EXECUTIVE SUMMARY

New LTS Tools to Accelerate Covid-19 Research

LTS (Linguistic Technology Systems) is founded by Amy Neustein, PhD, Series Editor of **Speech Technology and Text Mining in Medicine and Health Care** (de Gruyter); Editor of **Advances in Ubiquitous Computing: Cyber-Physical Systems, Smart Cities, and Ecological Monitoring** (Elsevier, 2020); and co-author (with Nathaniel Christen) of **Cross-Disciplinary Data Integration Models for the Emerging Covid-19 Data Ecosystem** (Elsevier, forthcoming).

Introduction

The Cross-Disciplinary Repository for Covid-19 Research (**CR2**) is a collection of open-access research data sets related to SARS-CoV-2 and Covid-19, which is being developed as a supplement to the forthcoming Elsevier volume examining Covid-19 research from the perspective of text and data mining. This repository is an opportunity for LTS to develop and showcase new database engineering and data-set construction technologies. Both the new technologies and the **CR2** repository will be outlined in the following sections.

Some of the **CR2** code/data will be hosted on GitHub at Mosaic-DigammaDB/CRCR (for data aggregation) and Mosaic-DigammaDB/LingTechSys (for code previews). The second repository includes "preview" code sampling database engine features (the logic for constructing a database in shared memory, encoding data types for persistence, and so forth). Companies or research groups interested in a more substantial code preview are invited to contact LTS, where we can discuss how the data management tools developed for **CR2** can be customized for your own projects.

Overview of the New Technology

The main challenge when curating a data repository such as **CR2** is reconciling heterogeneous data formats. In response to these problems, LTS has focused on hypergraph-based data models which can unify many different information structures into a common structure. In particular, **CR2** will introduce a Hypergraph Exchange Format (**HGXF**) which can take the place of disparate tabular or graph file formats (comma-separated values, numeric python, spreadsheets, graph-network serializations, etc.). In addition, **CR2** will introduce a new protocol for engineering hypergraph databases, dubbed **THQL** (Transparent Hypergraph Query Language). Databases conformant to the **THQL** model will be able to export data in hypergraph-based formats, thereby generating a data set which can be used as a published, citable Research Object. **CR2** will also introduce a new database engine (called DigammaDB) which serves as a "Reference Implementation" for **THQL**. In **CR2**, DigammaDB (ϚDB) is used to curate data sets before their final form is exported into the main data repository. However, ϚDB is only a prototype and reference demonstration for **THQL**; Customized commercial versions of a **THQL** engine can be tailored to each project's data storage and application-development requirements.

THQL is designed with a priority on application development. In particular, any instantiation of **THQL** should provide data persistence capabilities through (as much as possible) self-contained code libraries that can be included in source-code form within an overall application. **THQL** is designed to integrate seamlessly with native, desktop-style standalone applications — for example, it is easy to implement **GUI** components which are initialized with data obtained from a **THQL** database. In short, **THQL** represents an unprecedented combination of native desktop-style software development and hypergraph database engineering.

Complementing **THQL**'s application-development focus, **CR2** will also introduce "Dataset Creator," (dsC), a new tool for curating research data sets. The pre-eminent feature of Dataset Creator is its use of native software components (called "Dataset Applications") allowing researchers to view, manipulate, and reuse research data. In sum, the typical Research Object built with dsC will include self-contained source code implementing a customized desktop application providing access to the accompanying data set. In addition to **GUI** code, each Dataset Application will provide "data-access" code libraries for parsing the raw data-set files, so as to obtain the information visualized within the Dataset Application's **GUI** classes. These

data-access libraries are also useful for providing machine-readable access to such raw data. Subsequent researchers can then re-use the data-access code so as to transform, filter, or analyze the published data in the context of replication studies and/or novel research projects.

Development of **THQL** and dsC is keyed to **CR2**; the **CR2** repository will provide a practical test-bed for validating this new technology. Accordingly, the following sections will describe **CR2** in greater detail, followed afterword by sections providing more information about **THQL** and dsC.

