Distributed Knowledge Representation using Universal Identity and Replication

Peter Lucas, Jeff Senn, Dominic Widdows
{lucas,senn,widdows}@maya.com

MAYA Design Inc.
Technical Report MAYA-05007

Abstract. This paper describes a pervasive, scalable, and extensible design for distributed information objects. The system is designed so that all information has persistent identity, independently of the devices used to store and transmit information.
The system relies on the notion of a *u-form*, which is a bundle of attribute-value pairs indexed by a universal identifier. The Universal Unique Identifiers (UUID's) in our system are more ubiquitous than the typical uses of the URI or URN references of the World Wide Web, since even mundane objects are given UUID's that are guaranteed to be unique, even if they have no fixed location and are not objects in any other managed namespace.
UUID's allow u-forms to refer directly to one another, irrespective of their source or physical location, enabling the construction of a global semantic network. Requests for replicates of u-forms are managed by artificial agents called *shepherds*, whose goal is to promote consistency across multiple repository venues in a peer-to-peer network.
U-forms are not bound by any one schema and new attribute-value pairs can be added as appropriate. Schemata can be layered onto u-forms by adding relations to *role* u-forms, which state the intended interpretation of the attributes. The system is mature and has been deployed in a number of governmental, environmental, geospatial and human-services projects, some of which are presented as case studies.

1 Introduction

Over recent decades, it has become increasingly clear that, in order to build a machine that demonstrates useful and broad intelligence, access to a rich and varied yet coherent supply of facts about the real world is as vital as a robust reasoning capacity. However, the available knowledge representations suffer from being highly specialized, inextensible, or too shallow and unstructured. This paper describes both the abstract and physical components of a Universal Database architecture for knowledge representation, which we believe is a natural and simple foundation for solving this problem.

The representation is derived from both abstract and practical considerations, and is designed to be comprehensive, flexible, distributed and evolving. It

is based upon the concept of the *u-form*, an abstract datatype that consists of a unique identifier and a bundle of attribute-value pairs. A u-form is by definition extensible and does not have to conform to any fixed schema; however, metadata or *roles* can be included in u-forms that express the way the attributes and values are to be interpreted. The u-form is an abstract data-object, though to build any real system it must have a physical representation. This is provided by a network of data repositories, which pass u-forms to one another through a process of *shepherding*. The architecture is mature and has been used in a number of government, private research, community information, and commercial projects, and its goal is to provide a fully interoperable store of public knowledge called the Information Commons.

The sections in this paper discuss the properties necessary for a universal knowledge representation system (using a variety of current examples as motivation), the *u-form* datatype and its deployment in a large peer-to-peer network, metadata schema or *roles* which may be layered onto u-forms to enable their correct interpretation, and finally, examples that demonstrate the usefulness of persistent data identity and extensibility in a large data-modeling and navigation project.

2 Requirements for Universal Knowledge Representation

This section describes some of the properties that a truly global knowledge base would need in order to be successful. Our goal is not to build a single system that provides input data for a single artificial intelligence application, but to create the infrastructure for a system that can represent the knowledge required by *any* artificial intelligent agent in the future, and (this is crucial) encourage an information culture where this representation is readily available.

Object Extensibility One of the key ways humans acquire world-knowledge is by gradually discovering more information about individual objects or groups of objects. It is thus vital to be able to add information to the knowledge representation of a single object, without necessarily adding attributes or variables to all cognate objects. This is one of the key benefits of extensible formats like SGML and its descendant XML, in which new tags can be introduced at any time to represent new kinds of information. This is extremely difficult in traditional relational databases or ontologies, where schemata have to be decided in advance and are costly to re-engineer.

Data Liquidity Any universal database must be vastly distributed, both in space and across many diverse devices. This stipulation implies the need for a high level of data liquidity, that is, the ability for data to flow readily from venue to venue. Currently, the World Wide Web is the closest example of such a system, though for some devices the currency unit of a Webpage is too large and nonspecific (for example, a handheld device should not give you a list of subway stations; it should tell you how to find the nearest one). Also, each webpage is tightly coupled with the server that delivers it — data objects have no way of

moving between locations without losing their intrinsic identity. This problem is still present with the Universal Resource Indicators (URI) described in the Semantic Web literature [1]. Information contained in a URI cannot be authoritatively replicated in a different location because the identity *is* the location. For the same reason, you cannot publish data to the World Wide Web without declaring where you want to store it, which prevents the system from taking advantage of the ever-growing number of mobile devices that (in the current paradigm) are often consumers but rarely providers of information.

