A New Pluribus: Classification and Markup Topology for Civic Data

Nathaniel Christen

Abstract. In this paper I explore problems and solutions related to data sharing and data markup, with an emphasis on information relevant to community-oriented or "civic" applications. This can mean data for government, education, medicine, or other kinds of social and personal information which is managed by governmental and/or community organizations; or also information and presentations related to culture, social groups, ideas and platforms, or digital humanities and similar social and cultural scholarship, using the affordances of digital technology to empower cultural and civic organizations, representing both geographical and virtual, online communities. The cultural, governmental, and civic dimension affects both the kinds of data encountered and the nature of networks: issues of language, cultural sensitivity, and privacy or cybersecurity, affect data and its representation (for example, the choice to present content in multiple natural languages is not just a commercial decision, as it might be for a private business, but a conscious recognition of diversity); meanwhile, networks which carry this data will be open-ended, with different business and civic groups participating, subject to collectively negotiated regulations, and unifying technologies whose implementations evince a range of cost and expertise. Here I will discuss the semantic and security issues raised by "Civic" Data in this sense, and describe a markup language specifically designed for modeling and enabling sophisticated solutions — and inspired by initiatives related to cybersecurity (like "Pluribus", a network architecture proposed in relation to the `E` programming language), and sustainability (such as the recent "Karlskrona Manifesto for Sustainability Design").

Before there was an Internet, there were *networks*: physically separated computers sharing data. The Internet is an *internetwork*, a union of hitherto disparate networks. Some of these networks might have been better left unattached (given the dangers of remote access to critical infrastructure, like electric grids). But 20 years on, though we are more likely to think of the World Wide Web as an integral whole, it helps to revisit the picture of a federation of *sub*networks — no longer physically distinct backbones, but logical partitions reflecting geographical, semantic, and policy divisons. Emergent islands form among web sites with similar technologies, topical emphases, and user communities, particularly when these similarities dispose them to interconnect and guide how they do so.

One example is content related to specific (virtual as well as geographical) communities, which I will call "Civic" data. This can include: ⋆ Research and data sets of cultural, social, humanities, and community interest, analogous to sharing

papers and data within scientific and research communities (tending to be generated by scholarship and academic institutions); ⋆ Information about community and cultural events, giving local communities a chance to find content which interests them, and a platform for local businesses and organizations to communicate with the community (tending to be generated by businesses, not-for-profit organizations, and cultural institutions); ⋆ Content created by community members to share their ideas and projects, with other individuals and groups and institutions (tending to be produced by individuals, using a wide range of techniques and technologies); ⋆ Information provided by governmental and non-governmental institutions for the benefit of community members, presenting news, facts, or proposals pertaining to government, education, medicine, urban and neighborhood planning, and so forth; ⋆ Private or potentially sensitive information which government and/or community groups share between each other or with the community, potentially using online platforms to serve brick-and-mortar concerns, like voting, schooling, health care, immigration, and law enforcement; ⋆ Private and potentially sensitive information which is used or generated by community-oriented web services that aggregate content developed along above lines. Negotiating the interrelationships between government- and institutionally sponsored web content, and related third-party or community-oriented frameworks, can present complex challenges, especially when personal data is involved: cultural platforms may link to e-commerce services; voting platforms may link with voter registration and government services; educational and Medical platforms can generate secondary frameworks to help explain and simplify users' access to them.

Certain issues reappear: the architecture of cultural and community data; how collective community involvement affects Civic Data platforms, which must accommodate diversity in community participants' belief systems, natural language, expertise in using and/or creating "content" in various forms, and access to technology; protection of private data; the extent and limit of privacy, and how communities establish this.[1] Simplistic solutions (both ideological and technological) are counterproductive; blanket pronouncements that personal data should be wholly off-limits to (or available to) government, or either demonizing or lauding third-party data sharing, actually inhibit fine-grained solutions. The richness, nuance, and human dimension of cultural and "community" data — "Digital Humanities"; data related to cultural, social, ethnic, or linguistic groups — effectively obtaining and communicating this data, shaped in part by conceptualizations and human language, in diverse, real-world communities. The blend of concerns which can be collectively studied in terms of "sustainability", broadly understood [12]: "Green" issues like energy efficiency (the role of geography and the localization or decentralization of networks, energy costs of transmitting and/or storing data, relation of data centers to the communities where they are located and which they serve, which may or may not be the same); but, also, networks which fail to earn the trust of their users, which

[1] Including "Social" content, which may not carry the same presumption of privacy as encrypted personal data — being public is its whole point — but controversy still arises with aggressive collection of this content, manipulating social networks (including with fake accounts), and data sharing between government and social internet-related companies (often with ambiguous or contested legal and ethical guidelines).

are too easily breached or politically manipulated or appear to have no internal standards of decency (whether engineered or organic) are not sustainable in the long run. Social and environmental sutainability are mutually reinforcing.

I believe all these topics, behind the scenes, share some common features: data classification, the relation of data and meta-data, the balance between centralized and collaborative standardization, and the translations or impedence between complex human concepts and tractable formal structures. "Data" is not just key-value breakdown of information, maybe nested — this is a partial structure, losing context: what data has *not* been shared, what should the application which obtains the data do (or not do) with it, what are the intentions of the software that gathered the data into its form, what capabilities are transferred to the software that accesses it — these questions should also be directly modeled. Providing proper context, provenance, instructions — and clarifying the nature and role of specific metadata in its relation to specific data — are all issues of data *classification*, which I claim is a fundamental topic for Community Informatics.

