InfoCentral and the Semantic Web

An alternative vision for global-scale property graph data models

Draft – Updated March 5, 2019

by Chris Gebhardt

Short explanation of relationship to the Semantic Web

“InfoCentral proposes a comprehensive retooling of the Semantic Web for a fully-decentralizable internet. It mandates a network and information architecture with ***exclusive cryptographic hash referencing*** of data resources. In conjunction, a ***reference collection model*** allows continuous discovery and propagation of new data. As these axioms permeate designs, InfoCentral aims for **identical expressivity** at the semantic graph level.”

InfoCentral is ...

- Not strictly a “Semantic Web project”
  - It does not depend upon any Semantic Web technologies at the foundation.
  - Compliance with existing W3C standards is not a project-level requirement. (clean slate, open to break stuff, etc.)
- Not strictly a replacement for Semantic Web
  - We love SW and build upon the same foundations.
  - SW technologies will be used and adapted whenever possible.

Sounds painful! Why do this?

- Lack of reference stability is the Achilles heel of the classic web. It drives users to large centralized services to regain stability.
- The reference model ***must*** change:
  - Semantic Web / Linked Data: names to mutable data
  - Infocentric architecture: hashes to immutable data
- Data management becomes easy:
  - conflict-free, multi-party data layering
  - default versioning (guaranteed unique ID per revision)

The technical strategy..

- Mandate immutable identity and versioning semantics. (for reliable referencing & portability)
- Guarantee independence from centralized data storage and authoritative naming. (DNS, etc.)
- Require fine decomposition of graph data as it is persisted into immutable data entities.
- Promote client-side cryptography over server ACLs for protecting private user data
- Ruthlessly eliminate unneeded complexity, redundancy, and legacy cruft (clean slate design)

What about?..

- Don't people want/need meaningful naming?
  - Meaningful naming implies reference instability. (mutability, authoritative services, etc.)
  - Naming belongs in metadata, not data or network architectures. (Proper separation of concerns..)
- Lost efficiency from aggressive decomposition into small graph data entities? (vs. larger docs)
  - Canonical persistent data entities are usually ingested into more efficient local data structures.
  - Network protocols support batch and differential transfer. Repositories will internally aggregate related data.

Hash Reference Challenges:

Changes to basic data modeling aspects

- In contrast to mutable named documents, cyclical references are not possible with hashes.
  - Data entities A and B cannot refer to each other, but C can refer to both and indexing allows bidirectional traversal.
- Projection between the graph and data model is rarely 1:1.
  - The data entity graph is acyclic; the semantic graph model is not.
  - The data model contains security and management details not needed by the semantic graph model.

Hash Reference Challenges:

Changes to language design and software architecture

- Heavily favors declarative / functional paradigms
  - Elimination of persistent data mutability (append only) opens the door for massive inversion of control..
  - Logic alongside to support fully-independent data
- Strong typing and semantics, not encapsulation.
  - Validation through public schemas and ontologies
  - Business logic is external but context-preserving
- Dynamic composition, not pre-compiled apps
  - User is free to adapt their information environment at will, without risk of losing data or context.

Managing atemporal data

- The RDF data model is atemporal. Statements in the graph have inherent self-identity and immutability – any change makes them a different statement.
- However, web *containers* of RDF data are not atemporal. Statements come and go. New statements may replace existing statements, with or without annotation that this signifies a related real-world change of state.

Managing atemporal data

- A practical breakdown in nominal atemporality is that the context of an RDF statement is ambiguous. It may be later enhanced by layering metadata, effectively making it a different statement. (and effectively temporal)
- Example:
  - Pittsburgh's air temperature is 50°F.
  - Pittsburgh's air temperature is 50°F, said John Smith[PK_ID], on 20170123-15:38:22.31UTC, at location [80.1254,42.5315].
- There may exist many identical “Pittsburgh's air temperature is 50°F” statements in the global graph, but they are not the same statement when context is considered.