The Cross-Disciplinary Repository for Covid-19 Research

The sudden emergence of Covid-19 as a global crisis has cast a spotlight on computational and technological challenges which, in the absence of a catastrophic pandemic, would rarely rise to public attention. In particular, an effective response to the dangers of SARS-CoV-2 requires coordinated policy making integrating diverse modes of scientific inquiry. Genomic, biomolecular, epidemiological, socio-demographic, clinical, and radiological information are all pertinent to Covid-19. In this environment, it is important that the empirical foundations for expert recommendations — which in turn drive public policies of enormous social and economic consequence — be transparently documented and critically examined. The proper synergy between government and science depends on data centralization: given the gaps in our current Covid-19 knowledge, it is understandable that different jurisdictions will craft responses to the pandemic in different ways. There is no central authority with sufficient epistemic force to legitimize homogeneous mandates across the entire country. However, such policy differences should be a consequence of alternative interpretations of scientific knowledge or the diverse needs of local communities — rather than being a haphazard consequence of governments working with divergent, competing, and poorly integrated data.

The current administration, along with numerous corporate and academic entities, has clearly recognized the need for a more centralized paradigm for sharing Covid-19 data. For example, the White House spearheaded a scientific initiative to develop **CORD-19**, an open-access corpus of over 46,000 peer-reviewed publications related to Covid-19, which were transformed into a common machine-readable representation so as to promote text and data mining. Similarly, large institutions such as Google, Johns Hopkins, and Springer Nature have all implemented some form of coronavirus data-sharing platform targeted to both scientists and policy makers. However, these two aspects of the corporate/academic contributions to Covid-19 data sharing (exemplified by the **CORD-19** White House initiative and by institution-generated portals, respectively) have been incomplete, for opposite but complementary reasons. Specifically, **CORD-19** is highly structured and tightly integrated, but it focuses primarily on text mining and scientific documents, not *research* data. While it is possible to find data sets about Covid-19 through **CORD-19**, the techniques to do so are both cumbersome and non-scalable. On the other hand, projects such as the Johns Hopkins coronavirus "dashboard" provide accessible data sets, yet these projects are isolated and do not offer the level of structure and integration evinced by **CORD-19**. In short, an optimal Covid-19 research platform would merge the structural text-mining rigor of **CORD-19** with the data-centric focus of isolated projects that share Covid-19 data with the scientific community, policy makers, and the general public.

The benefit of **CR2** is that it can accelerate Covid-19 research by (1) pooling a diverse collection of data sets into a single resource which scientists can utilize; (2) serving as the prototype for larger research portals that can aggregate new Covid-19 data that will emerge from hospitals, labs, and academic institutions in the future; (3) formalizing a framework for aggregating patient narratives to accurately capture first-hand subjective symptomatology of the patient suffering from Covid-19; and (4) accelerating the implementation of novel data-integration and software-development technologies which can contribute to scientific progress vis-à-vis Covid-19 in particular, and biomedical/scientific computing methodology in general. The software used to curate **CR2** data has diverse applications for software and database engineering, and provides solutions to technical problems with a broad reach in the private sector. Further documentation of the **CR2** technology and products may be found on the development repository for the aggregation of **CR2** data (Mosaic-DigammaDB/CRCR).

The design of **CR2** derives from the principles outlined in the previous paragraph. In particular, an ideal data-sharing ecosystem should merge data from multiple sources, but should do so in a fashion which yields a machine-readable totality, analogous to **CORD-19**'s structuration with respect to text mining. The merit of **CR2** therefore lies not only in the data which it will encompass but also in novel technology

that it will concretize for constructing data repositories adhering to these principles. Accordingly, **CR2** can provide value at different scales of realization. Relatively small data sets serve several scientific and computational purposes: (1) they can provide researchers with a mental picture of how data in different disciplines, projects, and experiments is structured; (2) they can serve as a prototype and testing kernel for technologies implemented to manipulate data in relevant formats and encodings; and (3) they can lay the foundation for data-integration strategies. For example, when designing a representation format and/or implementing code to merge different data formats into a single structure (or meta-structure), it is useful to work with small, representative examples of the data structures involved, so as not to complicate the integration logic with computational details solely oriented to scaling up the data-management logistics. As a result, **CR2** can provide a testbed for implementing data-integration technologies which can scale up as needed. To fulfill this mission, **CR2** can aggregate relatively small data sets which have previously been published on academic and research portals, such as Springer Nature, Dryad, and DataVerse. At the same time, a more substantial (and not necessarily fully open-access) Covid-19 data-set collection would also be beneficial to the scientific and policy-making community. Ideally, then, **CR2** will be paired with a larger technology which shares a similar implementational strategy but with different accession paradigms, allowing for an open-ended collection of Covid-19 data which users may selectively access (instead of a single package that users may acquire as an integrated resource). The common denominator in both cases (whether the focus is on relatively smaller or larger data sets) is the importance of deploying novel and contemporary data-integration techniques to centralize Covid-19 research as much as possible. Accordingly, this summary will briefly explain how **CR2** can accelerate Covid-19 data integration on both a practical and technological level.