On the other hand, "Markup Topology" might seem like a narrower and rather exotic topic. However, proper data classification goes hand-in-hand with designing correct data sharing protocols, and data sharing almost always requires some form of marked-up textual representation, for data and/or metadata. How (efficiently) data is expressed, how clearly intentions and precautions can be expressed and enforced, and how data networks interact with applications that enable and use them, are all directly influenced by formal properties of languages whose specific purpose is to encode and transmit data. Aside from their superficial syntactic difference, different markup formats also reveal different underlying conceptions of data structure, the bedrock forms upon which particular infosets are built. There is a degree of autonomy between fundamental structure and particular models: with enough discipline, information created in one format can usually be adapted to other formats. But markup languages can make working with some structures easier than others, and directly influence how projects conceive and curate their resources. The paradigms of different formats can be analyzed by considering mathematical representations of their underlying structures — trees, Directed Acyclyic Graphs, Labeled Graphs; comparing their relative expressiveness, and the complexity of parsers which support them. This in turn involves analysis of connectivity structures which address semantic network "topology" broadly understood, especially if we understand topology in discrete terms. [2] Without implying that other aspects of the data lifecycle are less important (like database engineering and application design), here I will

[2] A "connection structure" is not always a topology in the classical sense, depending on how "open sets" are understood [21]; but classical conceptions of topology can still be applied if complex "connection" relations are modeled in terms of, potentially, multiple topologies superimposed [25]: two parts of a network may or may not be connected according to different connection criteria, which collectively describe the "shape" of a network.

focus, for purpose of discussion, on markup and type structure, insofar as these can motivate a larger discussion of formal (but flexible) models for Civic Data.

Broadly speaking, rigorous data sharing demands that data has not only *structure* (in effect, a well-defined syntax), but also *semantics* (a systematic theory of how data concepts and categories relate to cognitive and empirical phenomena). But there are several different accounts of semantics, not fully isomorphic, which are all applicable to contemporary computer science and Information Technology: Ontologies and the Semantic Web; formal semantics of programming languages, especially the type systems of functional programming languages and the Typed Lambda Calculus; semantic models of Natural Language Processing, which try to partly reduce and partly formalize the nuances of human language so as to make meanings, like Word Senses and Named Entities, computationally tractable. While there is a detailed theory and science of each of these notions of Semantics, the next stage in the evolution of formal but real-world semantic theory, and initiatives like the Semantic Web, must be unification in an interdisciplinary spirit. Classes modeled by OWL-style Ontologies, for example, should be considered in terms of formal data types that may represent them for purposes of practical software application; and situating types in a formal theory means not only asserting properties of these specifications in particular, but how they engender higher-order and dependent types, such as collections and such as subtypes restricted by various tests. The relation between types and both sets and expressions needs to be clarified: types are notionally intensional, defined via indicative concepts, whereas sets are extensional, collections formed from arbitrary criteria: but for each type there is a set of its instances in some context, and for each expression applied to a type, there is a set (and possibly subtype) representing the domain for any codomain we may be interested in selecting from among all possible values of the expression. Which of these sets and codomains correspond to meaningful *types*, both in theory and in practice, is an open question. How does semantics, grounded in real-world (albeit simplified and schematized) concepts, constrain the proliferation of sets and potential (sub)types; and, in the other direction, how should formal semantic categories relate to Word Senses and other phenomena in Natural Language? Formal semantics of the Semantic Web and other network systems needs to bridge the mathematical world of type theory with the social world of human language.

Automated language understanding — like search engines, machine translation, and speech recognition — are a familiar part of contemporary commerce and technology. Nevertheless, computers seamlessly conversing with people may still be an impossible goal. To illustrate, consider how hard it is to design and implement programming languages, which in theory should be much simpler than natural language, and tailored to computers' innate representational paradigms. Despite this concordance, automated porting amongst programming languages is if anything harder than translating between human tongues; and formal languages evolve in fits and starts, as anyone who has read proposals or reviews from standardization committees can attest. While qualities like Turing Completeness indicate the theoretical equivalence of almost all full-featured program-

ming languages, partisans of different languages trump the features of their favorite dialect and the paucity of features in others. Capabilities of one language — objects, generics, continuations, lazy evaluation, monads, macros, reflection, exceptions, single and/or multiple class inheritance ... — can be unwieldy or impractical to add to others; languages tend to patch together coverage of a complex space of formal possibilities. It is not even clear what is the extent of this space — what is the theoretical maximum, the richest semantics that a formal language can exhibit? [3] For someone aspiring to design a most general programming language — or tools like an Integrated Development Environment which can serve any computer language in principle, modeling its syntax and semantics for internal use — it is not even clear what this "most general" means.

These considerations may suggest that computer languages are less removed from human language than we might think. The upper limit on formal semantic complexity may depend on philosophical details of "meaning", attested by human language and cognition: organizations of attention and conceptualization, which guide grammar and semantics, from the viewpoint of cognitive linguists such as Ronald Langacker or Peter Gärdenfors [17], [8], [28], [24]. Suppose I assert that a meeting should begin one hour after a seminar, which can have different meanings in context: if everyone knows the seminar should end at four, I can be heard as simply stating that the meeting will start at five, but in a way that provides a rationale; if the seminar's end time is not yet known, I am stating a procedure for defining the meeting's start time when all relevant information is available; if the seminar might run late, I am giving a prescription for assembling the meeting sensitive to dynamically evolving circumstances. The seminar's end time can be understood *de re* or *de dicto*: as a planned property which may deviate from what actually transpires (so I can say "the meeting ends at four", speaking of a future event in present tense), or as a future event whose time is not guaranteed until it happens. Cognitively, conversants need to assess which form of reference is intended based on context, potentially on the speaker's tone, and based on information available to them and to others in the conversation.