Managing atemporal data

- The dimensionality needed to make a statement unambiguous is sometimes unknown at the time it is initially made.
- Using reification to annotate a statement gets ugly very quickly. (ex. reifying a previous reification of a statement) This was abandoned.
- Restructuring a statement using blank nodes, to create an n-ary relation, causes it to become a different statement. It also prevents external annotation, and future changes must occur in the subgraph containing the blank node.

Managing atemporal data

- SW struggled with this issue for a long time. There were proposals to increase arity of RDF tuples beyond the subject, predicate, object triplet.
- Solution: RDF named graphs – create addressable identities that may be the subject of future metadata. (most notably provenance)
- However, fine-grained named graphs are not mandatory. The scope of addressable graphs may be huge and have no fragment IDs, making annotation impractical.

Managing atemporal data

- Our graph data model has stricter decomposition:
  - Statements exist in ***immutable*** 'named graphs' (Standard Data Entities, which are not named but hash-referenced.)
  - Statements in an entity have a ***single external subject***. (Entities may also contain self-statements)
- We provide a shortcut for statement reification. A statement (and optionally its components) in an entity may be referenced by a simple numerical index.
  - Hash_ID + statement_index [+ component]
  - This is not visible at the network level, where only hash identities are used for metadata aggregation.
  - This indexing method allows a statement's predicate to be separately annotated, a novel expressive ability.

Persistent Data Model to Statement Graph Projection

- Not everything in the base data model needs to be projected into the statement graph at all times.
  - ex.) identity-grounding artifacts like nonces and data management oriented metadata
  - We often use a curation step (revisions, trust, etc.) when creating a **graph view** for a particular scenario.
- An IC hard rule: Never create graph statements without creating data entities for them to live in.
  - In other words, statement graph data must always be immutably projected onto the base data model at creation.
  - This provides a standard identity and data management layer that is currently ad-hoc with URI+RDF.

Major differences

- InfoCentral's scope covers many aspects of network and software architecture, which are outside the domain of the Semantic Web and its primary focus upon knowledge representation and machine reasoning.
- As a broader, multidisciplinary approach, we believe there is justification for breaking strict compatibility with well-established standards that get in the way of needed architectural shifts.

Major differences

- InfoCentral data model prohibits all current URI schemes, in favor of a native hash-only scheme.
  - Mixing mutability semantics is hazardous and unnecessary.
  - Legacy URI schemes carry enormous complexity.
  - The new identity infrastructure has wide-reaching effects and must become ubiquitous to work as intended.
- Bridges to legacy web systems:
  - Mutable data may be sampled into native data entities.
  - Entities may reference legacy URIs, but only in a bibliographic sense, not as first-class references.
  - Web systems may use the IC Persistent Data Model behind the scenes. (Though overhead is higher than traditional DBs..)

Major differences

- InfoCentral re-uses a subset of Semantic Web technology, with a few extensions.
- We rigidly disallow some features of the RDF data model that don't fit the architecture.
  - Explicit reification syntax (ugly, unneeded)
  - Labeled blank nodes
    - for concrete subjects: unjustified as there is near-zero cost to create and maintain a hash ID (ie. new root entity)
    - for abstract subjects: most existential variables will likely be handled separately, TBD. (some unlabeled bnodes allowed)
- Our proposal subtly varies from RDF semantics. This will need to be formalized.

Otherwise, the W3C Semantic Web standards are well-engineered with respect to separation of design concerns.

RDF / OWL / RIF / SPARQL don't depend on HTTP or other URI schemes.

OWL, RIF, and others are abstract until they are mapped to RDF semantics. Only minor rework will be needed here. SPARQL would be implemented in higher software layers, along with graph stores. It is abstracted from the low-level data model.

InfoCentral is more concerned with aggressive divergence from stale Internet / Web architecture.

Practicality speaking.. Semantic Web has a small installed base, manned by smart people who are quite capable of doing a conversion. Now is the time to fix the problems!

Unfortunately..