Scalability This is a *sine qua non* of any universal system, and is closely related to Liquidity. As well as being architecturally scalable in the large, the representation of a single object must itself be scalable, without conflicting with the requirement for extensibility. This is a significant problem for XML, where it is all too easy for greater richness of information to impede its dissemination. For example, in the MUCHMORE project (a significant interdisciplinary effort to use Multilingual Concept Hierarchies for Medical Organisation and Retrieval, see Volk *et al.* [2]), a large knowledge base was used to annotate documents with syntactic information, phrasal chunks, medical terms from the UMLS interlingua,[1] and semantic relations between these medical terms. Such a system contains enough information in XML for the system to behave intelligently in a very challenging domain, but in the process, the length of each document in the corpus is multiplied several-fold, imposing a bottleneck for many devices with less bandwidth whose users do not need to access all of this information.

This is a widely recognized limitation of XML and other linear text-based formats, which were originally conceived as serialization formats rather than implementations. The annotation necessary to put a even small amount of RDF/XML markup on a page often dramatically increases its size [1, Example 3].

Universal Identity and Reference One important way to improve scalability is to refer to information by using relations or pointers. For this to work, it is vital for one information object to be able to refer to others. This is the mechanism by which knowledge bases such as Cyc [3] and WordNet [4] make the transition from being ad hoc lists of information to being semantic networks.

This is the lifeblood of relational databases, though in relational tables identity is not universal, but scoped only to a particular table. For one data object to cite another, it must be able to refer directly to a table and penetrate its local namespace. To solve this problem in the large, we need a single universal identity, not for each data-table, nor for each object within a data-table, but for each information object to which another object might refer. The URL/URI system tries to solve this problem, though as described above, the binding of the identity to a particular device restricts data liquidity.

Widespread Collaborative Construction The ease of publication to the World Wide Web has enabled millions of authors to write documents in electronic form, and include some notion of universal identity and reference. The problem with the Web as a knowledge representation system is of course its lack

[1] Unified Medical Language System, `http://www.nlm.nih.gov/research/umls/`

of structure other than links and formatting guidelines, which is one of the reasons that the step to the Semantic Web requires such an enormous annotation effort [5]. The Web stands in great contrast to most "knowledge representations." To produce a universal database of world knowledge, millions of users will need to collaborate, but this is very difficult to orchestrate while ensuring that some vital datasets have reliable and authoritative structure and content.

Layered Semantics One way (and as far as we can see, the only way) to enable authoritative knowledge representation to grow from any system with widespread contributions from voluntary authors is to ensure a careful system of *semantic layering*. The information architecture should allow all sorts of information to be entered, in such a way that it is automatically possible for any device in the system to replicate and refer to this information. Complex metadata requirements should not stand in the way of publication: instead, the adoption of metadata standards should be encouraged by the availability of indexing and visualization tools that make the creation of well-structured data worthwhile.

3 The U-form: A Universal Data Container

By studying the strengths and weaknesses of systems including those described in the previous section, we have designed and widely implemented an architecture for knowledge representation. This combines the important considerations outlined in the previous section, as far as this is possible. The basic abstraction of the system is an abstract data type called the *u-form*. A u-form is simply a bundle of name-value pairs associated with a universally-unique identifier (UUID). A u-form may be conceptualized as follows:

$< UUID >$:

`attribute_name_1`	value 1
`attribute_name_2`	value 2
...	...
`attribute_name_n`	value n

U-forms have the following properties:

- *UUID* is a sequence of bytes that, within acceptable engineering tolerances, is assumed to be unique in the Universe.
- In addition to the UUID, a u-form comprises a set of attribute name/value pairs.
- Each attribute name/value pair forms a 2-tuple comprising a single attribute name and a single value.
- Each attribute name is a text string of arbitrary length. It must be unique within the u-form.
- Each value is a sequence of bytes of arbitrary length.

$\sim$ 013a64ab30588c11d6b09a4cf30b756185

`name`	Pittsburgh
`country`	US
`state`	Pennsylvania
`latitude`	40.44
`longitude`	-79.996

Table 1. Simple u-form representing the city of Pittsburgh, showing the UUID and a few attributes and values.