Such cognitive details would not seem to pertain to computers, which have no native theories of other minds, or sensitivity to tone or nuance or context. And yet the contrast in referential meaning just invoked has parallels in the contrast between evaluation modes for logical, functional, and procedural programming languages. Equality has different meanings for someone writing Prolog, Haskell or C++. These kinds of examples suggest a cross-disciplinary strategy for modeling different semantic domains and procedures, by consulting both the cognitive structures in natural language and the implementational structures of programming languages, insofar as they tackle related aspects of meaning. Concrete applications need to manage both Natural Language and network data, through the lens of data types available to and/or defined by the application. The capabilities

[3] "Universal Lambda Calculus" [19, p. 23] combines multiple type extensions, but we also have to consider "real world" coding constructs: Object-Orientation, Exceptions, Parallelism, and so on, with their own "lambda" types and calculii, and related algebraic, category-theoretic, and/or topological formulations: [1], [9], [10], [6], [7].

of programming languages, defining and managing network data, therefore offer a testing ground both for the type-theoretic formalization of network semantic models (such as the Semantic Web) and for making sense of Natural Language, particularly insofar as Natural Language is a kind of network data (e.g., on social network sites) which coexists with data about topics and entities with their own formal models. People on social media tend to talk about e-commerce and travel, news and entertainment, culture and government, domains which meanwhile provide structured data for network traffic, ranging from web searches to financial transactions. Programming language types provide an empirical anchor from which both Natural Language and network semantics can be understood. Here I will carry this perspective into the space of markup languages, assuming that segments of markup are understood to carry data structures which, in turn, are classified according to the type systems of the programming languages whose code will send and receive, encode and decode, serialize and deserialize them.

1 Network Components

Preliminary to a theory of Civic Data networks, I will provisionally distinguish different network points — web sites, applications, and services, focusing on software components (as opposed to "non-virtual" participants — community members, organizations, programmers and software design researchers — who can influence network architecture). I'll make the simplifying assumption that "personalization" is the critical contrast between sites and applications: sites are public repositories of general information, whereas web application provide online platforms giving users access to a distributed software ecosystem, often mimicking desktop functionality, but using web browsers as application frameworks. Web *services* are similar to web applications, but tend to work with other applications, rather than directly with users. Because civic institutions play a presumptive community role, they should honor limits to personalization, whatever the technical possibilities of their platform [23]. A "public record" is non-personal and preserved in perpetuity (a news forum as a "paper of record", for instance, aside from stories it may select for each reader, also provides headlines of general interest). *Avoiding* personalization, for the consistency of data across users, time, and methods of access, can pose challenges comparable in difficulty (if different in kind) from personalization. When Civic Networks *do* carry personal data, there are corresponding responsibilities, not only at each point but holistically; links from one point to another are a kind of civic endorsement, especially if these links reinforce operational and thematic connections. As Civic Data morphs from sharing general information (like concert dates and venues), to complex operational requirements (like buying concert tickets online), norms for modeling shared data become important, so that robust security can be designed, and from a perspective on the network as an integral whole.

Finally, web *sites* and *applications* should model their information with the anticipation of sometimes sharing data more in the guise of web services. As community-related information is published, by its nature there may be other

groups of people who are motivated to share, curate, and preserve this data more rigorously than typical web surfers. Information a web site deems worth sharing with web surfers, via such generic means as web pages, should also be considered as structured data used by web applications as part of a community-wide platform (technically rendering the web sites also web services).

1.1 Civic Networks as Integral, but Semantically Localized, Wholes

Civic Data points may therefore transition their role from presenting public data, to managing private data or routing it on users' behalf. As a case-study, consider a concert listing which gives users the option of buying tickets. A successful transaction will link concert venues' data to a point-of-sale application. For sake of example: a student uses the network to find a concert and buy tickets, so her student status and financial data need to be verified, the venue's available seats updated, and she may provide an address for the venue to send tickets (or be already a registered member). Here multiple transactions aggregate, each involving private data, but different pieces which can still be restricted. The venue does not need her student number, only a confirmation token of her status. So the point-of-sale and her school can use a private channel for that confirmation. The point-of-sale can be uniquely responsible for obtaining her payment info. She may have registered accounts with the venue, point-of-sale, and/or school, each of which stores different personal info. Instead of directly accessing this data, most network points can request some limited information be transferred between two other points, requiring triple authentication (as outlined below).

Making the transition from isolated web sites to Civic Data networks, developers and maintainers need to make the conceptual shift to understanding data models in operational terms related to the network as a whole, not just their one "node". This requires classifications, and distinguishing "names" from "things". Given the operations of a single hypothetical concert hall, textual labels may have an obvious "local" meaning — say, "A30" to designate a seat in the Orchestra section, and "Vienna Philharmonic" to designate a performer. But different venues label seats differently; the *Wienerphilharmoniker* has several names. The problem of semantic agreement among web sites is well-known, and a common solution is to propose global "Ontologies", which act like taxonomies and encyclopedias to avoid ambiguities in names and definitions. For Civic Data, whose cultural and societal content can be creative, innovative, or politicized, "formal" concepts have still to be flexible and allow for special cases.

This is one reason why localized networks can be more true-to-life, reflecting actual community conceptualizations rather than semantic "frame" representations whose intellectual origins lie with *Artificial* intelligence. Case study: locals think of New York City places in terms of borough and street intersections, not postal addresses; custom geospatial types, specific to New York, better represent the "local" Semantics. In general, many locales have idiosyncratic taxonomies of place, governance, culture, or civic infosets in general, hard to convey via non-localized semantic frames. Local proximity also gives developers a chance

to work in collaboration, which serves as a check on overly simplistic or *a priori* specifications. Semantic models, localized but unified, are especially valuable for sensitive data, where security and privacy adds extra layers of classification. In security-oriented, multi-agent or "distributed capability" systems [3], [14], network points have enough authority to initiate private channels of communication between other points, but sometimes may not observe the results, yielding semi-opaque "proxy" or "facet types" and "distributed capabilities". I will discuss these systems in very general terms, to illustrate some basic issues they involve.