Most practical Semantic Web tools and products are strongly convoluted with traditional Web architecture.

ex.) All existing OWL ontologies must be converted from using HTTP URIs to Hash IDs, before use in InfoCentral. We suspect the best strategy is to create fresh ontologies, using existing useful ones as mere templates, while applying design patterns better suited for the new, collaborative regime.

General assumptions

- **Semantic Web**
  - *vision*: a world of authoritative but largely-independent publishers of incomplete graph data
  - *solution*: allow domain-based silos, aggregate as much graph data as possible, filter on provenance as determinable, and use logical inference to de-duplicate and fill in the gaps
- **InfoCentral**
  - *vision*: a world of socially-networked collaborative publishers, who continuously improve and consolidate information within a fully shared graph (Git / wiki-like)
  - *solution*: fully decentralize information, develop trust networks and communities to guide the process of producing and curating quality information

Hash-based Identity Changes Everything..

- Semantic Web data is bound to URIs, which are subject to arbitrary change in availability and what they point to. Modifications must pass through gatekeepers at each authoritative source. This frustrates fluid collaboration between data publishers and limits involvement.
- InfoCentral data is immutable and bound to a nameless and location-free hash identity which can never change. This encourages all parties to collaborate on the same distributed graph, knowing that references will never go stale, availability cannot be taken away by any one party, and anyone can directly publish new and modified data. (ie. Git-like)

Primary Focuses

- **Semantic Web**
  - authoritative publishing of semantically rich information over traditional host-based (centralized) networks
  - automated inference and reasoning about graph data
- **InfoCentral**
  - globally collaborative production and refinement of semantically (and socially) rich information
  - new software architecture for working with graph-structured information
  - decentralized systems and information-centric networking
  - Use AI today to boost productivity of semantic labeling

These are mostly complimentary!

(If we can get everyone on board..)

Approach comparison

- **Semantic Web**
  - Build as many bridges as possible to existing technologies. Adopt designs that incorporate the needs of existing and legacy systems, both technological and social.
- **InfoCentral**
  - Start from scratch and build a new software ecosystem. Accept that, given the extent of needed change, the cheapest and easiest option is often full conversion of legacy systems and information. (But don't be closed to building bridges either...)

Approach comparison

- **Semantic Web**
  - Enhance existing software applications with semantically-rich content. Attempt gradual, voluntary conversion of the document Web into a hybrid Web of data and documents.
- **InfoCentral**
  - Obsolete the very concepts of applications and the document Web. Start a software engineering revolution that genuinely excites developers.

Approach comparison

- **Semantic Web**
  - Rework familiar UI paradigms and metaphors, in hopes of making the new technology accessible with reduced up-front investment.
- **InfoCentral**
  - Explore completely new UI paradigms native to graph-structured information. Recognize that true progress will mean breaking *everything* here. The higher up-front cost will be offset by plummeting sustaining costs as network effects emerge.

Semantic Web's Commercial Issues

- Incompatible with current software architecture and economics
  - Incorporation of Semantic Web technologies costs more and the added openness only benefits competitors. Yet open network effects are required to add value, for mainstream applications.
  - The typical developer of silo'ed applications (or walled-garden cloud software ecosystems) doesn't see any tangible benefits.
  - ex.) If Facebook fully adopted the Semantic Web, it would lose control and visibility of how people are using its social graph. (its cash cow!)
- Limited practical application
  - SW adds little to monolithic software architectures, relative to the latest generation of graph and document DBs, machine learning tools, etc.
  - Benefits for large graph databases in some scientific research
- Unmanaged complexity
  - severe learning curve, inaccessible to the typical developer (though better tooling could help here..)

In Summary

- The Semantic Web project has produced some great tools and techniques for creating, managing, and reasoning with semantic property graphs.
- Legacy web architecture was ideal when invented but is now is holding us all back.
- The Semantic Web has failed to gain acceptance as a result.
- Let's build the **Global Information Graph**!