Methodology for Covid-19 Data Integration

As indicated above, pertinent Covid-19 data is drawn from multiple scientific disciplines. On a technological level, Covid-19 data is documented via a wide array of file types and data formats. This diversity presents technological challenges: if a Covid-19 information space encompasses files representing 25 different incompatible formats, users would need 25 different technologies to fully benefit from this data. In many cases, however, data incompatibilities are merely superficial — an important subset of Covid-19 data, for example, has a common tabular meta-model, even if the data is realized in discordant technologies (spreadsheets, relational databases, comma-separated-value or Numeric Python files, and so forth). Applying **CR2**'s technology, one level of data integration can thus be achieved simply by encoding tabular structure into a common representation: any field in a table can be accessed via a record number and a column name and/or index. In some cases, more rigorous integration is also possible — for example, by identifying situations where columns in one table correspond semantically or conceptually to those in another table. In either case, it is reasonable to assume that a single abstract data format lies behind surface data-expression in patterns such as spreadsheets and comma-separated values (**CSV**), so that all files in an archive encoding spreadsheet-like data can be migrated to a common model.

Other forms of clinical and epidemiological inputs are often more amenable to graph-like representations. For instance, trajectories of viral transmission through person-to-person contact is obviously an instance of social network analysis. Similarly, models of clinical treatments and outcomes can take graph-like form insofar as there are causal or institutional relations between discrete medical events: a certain clinical observation *causes* a care team to request a laboratory analysis, which *yields* results that *factor* into the team's decision to *administer* some treatment (e.g., a drug *from* a particular provider *with* a specific chemical structure), which observationally *results* in the patient improving and eventually *being* discharged. In short, patient-care information often takes the form — at least conceptually — of a network comprised of different "events," each event involving some observation, action, intervention, or decision made by care providers, and where the important data lies in how the events are interconnected: both their logical relationships (e.g., cause/effect) and their temporal dynamics (how long before a drug leads to a patient's improvement; how much time elapses between admission to a hospital and discharge). These graph-like representations are a natural formalization of "patient-centered" data models.

Using **CR2** associated software, a higher level of data integration can then be achieved by merging tabular and graph-like models into a single *hypergraph* format. A significant subset of Covid-19 data (or, more generally, any clinical/biomedical information) conforms to either tabular or graph structures; thus it is

feasible to unify all of this information into a common framework. A graph-plus-table architecture is generally considered some form of Hypergraph model, and indeed **CR2** adopts a hypergraph paradigm to merge many different sorts of information into a common structure. In particular, **CR2** introduces a new "Hypergraph Exchange Format" (**HGXF**) which can provide a text encoding of many files that, when originally published, embodied a diverse array of file-types requiring a corresponding array of different technologies. **CR2** will include specialized computer code that would enable machine-readability of the **HGXF** files, and use them to create hypergraph-database instances. In short, **CR2** will promote Covid-19 data integration by translating a wide range of files into a common **HGXF** format, something that has not yet been done before.[1]

Hypergraph Data Models and Multi-Application Networks

As has been outlined thus far, via the **CR2** technology most Covid-19 data can be wholly or partially integrated into a single hypergraph framework, which accordingly simplifies the process of designing software applications and algorithms to analyze and manipulate this data. Specifically, software components can employ a single code library to obtain, read, consume, and store data, rather than needing to re-implement this logic for a large number of different file formats and/or database models.