1.2 Securing and Classifying Civic Data

In a "triangular" encryption, party A may convey something known — by party B, — to a party C, but A may not learn what this message contains. Corresponding to a given data type — for example, a students' authenticated account at a University — there is a facet representing an encrypted channel between the B and C in such a triangle; a variant of types known in some languages as *promises* or *futures*. A is granted the capability to manage certain *promise* types but not the types they encapsulate. This should be a basic model for Civic Data sites sharing information about community members, matched to user accounts in various settings (as students, medical patients, voters, financial account holders). Web sites can use encrypted keys to represent individual persons, providing each other with unforgeable codes to designate a person internally recognized. Civic Data sites, in other words, should aspire to identifying individuals via layers of indirection, using a semantics roughly expressible in natural language as, say, "the person you know as ..." (using an encrypted key), but encoded so that the reference is unique between both parties of a communication. Multiple double-encodings may be needed, building a data complex whose various parts are proxied for different parties. The complexity, sensitivity, and multi-party nature of such operations calls for carefully and collaboratively designed operational requirements. Almost inevitably, this will demand complex data classification, via a rigorous type system with polymorphic, higher-order, and dependent types, or via more recent alternatives (like Semantic Web Ontologies), or both.

While some communities may feasibly develop their own e-commerce platforms, many of the most important technological details are not community-specific. Organizations which choose to support e-commerce have various options, differing in terms of their volume, cost, complexity, risks, and effects on users. On one extreme a business can use third-party tools which are little different than electronic money transfers between individuals; on another, a business can process credit card payments directly, giving repeat customers the option of storing data for future convenience. Which point on this spectrum is right for a given organization, and which tools actually to use, are judgements that each organization has to make on legal, financial, and reputational grounds, influenced by the details of contracts which can be made or will be required with both third-party providers and eventual customers. Given these variations, it might

be argued that e-commerce is (a good example of a) "business decision" shaped more by legal and institutional considerations, than by community initiative.

On the other hand, legal and cultural norms pertaining to private and financial data do vary across territories, and communities may offer various sorts of protections or assistance vis-à-vis security breaches; moreover, criminal theft of sensitive data is at least in part a concern of local law enforcement. To the degree that e-commerce is potentially beneficial to businesses, and communities have a collective interest in local business succeeding, communities also have a general interest in helping businesses find and implement e-commerce solutions. Under these circumstances, it is reasonable for communities, even if they do not actually build wholly autonomous e-commerce platforms, to train and advise local institutions on how to utilize third-party platforms, which can lead explicitly or implicitly to some measure of community norms in terms of how e-commerce data is modeled. Similar communal concern applies to platforms and domains for e-government, medicine, education, and other potentially sensitive areas.

In all of these scenarios, personal data is important for community members' interactions with government and access to communal rights and resources, but must also be secured and monitored. Generic solutions are not designed around local norms: biometric user authentication, for example, may be construed as a greater invasion of privacy in cultures which are more guarded about public representations of the human form. This is one example; my larger point is that security and data privacy concerns are social and cultural, not only technological. The intersection of security and data network operational specifications, on the one hand, and language, concepts, and culture, on the other, require a delicate balance of formal and communal/collaborative modelings. I will in the second half of this paper explore how this balance can be supported by computer language tools, programming platforms, and software. In particular, the balance of conceptual nuance and type precision has to be supported at the programming language level, amongst different kinds of formal languages — query languages for databases, markup languages for data serialization, as well as general-purpose application languages. To illustrate, here I will focus on markup languages.

2 Markup for Civic Data

Markup topologies organize how data is modeled at all points in an exchange, because they clarify the structues which data needs to support: markup conceptualized as key-value pairs (as in JSON) compel developers to shape their data into that format, which can also influence internal data models. Similarly for tree-form structures like XML, the more text-oriented HTML, and the more graph-oriented RDF. These represent degrees of emphasis: even if interconvertible in principle, different languages make different structures easier or harder to express, which subtlely guides us to "see" those structures in data.

For a theory of data exchange it can be useful to consider markup and binary encoding together. Encoding may include *encryption*, obfuscating data so that

only select parties have a recipe to decode the data they obtain; either way, more detailed algorithms are needed to send and receive encoded data, which is also more compact and energy-efficient. Human-readable markup is more flexible, using suggestive conventions (like tag names) instead of *a priori* codes. But this is a spectrum: between binary streams and free-form markup, textual markup can be matched against more or less flexible schema ("flexible" meaning a valid schema can be essentially an alternative between related but non-isomorphic more rigid schema). Encrypted data can also be part of mostly textual markup. Overall, defining markup by contrast to binary and encrypted data is imprecise.

Instead, we can focus on how data is classified. A schema being open-ended is more salient than whether conforming code is binary or textual. So, consider "markup" as any mechanism for sharing data across computational boundaries, whether physically separate computers, or logically separated processes. With finely-sorted data types, values can have compact (potentially encrypted) encodings (and fewer bytes over the wires). Text-based encoding is more flexible, but less (formally) "sustainable"; less energy-efficient and often less secure. With advanced data classification, trade-offs between flexibility and sustainability are mitigated. A single type can have a spectrum of encodings, some more flexible and some more compact and efficient. This calls for a study of markup through the lens of type theory, with parts of markup being type-specific "constructors".