The above is the entirety of the definition of the u-form datatype. A basic example of u-form may be the representation of the city of Pittsburgh given in Table 1.

The idea of collecting sets of attributes and values into distinct bundles representing particular objects is due to Dertouzos [6], who proposed this architecture for an "electronic form" or *e-form* that computers would use as a universal means of communication. Our only innovation on this structure is the addition of the UUID, which supports the need for persistent identity and reference of information objects. Adding this unique reference to an e-form gave the name *u-form*.

A u-form in our architecture a generic data container. That is, it is the simplest unit of information that can be modified or extended without changing its identity. In this regard, it plays a role similar to that of a variable in traditional programming languages or of an object in object-oriented programming. Unlike these, however, the u-form is a purely declarative data type: it exists entirely in information space. Variables and objects, on the other hand, are not products of information architecture, but rather of system architecture. That is, their fundamental nature is defined in terms of some specific programming environment. An object may be serialized (i.e., described as data) into some external format such as XML. But such a serialization is not itself an object, merely a description of one. Objects as such can exist only in the context of a running computer program. U-forms, on the other hand, are (like integers, XML, or relational tables) essentially data. They are message, not medium. Unlike integers or XML, however, each u-form has an intrinsic identity that remains the same even if the contents of the u-form are modified.

It is because of this invariance of identity that u-forms are able to play the role of generic data containers. This enables a u-form to potentially make the journey from being an initial ad hoc data import to being part of an authoritative knowledge base with accepted metadata standards. The cost of entry to the universal database is thus made very low, but this places no ceiling on the eventual quality of the knowledge representation.

3.1 Managing the UUID space

Users and readers who are new to u-forms are often perturbed by a problem that is actually very easy to solve: how do you create a globally unique namespace? Aren't UUID's very hard to manage, and isn't the system going to run out of UUID's at some point? New users are often loathe to experiment with data-structures that use a lot of u-forms, under the impression that UUID's are scarce, and that the more u-forms there are, the harder it will be to guarantee that UUID's are never accidentally duplicated.

We shall briefly consider this problem from first principles, and attempt to demonstrate that the supposed difficulties are of a historical rather than a theoretical nature, stemming from a confusion between the abstract idea of a global namespace, and the physical devices that have been created over the years to operate in this namespace.

The most certain way to create an unambiguous identifier for every distinct object in any universe of discourse is to count the objects, and assign each object a number. Clearly, nobody believes that we are ever likely to run out of numbers, so any problems with using numbers as a namespace must be practical rather than abstract..

From the earliest historical records (e.g. cuneiform inscriptions, [7, Ch. 3]), it is clear that civil servants and record keepers have used numbers not just as objects for measuring and comparing sizes, but also for identifying concepts. This is perhaps most obvious with the sophisticated calendaring systems of civilizations such as the Egyptians and the Mayans, who recognized early on the benefits to historiography of having a single namespace for dating events, rather than a namespace specific to the reign of a particular ruler. It has taken many millenia for the principle of a global namespace for denoting instants of time to be adopted, though this namespace is by now largely agreed upon and implemented (see e.g. [8]).

Other examples of the historic use of numbers for both counting, measuring magnitudes, and providing an unambiguous namespace, are described by [9, Ch 1]. One of the simplest things to realize is that numbers have always been used very ambiguously, in the sense that they mean different things in different contexts. "Table 12" in one restaurant is always a different table from "Table 12" in any other, and "House number 9" on one street is always different from "House number 9" on any other. This ambiguity is licensed by a conventional agreement between language users, fixing the scope of context within which the number is a unique identifier.

As long-distance communication has become possible, such simple local conventions have been supplemented by attempts to secure interoperability. One way to do this is by prefixing numbers in a namespace with a unique number to refer to that whole namespace. This process can be repeated indefinitely, one of the most everyday examples being the way a long-distance code is prefixed to a local telephone number to make it national, and an international code is prefixed to this local number to make it worldwide. Such a method constructs a hierarchy of namespaces, in which the potential ambiguity of any number can

be resolved by considering its chain of inheritance in the hierarchy. This process also occurs in language, without the use of numbers (for example, a speaker could easily say "Toledo, Ohio" to distinguish this city from "Toledo, Spain" or any other city with the name "Toledo"). Such a hierarchy can also be extended downward to more specific cases. For example, once a company owns a specific telephone number, they can make that the number of a central switchboard and issue extensions that now uniquely refer to telephones on individual desks.