Quality software (especially in the clinical and biomedical context) demands a balance between applications which are either too broad or too narrow in scope. On the one hand, doctors often complain that homogeneous Electronic Health Record systems (where every digital record or observation is managed by a single all-encompassing application) are unwieldy and hard to work with. This is understandable, because the clinical tasks of health care workers with different specializations can be very different. On the other hand, doctors also complain about software and information systems which are so balkanized that they must repeatedly switch between different, non-interoperable applications. In short, clinical, diagnostic, and research software should be neither too homogeneous nor too isolated; finding the proper balance between these extremes is, no doubt, a major challenge to the usability of electronic health systems going forward.

Against this background **CR2** demonstrates novel solutions to this problem: it focuses on the dimensions of data acquisition and management that are specific to individual scientific or medical specializations, while also identifying requirements that are consistent across domains. Scientific software generally needs to hone in on the data visualization and analytic requirements of particular disciplines; for example, biochemists use different programs than astrophysicists. However, much of the code underlying scientific applications has nothing to do with these high-level models or theories, but is simply a fulfillment of basic data-management functionality — data storage, accession, provenance, searching, user validation, and so forth. In effect, the computational requirements of scientific and biomedical software can be partitioned into two classes: (1) domain-specific logic which reflects the quantitative or theoretical models of narrow scientific fields; and (2) data-management logistics which can be realized within a central access hub, rather than being re-implemented by each application in isolation.

In short, the architecture enabled by **CR2** conceives of a central hub which is responsible for storing data and for serving as a common access point — providing the "gateway" where authorized users can gain access to heterogeneous information spaces utilized by an array of domain-specific software applications. Since peer applications would not be directly responsible for data persistence or user identity management, they can focus on their specific data analysis and visualization capabilities. The central hub, serving multiple peer applications, is then a heterogeneous data space managing information from multiple applications while also tracking information about the applications themselves: helping users to identify and launch the software which is most directly relevant to their clinical or research needs at the moment. Meanwhile, because peer applications are jointly connected to a central hub, it is possible to implement scientific workflows where one application may send and receive data from its peers, allowing applications to complement each others' capabilities.

---

[1] Not every format relevant to Covid-19 can be realistically translated to **HGXF**. In particular, scienctific fields requiring substantial quantitative analysis — e.g., biomechanics or genomics — express data via encodings optimized for relevant mathematical operations. In this scenario, **CR2** will not attempt to migrate *all* of a data file to **HGXF**. However, even for these files **CR2** will generally provide a supplemental **HGXF** encoding supplying data *about* the original file, with information about the file type, preferred software components for viewing/manipulating its data, and so forth. In this manner the contents of non-**HGXF** files can be indirectly included into the **CR2** hypergraph-based ecosystem.

This multi-application networking architecture has precedents in some of the current database and engineering technologies. For example, many hospitals and medical institutions employ some version of a "Data Lake," pooling disparate data sources into a heterogeneous aggregate which is then accessed by multiple client applications. Similarly, Machine Learning and Artificial Intelligence often adopts "software agents" or analytic modules in contexts such as Online Analytic Processing, which again represent semi-autonomous software components sharing an originary data hub. Web applications, too, often act as domain-specific subsidiaries deferring operational requirements, such as user authentication or transaction processing, to a central web service. The limitation of multi-application networks in these existing contexts are that the software agents involved are generally "lightweight," with relatively primitive user-interface design. By contrast, the hypergraph technology introduced with **CR2** will support multi-application networking in the context of more substantial desktop-style scientific applications. In sum, the novel hypergraph technology developed by LTS offers a hybrid of the development methodologies employed for desktop scientific software and those applicable to multi-agent heterogeneous data stores, like a Semantic Data Lake. To accomplish these goals, **CR2** will utilize a new hypergraph database engine, coded in the **C++** programming language, which has a unique focus on supporting native **GUI** applications from the ground up, including persisting application state and storing application documentation within the database itself.

A New Paradigm for Data Sharing and Data Transparency

One exceptional feature of Covid-19 research is the extent of public attention focused on scientific discoveries about the disease. Academic and commercial research teams find themselves in an unprecedented situation where there is unusual pressure to accelerate the Research and Development process, and a concomitant demand for a novel level of transparency and openness. For example, vaccine development protocols are being fast-forwarded to take months instead of years, and information about the development process (such as trial results and scheduling) will likely be shared with the public much more than is standard practice. This new reality, in turn, calls for a commensurate evolution in the technology for public data-sharing.