All markup arguably shares typed data values and structures (even if the types involved are broad and unspecific, like just "XML markup"). Binary and encrypted data also exchanges typed values. Given this way of studying the situation, binary data is distinguished from markup not because of how the data they carry is classified, but based on the design process for send/receive tools. For Civic Data shared via (say) XML, new applications can join the network by studying the XML code. A given document (or collection of documents) is not a guarantee that future documents will have similar structure, but it is a good heuristic. Binary data, on the other hand, can be almost impossible to understand without access to algorithms which encode or encrypt the data. Although formats like XML are "self-documenting" only by approximation, a developer, assuming that all XML documents in some context will have a roughly predictable form, can prepare send/receive logic accordingly. Documents provide only *a posteriori* evidence of this form, and are weaker guarantees than formal specifications; but, markup being human-readable, coders can develop reasonable hypotheses about proper document structure. This allows data-sharing networks to evolve in a manner which is, at least to some extent, free and decentralized.

Data sharing means creating instances of serialization types (like *XML markup*) and then using these as *constructors* for general data types; the serialization type being a facet of the primary type. Two applications which share data need sufficiently similar internal implementation of the primary types involved — to be sure that values produced at one point, then shared with the other, are reconstructed well enough. Specification requires both synchronizing each applications' representation of types, to ensure that a type like concert is (to some

approximation) "the same" type on different applications, as well as synchronizing procedures for encoding type instances via markup. The goal is that a data value (or complex data structure) is shared from one application and then reconstructed by a different application, and the resulting data is somehow "the same". Criteria of identity can vary here, but certainly must agree with respect to their interactions with the surrounding world: a data type like *seat* refers to something in the world, and seats are "the same" according to commonsensical assumptions (like it being an error to sell multiple tickets for the same concert seat). Markup remotely conveys typed values, sometimes grounded in empirical things, which impose criteria of identity, through which we can reason about how two values can be "the same" [13]. Remote data sharing succeeds when values sent are (on proper criteria) "the same" as those received (once reconstructed).

Markup plays different roles in a data exchange operation: it may describe human-targeted views, or also "templates", less data structure than visual or textual formats which become presentation when "woven" with specific data. It may also be a more formal data representation; or a template for data, either defining a schema prototype or serving as a query language, using partially complete representations to obtain a set of matching values. This contrast plays against a backdrop of distributed operations, conveying data between disparate points: for each such line of communication, there is markup serving one or more of the above roles. How markup is organized therefore follows directly from operational requirements. A good grammatic architecture will allow these to be declared and confirmed. To varying levels of precision, this can be a matter of markup structure, of types, and of specific values and "semantics". For example, *structural* requirements (say, a markup fragment for a concert must have a *performer*, *date*, and *venue*) are different from type requirements (say, a *date* must be valid according to one of several date-time encodings). More fine-grained "value" requirements can request markup to declare (in response to a search, perhaps) that a given date falls during the first week of the month, or that a number (say, the cost of a ticket) falls within some range (say, under $100).

The association of markup segments with data types depends to some degree on matching classifications of markup with classifications of data. The latter can, however, be more or less fine-grained. For a general expression stating some criterion (say, a "seat under $100"), should we propose a *type* of just those values which match? For complex type systems, arbitrary formal code may be needed to specify some types, which are therefore hard to encode in markup (except indirectly as character data). Suppose an XML tag were written as "<price currency='$US' criterion='< 100'>150</price>". This *looks* wrong, because it declares that something costing $150 fits a field matching criteria narrowing to a range under $100. But such a semantic or logical confusion would not make the markup "ill formed" on syntactic (or even structural) grounds. Achieving this more precise and *semantic* verification belongs more to the aspirations of the Semantic Web, where data can be broken down into arbitrarily small units and each individual unit separately annotated. In a Semantic Web format, like RDF, the minimal unit of information is a "triple", which can be visualized as an arrow

pointing from some small unit of data to a description, characterization, or qualification on that data, and each unit can have many arrows. Therefore, many numeric and semantic details can be specified; however, the obstacle here is that data complexes are broken down into elementary units, like lego building blocks, and specifications are needed to reconstruct the whole from its parts.

Whether more tree-form (like XML) or more graph-form (like RDF), serialized data contains smaller parts (or "nodes") that have to be reassembled. Just as the same ingredients can make a cake, soufflé, or omelet, markup nodes may not carry enough information to reconstruct a whole; instead there must be some external specification or convention. With XML, the tree structure at least sets parameters on how nodes fit together (each node has exactly one parent, in an ordered group of siblings). RDF is more flexible, but with fewer aggregative norms *a priori*; to serialize data via RDF demands even more disciplined encoding rules. There are also (less well-known) languages whose expressiveness lies roughly between XML and RDF. XML has only single parentage (one parent for each child node) and no interleaved tags. To "extract" formal data, which has been woven together with visual and textual features and must be re-assembled from different points in the larger document, is commonly expressed (as in RDFa) by conventions like special attribute codes, that are ad-hoc regulations, not intrinsic to a markup language. A more formal methodology would model data markup directly as a separate document hierarchy interspersed with the main document. This option is available to languages with "concurrent" markup, but only indirectly to XML or HTML. Languages more complex than XML, RDF, and HTML can therefore better describe how to build intended wholes from provided parts. The expressivity of a markup language is closely related to the mathematical form which a markup syntax takes on, when we consider how nodes as data units are aggregated into complex wholes. RDF considers labeled graphs; XML uses trees; other languages, between their respective flexibility, model various narrower kinds of graph, often based on *directed acyclic* graphs (DAGs), where nodes may have many parents. The mathematical properties of markup structure influence the algorithms which applications must implement to convert markup to and from specific data values. A markup language therefore should be chosen which is expressive enough to allow data to be encoded and decoded systematically, but restricted enough that the proper algorithms can be selected. The relation between markup and its encoding/decoding algorithms describes the interface between markup (which exists outside and between different applications that us it) and the applications themselves: given a marked-up document, or a segment thereof, an application needs to know what to do with it — which algorithm to select, so as to reconstruct some particular kind of information from the jumble of nodes which are the document's bricks and mortar. Just as data structures, when initialized from markup, imply that constructor functions have been used that take markup as input, choosing algorithms to process markup implies type-specific functional resolution. For Civic Data, these selections can be guided by communally established type systems. The challenge is to design these while recognizing plurality in developers' coding languages and platforms.

