewhere on the web), but **JATS** has no way to insert annotations documenting when a sentence, paragraph, technical term, or figure/diagram within a publication is connected to a data set, such as a visualization of statistical distribution, the name of a table/column/data-type, the index/identifier for one data record, etc.

To redress the limitations of **JATS** and similar document markup formats, LTS has implemented a

C++ parser for **TAGML** (the Text-As-Graph Markup Language), which was originally developed (in Java) by the KNAW (Royal Netherlands Academy of Arts and Sciences) Humanities Cluster (a consortium of several Dutch universities). **TAGML** is an example of "going beyond **XML**" with support for concurrent/overlapping markup and an underlying hypergraph text representation which is conducive to text mining. In short, LTS extends **TAGML** with features allowing **TAGML** documents to be granularly linked to data sets. In addition, LTS adds features allowing **TAGML** to produce LaTeX code optimized for interactive **PDF** viewers. By so doing, LTS enhances **TAGML** with features allowing **TAGML** to be employed as a general-purpose graph-serialization language. What this means is that **TAGML** can then be utilized for encoding data sets, so that the same input format could be used both for data sets and for the publications associated with them.

Extending Scientific Software to Interoperate with Publications

As indicated on the right-hand edge of Figure **1**, research data sharing sometimes occurs through special-purpose file formats which require special domain-specific software. Domain-specific file formats are associated with individual scientific disciplines: **PDB** (Protein Data Bank) with Organic Chemistry; **FCS** (Flow Cytometry Standard) with immunology and serology; **OME-TIFF** (the Open Microscopy Environment version of the Tagged Image File Format) with microscopy; **DICOM** with radiology; and so forth. Research initiatives grounded in those disciplines — the "Flow Repository" (for **FCS** data), the Radiological Society of North America's Imaging Data Repositories, the International Nucleotide Sequence Database Collaboration, etc. — share data encoded with these special formats, which are optimized for their specific discipline, with the understanding that most researchers who would have occasion to re-use that data would have access to the proper software.

While there is nothing wrong overall with data sharing through special-purpose scientific computing applications, the problem from the author's and the publisher's perspectives is that these applications do not usually have features enabling them to interoperate with software that is utilized to compose or read publications. For example, one cannot at present use a **PDF** viewer to read an article about image analysis and then (simply by clicking on an example image) load image-processing software, so as to reproduce the analysis described in the text. In order to make such interoperability possible, vendors of scientific applications need to implement plugins or extensions through which these applications could receive data from **PDF** viewers (and other e-readers for digital publications). Only via such capabilities (for example, to receive data from **PDF** viewers) can scientific applications support what the Research Object model calls "Executable Papers."

LTS proposes to apply the techniques of Step-Based Virtual Machines mentioned above in order to provide a programming infrastructure to facilitate the design of plugins which support such document-to-application interoperability. The concept is to decompose the requirements for interoperation between document-viewers and science applications into several simple steps, some to be executed by the document-viewers and some by the science applications themselves. This step-by-step breakdown would then offer a blueprint for implementing a common data-sharing protocol which ensures that scientific publications and scientific software may be properly *interconnected*. Moreover, the encoding of this protocol could be folded into a general-purpose step-based model also used for query processing and documenting research methods.