In conventional biomedical R&D, much of the research data is proprietary, and revealed only in restricted contexts to select parties (such as the Food and Drug Administration). Data which *is* then publicly shared tends to be tied to published research papers in peer-reviewed literature, primarily read by a relatively small, specialist audience. All of this is changing with SARS-CoV-2: companies pursuing Covid-19 R&D (in the context of vaccine trials, for example) are facing pressure to publicly share their results as soon, and as transparently, as possible; and policy makers, scientists, and journalists are no less looking for quick access to research data directly, rather than circuitously through academic publications.

CR2 will introduce the new Dataset Creator technology targeted toward this new environment of direct, transparent data-access (dsC will be discussed in greater detail below). Data sets created via this technology therefore implement the "Research Object Protocol," which mandates that research data be bundled with code allowing scientists to analyze and manipulate the information in the corresponding data set. The Research Object framework was designed by a consortium of academic and governmental entities, such as the National Institutes of Health, to promote a paradigm for data publishing which prioritizes multi-faceted research tools over "raw" data that can be difficult to reuse in the absence of supporting code. In particular, Research Objects should be (as much as possible) *self-contained*, which means that scientists do not need external software dependencies to access and study the data — any special code which is a prerequisite to using this data should be included, alongside the raw data, as part of the Research Object itself.

Dataset Creator enables standalone, self-contained, and full-featured native/desktop applications to be uniquely implemented for each data set, distributed in source-code fashion along with raw research data. Adopting such a data-curation method makes data sets easier to use across a wide range of scientific disciplines, because the data sets are freed from having to rely on domain-specific software (software which may be commonly used in one scientific field but is unfamiliar outside that field). In addition, Research Objects composed with dsC can be integrated into Multi-Application Networks (which are described in the previous section) because the dataset applications are autonomous native **GUI** applications that can easily interoperate via **Qt** messaging protocols.

Of course, most of the **CR2** data sets are previously-published work composed via older technology. Many of these resources, created with a wide range of software products, predate (or fail to apply) contemporary

specifications such as the Research Object Protocol; not every CR2 data set will have the full set of features described in this section. However, CR2 will try to maximize the value of each data set by translating them into a QT-based format — in particular, CR2 will provide QT code for reading HGXF files, as well as a QT-based hypergraph representation library. Following the data integration methods outlined earlier, much of the CR2 data can be merged into a QT-based framework, which can facilitate the implementation of new, more sophisticated Dataset Applications as the information in CR2 gets reused for subsequent research. CR2 will also include QT-based software, such as a customized PDF viewer, which will help researchers utilize the corpus in its entirety. For example, CR2's PDF viewer will include special code to connect PDF files with data sets via "micro-citations," as discussed in the next section.

Supporting Data Micro-Citations to Improve Machine Readability

The CR2 database engine supports annotating individual components of a database — a technology sometimes referred to as "micro-citation." Data micro-citations are references to integral parts of a data set, such as an individual table, or a single row/record or column in a table. Micro-citations allow these integral parts within the data set to be cited by and linked to publications, for purposes of machine readility and attribution. As an example, preliminary vaccine trials often target a patient cohort selected for demographic or medical criteria matching the population who would most benefit from the vaccine. These criteria for selecting the cohort for the vaccine study are usually described in the texts of the articles. However, these criteria are also identified within the data set by socio-demographic data which is part of the information generated by the trial. By making these connections between criteria discussed in the article and those represented in the corresponding data set explicit, text and data mining can be *merged* as analytic tools targeting a data repository, so that machine reading is able to mine not just article text but the corresponding data.