3 A Specific Markup Language Case-Study (NGML)

There are several limitations of the most popular current markup languages. XML can be taken as a case in point. This does not mean that XML is inadequate against these limitations; it is certainly possible to use XML for networks requiring the relevant features. However, this depends on conventional use of XML features, in contrast to features specifically designed into XML. Potentially important features, not directly modeled in XML, can be identified at multiple levels, from small scale to large. In terms of individual words and sentences, XML (and HTML) are not directly concerned with (natural) linguistic phenomena, such as words and sentences. While there is nothing preventing users from designing XML tags to represent, say, individual sentences (and XML encodings for linguistics do precisely that [15], [26]), integrating sentence- or word-level markup with other markup can yield some challenges or ambiguities. [4] Moving upward, for digital publishing and text encoding, designed to generate or reproduce physical books and manuscripts, there are high-level patterns do not logically fit the structure of XML (even if they can be simulated). For example, when building electronic repositories which are faithful to print resources, affordances like searching and indexing and statistical analysis will, ideally, coexist with faithful reproduction of print details, even errors, insofar as these may be relevant for scholarship.

There is an analogous logical mismatch at the level of structured data encoding, and the scale of general application integration: using (say) XML to serialize data structures. Data typing is one example: XML Schema, designed apart from XML itself, is weaker than programming type systems in general, achieving only a "least common denominator" for data specifications. These are useful heuristics but not a comprehensive model, given type relations involving proxying, references, generic or higher-order types, and so forth. Finally, XML does not express certain semantic properties of data networks as a whole, at the scale of a distributed platform: norms of who can access what data and the intended use of information serialized by markup. Another fairly general limitation of markup languages reflects the formal incompatibilities between different data structurations [18]. Semantic Web content is intrinsically graph-form, with potentially many "edges" emanating from each node. Most markup languages are instead tree-form, or at least acylic, with "parent" nodes and therefore a direction of "descent" among node paths. Encoding graph data in a strict (acyclic) subset of graphs requires somewhat awkward conventions, typically via some notion of references, allowing one "logical" node to be expressed with many "lexical" nodes; however, acyclicity ensures that some important traversal algorithms terminate.

A markup language is intrinsically a *logical* structure. "Internally" markup is nothing but a sequence of characters, and wholly one-dimensional; it has no "containment" or "parthood" in that sense. We may speak of XML tags as *nested*,

[4] For example, there is no way to indicate when a markup tag does or does not separate words (occasionally one wants a tag to affect one *part* of a word, isolating perhaps a single syllable so as to suggest spoken emphasis). Sentence-level markup can also cause interleaving: e.g., a long footnote in the middle of one sentence.

but this involves logical abstraction: a region of characters is *logically defined* as that between start and end tags. It is a syntactic rule by fiat, to declare that one symbolic form (say, </quote>) "means" the end of a region whose start was earlier (with a corresponding start-tag, like <quote>). In the same way, "extensions" to XML involve its *logical* structure, not XML as used (with suitable conventions) in practice. A real-life arrangement, like page-and-paragraph interleave, can be encoded with XML, even if one needs a mechanism other than nested tags (like "milestones", empty tags which serve as place-markers, maybe meant as spanning text between them). Nevertheless, a well-designed formal language gathers considerations of semantics, syntax, and logical structure, so as to make certain kinds of meanings and patterns concise and intuitive to express. These dimensions are interrelated, because the semantic meaning of a concept is determined by how its component ideas are assembled, and a syntactic expression reproduces this logical structure by how syntactic rules break a string of symbols (or spoken units) into parts. Logical structures on the semantic side are mirrored, with perhaps some well-defined transformations, by those on the syntactic side. The kind of logical structures which are directly modeled by a language's syntax therefore indicates the kind of semantic structures which the language most readily encodes. The logical structure of XML directly supports nested tags, because the distinction of start and end tags is so central to its syntax (and in XML code is visually clear and simple). For this reason XML is naturally suited for hierarchical data, and more widely used for this purpose, even if flat-text formats could represent nested data if needed (by inventing ad-hoc symbolism analogous to tags). By the same reasoning, because XML does not support concurrent markup, it does not readily lend itself to data models for which concurrent markup is natural. XML can certainly be used for such structures, given suitable conventions, but the language does not guide the data model in this case; the model is designed from a different perspective and operationally adapted to XML. XML can also encode Semantic Web data, but again does not lend itself to conceptualizing data in these terms; this is why Semantic Web developers prefer to "think" in a format like RDF, which matches their models better at a conceptual level.

Any formal language (at least which can be coded into a computer) reveals a fundamental computational pattern which can be mathematically represented in systems like Lambda Calculus. For example, XML tags can be treated as "computations" whose child nodes supply arguments — so an *italics* tag is really the declaration of a computation, which results in all its range being italicized. This may seem circular, or uninterestingly self-evident — the result of an italics operation is italicized text — though we might also consider that to *render* italics requires some calculations, and the tag states a command that those be performed. It is not a specific computation but a declaration that one is needed (the actual computation to italicize text will depend on many factors, such as the current font face and the device where the text is viewed). Similarly, Lambda Calculus does not *perform* computations; it systematically outlines computations which need to be performed, specifically so as to construct results which then become arguments to other computations. Lambda Calculus recognizes interrelationships

between calculations, and ultimately these take on four kinds: entering a new calculation, concluding a calculation and proceeding with a prior one, crossing between two successive calculations, and the sequential relation between successive values which are arguments for calculations. From this breakdown we can define a simple "Lambda 4" Ontology that can encode any finite computation, and, as such, any computer language. Any programming language (including markup) can in theory be expressed using just this four-valued Ontology [4].

This depends, however, on describing computations at a purely syntactic level. For real languages (including formal ones), the distinction between syntax and semantics is less clearly defined. Among human languages, this is an insight of Cognitive Grammar — the fundamental grammatical categories, which are needed for syntactic rules, are also themselves broad semantic classes (nouns, verbs, adjectives, adverbs) [16]. Syntax is not the mindless morphology of language "games" but a model of how logical structures in meaning can be efficiently represented by logical structures in symbolic and enunciative organization. This lesson should be applied to formal languages as well. While the pure abstract form of computational interactions can be modeled in terms of "$\lambda 4$", a practical computer language will adopt other interrelationships, that declare more specific interconnections between computations. Among markup languages, there are many relations that can exist between markup "nodes" and the text which they influence. XML and HTML relate tags to namespaces, attributes, character data, and child and sibling nodes; LaTeX relates "commands" to both optional and mandatory arguments, and also relates text to "environments". Languages with Concurrent Markup connect different regions, where tagged regions may overlap, to the tags involved, incorporating at least a subset of the Mereotopological relations provided by "Region Connection Calculus" [11]. RDF relates triples to a subject, object, and predicate, allowing triples to be "reified" (described themselves as RDF resources, with their own internal network of relations).

By identifying relations which are important to the syntax — for declaring them and for their rationales — of different markup styles, a language can help translate between them. In NGML, nodes represent either "textual data" or "tag commands", taking terms from both XML and LaTeX: these may just "tag" some text but can also transform it. Spans of textual data are called "tiles". Between and among tiles and tag commands, relations endemic to different markups are possible — mandatory and optional arguments from LaTeX, attributes and tag bodies from XML, *subject-predicate-object* triples from RDF. Coders can extend the language by declaring new relations, as well. Relation-types (essentially a semantic or Semantic Web notion) describe syntactic constructs of markup languages, with which NGML seeks to interoperate. NGML also tries to interoperate with data systems by explicit type associations: types can be associated with NGML markup streams over all, NGML documents, tiles, and tag-commands, both a tag-command in isolation and seen as a container for nested content.

Markup validation can be defined in the narrow sense of confirming to a certain proscribed structure, or in the broader sense of semantic validity, conforming

to requirements on values, types, or the manner of access which markup confers to associated resources. *Security* validation ensures that markup is properly used. Markup itself can yield vulnerabilities (for example if the task of validating it results in data being sent over a network); at the same time, any markup code is an example of data being shared between two parties, and should be evaluated. Does the markup share more data than necessary, or intended? Can malicious parties use the data provided to obtain more? Any markup confers on its recipient a capability to reconstruct at least some information which is known to a sending application. However, these capabilities can be minimized. Data as sent can be potential access to information, not direct representation of that information. Any application which holds a document has a capability to examine all data it contains, but a capability to *examine* (say) encrypted data is not a capability to *read* it. Given these considerations, application (or "document holder") capabilities are modeled semantically. A document can declare further capabilities to its "holders": the software that sends and receives it. such as the capability to *request* a key to decode encrypted data, or to *transmit* that data to some other holder. A holder of encrypted Credit Card data may not acquire a capability to read this data, but can pass it on to a point-of-sale application.

For "security" validation, document holders are actors in a multi-agent system: each network point has an autonomous role with respect to the information represented (or parts thereof). Abilities to create, read, modify, share, or respond to information are distinct role kinds, mapped by markup between document holders and particular markup segments. While no guarantee that the holder will abide by its role, this provides a foundation on which encryptions can be designed; to match an encrypted data value with an encrypted representation of a document holder's permission to act on that data, or send it elsewhere, for example. Send/receive points must then implement these protections, but markup can nudge developers toward designing data networks on secure principles.

The larger point is that network security and user experience are interconnected: all software points need to bridge formal semantics (type systems, object-capability models) with human language and concepts (word senses, named entities). When this is robustly done, the security, usability, and visual/interactive sophistication of network tools and protocols will improve accordingly.

4 Conclusion

Civic Data is "civic" by intent, as well as the kind of data it tends to model (geospatial locations, government affairs, culture and society). It solicits applications that serve common interests, sensitive to the culture, norms, languages, and history of communities. Civic Data is intrinsically interdisciplinary because its semantics have to meet both formal and conceptual criteria; it calls for a deep science interrelating conceptual and formal types: not as an academic exercise, but theory and practice unified in sustainable and culturally sensitive ways.