One reason why micro-citations are important is that they clarify the scientific meaning attributed to data set elements by connecting these elements to scientific concepts and "controled vocabularies" (such as a list of drug names, diseases, proteins, etc.). For instance, micro-citations allow table columns to be mapped to statistical parameters, enabling their empirical properties (such as min/max values and distribution) to be queried by text and data mining software. Likewise, CR2 enables dimensional and measurement annotations to describe the empirical and experimental significance of the measured or calculated quantities which are stored in a database. Such quantity dimensions model the conceptual roles which particular parameters perform: e.g., the axiation "mJ/cm^2"(millijoule per square centimeter) indicates the intensity of ultraviolet light — any table (or other data aggregate) having a column or field with this dimension is intrinsically associated with observations or experiments pertaining to UV light. Consequently, to locate data sets relevant for research about the clinical uses of antiviral UV radiation, one method is to search for data fields dimensionalized in terms of joule or millijoule per square centimeter. As this example illustrates, data micro-citation — via annotations on data fields, statistical parameters, and table columns — is an important data-mining tool. In short, constructing micro-citations within a database serves two distinct benefits: (1) to aid data mining; and (2) to enable granular links (joining specific parts of articles to corresponding parts of the data set in the data set repository — analogous to hyperlinks between web pages) to be established between publications and data sets, making it *easier* for researchers to find the specific information most relevant to their own research.

Adding Patient Narratives to Covid-19 Data

In addition to aggregating published data sets, CR2 may be used as a repository for collecting new Covid-19 information. With that in mind, we are prioritizing the design of a standard for storing and accessing natural-language text representing patients' subjective symptom descriptions, which is quite useful for diagnostic/prognostic assessments of patients infected by Covid-19.

Just as CR2 envisions a curation of published data sets for data mining to improve machine-readability of Covid-19 research, LTS also sees the benefit of a repository of patient narratives prepared for text mining, to improve machine readability of the open-ended symptom descriptions offered by patients. While CR2 does not need to specify how these narratives should be collected, it will implement a common representational format so that patient narratives can be pooled, similar to to how CORD-19 research texts are merged and encoded with a system that permits annotation.

In modeling patient narratives, this technology will be oriented toward the scientific-computing ecosystem outlined in the previous section. In particular, we assume that GUI-based desktop applications will be the primary instruments for data collection and analysis; this means that the encoding of patient narratives may, at times, need to be paired with GUI or multi-media content. For example, the software for patients to submit medical history information could also allow them to pair (text-form) narratives with graphics indicating the location of their pain or discomfort. Furthermore, the software could allow narratives to be accompanied by an audio file where patients could cough/speak into a microphone. In light of this range of possible inputs, a patient-narrative encoding must, therefore, be flexible enough to include diverse multi-media content.

As described earlier, an information space adapted for multiple peer applications should encompass capabilities for saving application state (the current visual appearance of the program), which includes features for modeling instances of GUI classes. This technology provides the necessary infrastructure for managing patient narratives. For example, consider a multi-media intake form where patients may describe symptoms by placing icons (representing pain or discomfort) against anatomic silhouettes (head/body, back/front, extremities, and so forth). As patients use such a multi-media form, GUI application state corresponds to the patient's subjective symptomology; in this way the graphics-based represention of symptoms could then be incorporated into the overall patient narrative. This is an example of how application-persistence logic can be marshaled to the related project of curating patient narratives.

Native Application Development with THQL

As described earlier, THQL is a database engineering protocol which prioritizes data persistence components that can be included in source code fashion within application-development projects. THQL is "transparent" in that all layers of data persistence and query processing logic are provided via self-contained source-code libraries. The complete database functionality can then be statically examined via the source-code files, and dynamically examined by running the client application through a debugger. Moreover, because all THQL source code is bundled with application code, THQL can be configured to integrate seamlessly into the client application logic. For example, THQL can be extended to natively recognize client-specific datatypes as data fields, or to execute client algorithms as query parameters.

As a query *language*, THQL can be instantiated either by a special language with its own syntax and semantics (analogous the SQL or SPARQL), or as an interface and pattern specified for a conventional language, such as C++. In the latter guise, THQL provides a common protocol for essential database tasks, such as constructing, updating, querying, and backing up database instances. Each procedure comprising the THQL protocol is assigned a specific role, so the protocol can be abstractly modeled as a set of data-management roles mapped to corresponding procedure implementations. On this basis, custom query languages can be achieved by exposing each role-procedure to a scripting interface. For example, THQL's Reference Implementation, DigammaDB, exposes each protocol-specific procedure via a set of globally accessible pointers to C++ functions. Parsing a query languags is then a straightforward process of mapping query expressions to the requisite procedure, whose corresponding handle can then be obtained via C++ interop.