References

1. Martín Abadi and Luca Cardelli, *An Imperative Object Calculus: Basic Typing and Soundness*. http://lucacardelli.name/Papers/PrimObjImpSIPL.A4.pdf.
2. Diana Allam, *et. al.*, *Well-Typed Services Cannot Go Wrong*. Project-Teams INRIA-ASCOLA, Research Report no. 7899, May 2012. https://hal.inria.fr/hal-00700570/file/main.pdf
3. Shahriar Bijani, *Securing Open Multi-agent Systems Governed by Electronic Institutions*. Doctoral Dissertation, Edinburgh, 2013. http://www.dai.ed.ac.uk/groups/ssp/members/pubs/bijani_phd.pdf
4. Nathaniel Christen, *Perspector Algebra and the λ4*. https://github.com/scignscape/Re-Dev/blob/master/L4.pdf
5. Pierre Corbineau, *et. al.*, *Position paper: A real Semantic Web for mathematics deserves a real semantics*. SemWiki2008. http://ceur-ws.org/Vol-360/paper-16.pdf
6. René David and Georges Mounier, *An intuitionistic lambda-calculus with exceptions*. Journal of Functional Programming, Cambridge University Press, 2005, 15 (1), pp.1-20.
7. Simon Dobnik, *et. al.*, *Spatial Descriptions in Type Theory with Records*. 10th International Conference on Computational Semantics (CoSLI-3), 2013. http://www.aclweb.org/anthology/W13-0701
8. Peter Gärdenfors, *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2000.
9. Joseph A. Goguen, *What is a Concept?*. https://cseweb.ucsd.edu/~goguen/pps/iccs05.pdf.
10. _____, *Hidden Algebra for Software Engineering*. http://homepage.cs.uiowa.edu/~fleck/181content/algse.pdf.
11. Torsten Hahmann, *A Reconciliation of Logical Representations of Space: from Multidimensional Mereotopology to Geometry*. Doct. Diss., Toronto 2013. http://www.cs.toronto.edu/~torsten/publications/Hahmann_PhD_thesis.pdf.
12. Christoph Becker, *et. al.*, *Sustainability Design and Software: The Karlskrona Manifesto*. http://www.cs.toronto.edu/~sme/papers/2015/Beckeretal-ICSE2015.pdf
13. William Kent, *The unsolvable identity problem* Extreme Markup Languages 2003, Montréal, Québec August 4-8, 2003 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.6902&rep=rep1&type=pdf
14. Shriram Krishnamurthi, *et. al.*, *Features and Object Capabilities: Reconciling Two Visions of Modularity*. AOSD (International Conference on Aspect-Oriented Software Development) 2012, Potsdam, Germany. http://cs.brown.edu/~sk/Publications/Papers/Published/sfk-feat-ocap-reconcil/paper.pdf
15. Nancy Ide and Keith Suderman, *The Linguistic Annotation Framework: A Standard for Annotation Interchange and Merging* Language Resources and Evaluation, September 2014, Volume 48, Issue 3, pp. 395-418. http://www.cs.vassar.edu/~ide/papers/ide-suderman-LRE-LAF.pdf
16. Ronald Langacker, *Discourse in Cognitive Grammar*. Cognitive Linguistics 12-2 (2001), 143-188. http://terpconnect.umd.edu/~israel/LangackerDiscourse2002.pdf
17. _____, *Foundations of Cognitive Grammar*. Two Volumes. Stanford University Press, 1987.
18. Emanuelle Laponi, *et. al.*, *Off-Road LAF: Encoding and Processing Annotations in NLP Workflows*. Proceedings of the Ninth International Conference on Language Resources and Evaluation, Reykjavik, Iceland, 2014. http://www.lrec-conf.org/proceedings/lrec2014/pdf/978_Paper.pdf
19. Daniel Lincke, *A Transformational Approach to Generic Software Development Based on Higher-Order, Typed Functional Signatures* Doct. Diss., Technische Universität Hamburg-Harburg, 2012. https://tubdok.tub.tuhh.de/bitstream/11420/1073/1/Lincke.pdf
20. Mark Miller, *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. Doct. Diss., Johns Hopkins, 2006. http://www.erights.org/talks/thesis/markm-thesis.pdf
21. Joseph Muscat and David Buhagiar, *Connective Spaces*. Mem. Fac. Sci. Eng. Shimane Univ. Series B: Mathematical Science 39 (2006), pp. 113 http://www.math.shimane-u.ac.jp/memoir/39/D.Buhagiar.pdf
22. Birgit Penzenstadler, *Infusing Green: Requirements Engineering for Green In and Through Software Systems*. http://ceur-ws.org/Vol-1216/paper8.pdf
23. Eli Parisier, *The Filter Bubble: What the Internet Is Hiding From You*. Penguin, 2011.
24. Martin Raubal and Benjamin Adams, *The Semantic Web Needs More Cognition. Semantic Web Journal*, 2010. http://www.semantic-web-journal.net/sites/default/files/swj37_0.pdf
25. Marko A. Rodriguez and Jennifer H. Watkins, *Grammar-Based Geodesics in Semantic Networks*. Knowledge-Based Systems, 23(8), 844-855, December 2010. http://arxiv.org/pdf/1009.0670.pdf
26. Chunqi Shi, *et. al.*, *A Conceptual Framework of Online Natural Language Processing Pipeline Application*. Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT, pages 5359, Dublin, Ireland, August 23rd 2014. http://glicom.upf.edu/OIAF4HLT/pdf/OIAF4HLT06.pdf
27. David I. Spivak, *Database Queries and Constraints via Lifting Problems*. http://math.mit.edu/~dspivak/informatics/LiftingProblems.pdf.
28. Gregor Strle, *Semantics Within: The Representation of Meaning Through Conceptual Spaces* Doct. Diss., University of Nova Gorica, 2012.
29. Maik Stührenberg, *The TEI and Current Standards for Structuring Linguistic Data: An Overview*. Journal of the Text Encoding Initiative, 3 (November 2012). https://jtei.revues.org/pdf/523
30. Fernando Rosa Velardo, *Typing Techniques for Security in Mobile Agent Systems*. Master Thesis, Universidad Complutense de Madrid, 2004. http://antares.sip.ucm.es/frosa/docs/trabajo.pdf