As indicated by its name, THQL is centered around the operations to define and store hypergraph-form data: information which has several levels or scales of structuration. This means that the THQL protocol includes procedures for registering individual data fields (representing, in general, primitive types such as integers and decimals) in a database; aggregating fields into "hypernodes," or groups of interrelated information; connecting pairs of hypernodes by identifying a specific connection which they have; adding contextual details or annotations (via so-called *frames* and *channels*) which refine assertions of hypernode connections; and constructing "proxies" to database elements (such as hypernodes, frames, and subgraphs) which can be referenced via unique identifiers, as individual data fields. Since proxies can then be aggregated into hypernodes in turn, THQL graphs can have, if desired, arbitratily deep nested structures.

In addition to the operations just outlined, THQL recognizes additional stuctures corresponding to conceptual details described above — for example, fields within hypernodes can be linked with dimensional attributes (e.g. scales and units of measurement) and identified as microcitation targets. THQL likewise supports controled vocabularies, called "dominions" (which stands for Domain-Specific Mini-Ontologies), applying

to hypernode-types and/or connection labels. A graph can then be configured to only accept hypernodes which conform to one of the dominion-defined types; and/or to only allow connections to be asserted between hypernodes when these connections can be labeled from a dominion-specific list of connectors. Less restrictively, graphs can be defined in a more free-form style but use dominions to filter or query nodes and edges.

Another feature of THQL, relevant to application integration, is the notion of cofiguring each database to support different "modes" of data persistence. It is possible to use THQL for completely in-memory data management, with no direct data persistence at all. This would be an appropriate solution when data can be read all at once from a static source, such as a data set. In this guise, THQL would be used to build a structural model of the data set, which can then be queried by application code. Conversely, it is possible to employ THQL as a continuously-updated data store, where changes to an underlying THQL graph are persisted to disk as soon as they are registered. Between these extremes, THQL graphs can hold dynamically changing representations of a persistent database, which are only incorporated into the underlying database when instructed by the client application. To support these different operational modes, THQL engines need the capability to represent each data type in several different formats, tailored to different stages of processing through which values are routed before they can be stored persistently.

In DigammaDB, the THQL Reference Implementation, persisted data storage is implemented via the WhiteDB database engine. WhiteDB is a hybrid graph/record database which allocates a persistent data store in shared memory (allowing each database to be accessed from multiple applications). ςDB encodes hypernodes in WhiteDB records (although programmers can interface with the underlying WhiteDB instances if desired). ςDB can then use WhiteDB's index and query mechanism as the foundation for its own higher-level query system. To support different THQL operational modes, ςDB organizes a stage-structure based on encoding WhiteDB values: at the ground level, values are simply pointers to in-memory C++ objects. At an intermediate level, values are encoded (via the QDataStream class in QT) into structures which recognize the WhiteDB encoding scheme but do not themselves interact with WhiteDB. Finally, values may be recorded as WhiteDB fields and records ready for persistent storage.

WhiteDB also allows databases to be shared (including sent over a network) by storing all database information in a special file format. ςDB instances can be shared via this same mechanism, although another option is to export the contents of a ςDB database to HGXF files, which in turn form the core of a research data set representing the database contents at a specific moment in time. In this guise ςDB works in conjunction with dsC, serving as the engine to construct a data set publishing research data curated via a DigammaDB database. Our dsC technology will be described in the next section.

Dataset Creator

Dataset Creator (dsC) is a framework for constructing data sets which include computer code based on the QT application-development platform. Dataset Creator takes advantage of the QT platform to construct Research Objects with exceptional GUI and data mining capabilities. QT, the leading native cross-platform development toolkit, is a comprehensive framework encompassing a thorough inventory of programming features — networking, GUI implementation, file management, data visualization, 3D graphics, and so forth. Data sets based on QT require users to obtain a copy of the QT platform, but QT is free for non-commercial use and easy to install — importantly, QT is wholly contained in its own folder and does not affect any other files on the user's computers (in this manner QT is different than most software packages, which usually demand a "system install").

By leveraging the QT platform, dsC enables standalone, self-contained, and full-featured native/desktop applications to be uniquely implemented for each data set, distributed in source-code fashion along with raw research data. Adopting such a data-curation method makes data sets easier to use across a wide range of scientific disciplines, because the data sets are freed from having to rely on domain-specific software (software which may be commonly used in one scientific field but is unfamiliar outside that field). In addition, Research Objects composed with dsC can be integrated into Multi-Application Networks (which are described above) because the dataset applications are autonomous native GUI applications that can easily interoperate via QT messaging protocols.

Because every data set is unique, each Dataset Application will necessarily include some code specific to

that one Research Object. However, dsC will provide a core code base and file layout which is shared by all dsC data sets by default. This common core is structured in part by the goal of developing Dataset Applications in a Qt context; dsC projects are structured to use Qt's "qmake" build system as the primary tool for compiling data-set code. The common dsC code therefore includes qmake project files which support compiling application with several build configurations. Data set users are also classified into several different roles — in addition to ordinary users (specifically researchers who want to work with data sets but have no development connection to the data sets themselves), dsC recognizes roles for authors, editors, testers, and other users who are responsible for bringing data sets into publication-ready form to begin with. Depending on the administrative role, data set code can be compiled with additional features (e.g., unit testing features).

Another core component of dsC is LaTeX code that authors may use when preparing documents accompanying their data set. These LaTeX files encompass special functionality for defining code annotations and semantically significant points in article text, such as sentence and paragraph boundaries. This LaTeX code can be used in conjunction with a pre-processor that generates LaTeX files from a special input language. The goal of these text-processing technologies is to improve the interoperability between research papers and data sets. In particular, the LaTeX pre-processors, and subsequent LaTeX-to-PDF converters, generate HGXF files which store information about textual and PDF viewport coordinates locating semantically meaningful elements, such as sentences and annotations. These files are then zipped and embedded in PDF files. With dsC, these PDF files can then be loaded into a customized PDF viewer which can read the embedded HGXF data. This allows the PDF application to utilize the embedded information so as to provide a more interactive reading experience — for instance, viewing annotations or copying sentences via context menus, where viewport data maps cursor position to textual elements visible on the PDF page. These features provide an application-level interface between the PDF viewers (considered as GUI components) and the corresponding GUI components in Dataset Applications.

With proper customization, both the PDF viewers and the dsC Dataset Applications can interoperate, with PDF context menus calling up windows in the Dataset Application's GUI implementation, and vice-versa. For example, researchers reading the PDF version of a scientific article can launch the Dataset Application to explore some detail mentioned in the text. This is an example of where microcitations are practically useful: any microcitable element in a document (such as a table, column, row/record, or analytic procedure formalized as a procedural asset associated with a data set) can be linked to a corresponding GUI element in the Dataset Application. For example, a statistical parameter — mentioned by name in the text, and perhaps represented in serialization within raw data — can be mapped to a GUI table column, and specifically the column header; this is then an annotation target, in the sense that for readers to gain more information it is proper to link mentions of the relevant scientific concept in article text to the column header as a graphical element that can be made visible. The link is operationalized by implementing a procedure to show the GUI window where the table is located, and ensure that the column header lies in the visible portion of the screen, as a response to readers on the PDF side signaling a desire for information on the annotated text element. In the opposite direction, database elements can be annotated with links that the Dataset Application can use to launch a PDF window opened to the page and location where the corresponding conception is discussed in the article.

In order to properly model this semantic, viewport, and data set data integration, dsC uses a new document-representation format called HTXN (Hypergraph Text Encoding Protocol). With dsC, HTXN files are not only associated with data set assets; they are also machine-readable document encodings that can be introduced into publication repositories and other corpora oriented toward machine-readability. Authors can host HTXN files within data sets and link to them via services such as CrossRef, thereby ensuring that a highly structured, machine-readable version of their papers is available for text and data mining. The HTXN protocol is also useful for encoding natural-language content which becomes part of a data set as data assets in themselves; for example, patient narratives (as discussed earlier).

Further documentation of text-encoding methodology applicable to both patient narratives and publications associated with CR2 research data is available on the CR2 web site, such as here (this is a downloadable PDF link; visit the repository to see the larger archive structure). The document just referenced contains more information on dsC, HGXF, HTXN, and other technologies discussed here.

For more information please contact:
Amy Neustein, Ph.D., Founder and CEO
Linguistic Technology Systems
amy.neustein@verizon.net • (917) 817-2184