It is possible, in theory, to create a global namespace using such a combination of local uniqueness and a tree of identities, so that any agent that understands the path down through the tree can understand the identity of the information objects at the leaves of the tree. We will call this management strategy the *hierarchical approach*. The most significant implementation of the hierarchical approach (for the purposes of this paper) is the Universal Resource Name[2] framework designed by the IETF (Internet Engineering Task Force).

It is instructive to consider some of the assumptions underlying such a hierarchical approach to managing a global namespace. The IETF document outlining best practice for assigning URN's to information resources [10][3] states the following assumptions:

- Assumption #1: Assignment of a URN is a managed process.
  I.e., not all strings that conform to URN syntax are necessarily valid URNs. A URN is assigned according to the rules of a particular namespace (in terms of syntax, semantics, and process).
- Assumption #2: The space of URN namespaces is managed.
  I.e., not all syntactically correct URN namespaces (per the URN syntax definition) are valid URN namespaces. A URN namespace must have a recognized definition in order to be valid.

This places stringent restrictions on obtaining a universal identifier within the URN framework: not only syntactic requirements, but procedural and semantic conditions must be met. For certain namespaces, the authority conferred by such a process is a necessary part of maintaining standards, and for this reason, the URN framework is an ideal root pathway for subsuming mature standards such as the ISSN (International Standard Serial Number) and ISBN (International Standard Book Number) systems.

However, providing a universal namespace for objects that are already approved, edited, and published does not confer the full benefits of a genuinely universal database. Producing a final, bound copy of a book is a long process that requires all sorts of information management along the way: developing many versions of files, including some files in others, merging edits of the same file carried out on different machines. This process would be greatly aided by a universal identity for information objects: there is no engineering reason for reserving the idea of universal identity for the finished book alone. Ideally, we

[2] `http://www.ietf.org/html.charters/urn-charter.html`
[3] `http://www.ietf.org/rfc/rfc2611.txt`

want to build a system that gives permanent identity to information *as it is created*. The challenges involved in developing an information space for published books and manufactured goods is largely solved, but the challenges involved in creating an information architecture for users of pervasive computing technology are not.

Suppose, for example, that I wish to create a personal address book application. Many such applications have been built, and all modern mobile telephones incorporate such software. However, many users will have had the experience of getting a new mobile telephone and manually transferring their address book from the old device to the new one. This is mitigated by having devices that can easily be synchronized with personal computers, but this still involves some sophistication and diligence on the part of the user. It still leaves your information locked in a few devices, which is an improvement on just the single device, but still falls way short of the goal of having your information available wherever you need it.

The best solution to this problem is to give the entries in your address universal identities, and to build information devices that use this identity namespace to ensure interoperation (as advocated by [11]). For this to work, the publication bar to the universal namespace must be kept as low as possible, while preserving universal identity. This is where automatically generated UUID's come into their own.

One of the most successful schemes for generating such UUID's on demand is to concatenate the the current time (defined by a local system clock) with the unique network address of the device generating the UUID. This system has been used successfully by the Open Software Foundation's (OSF) Distributed Computing Environment, and later to give universal identity to COM object on Microsoft Windows platforms. Such UUID's are typically 16 bytes (128 bits) long, enough to ensure that a machine can generate some 10 million UUID's per second, with time uniqueness guaranteed until 3400AD! It is important to remember that this is only one working implementation: there will be no "year 3400 problem," we may simply need to add a new byte to UUID's at that time. In the meantime, we should be sure never to build systems that can only cope with UUID's of a fixed length, just as we do not build web browsers that can only cope with URL's of a fixed length.

Thanks to a recent proposal by Leach, Mealling and Salz [12], it is likely that the UUID namespace generated in this fashion will become a regular part of the URN namespace.[4] It is thus possible that the URN and UUID namespaces will become combined, a step to enfranchise all information types that we would applaud.

For these reasons, we believe that a universal namespace that includes the best of hierarchical organization and generation-on-demand is not only possible, but already a reality. UUID's in this namespace are not scarce, they are plentiful: it is much easier to create a UUID for every e-mail message (and even every toothbrush) than it is to create the e-mail messages (or the toothbrushes

[4] `http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt`

in the first place). The key difference between such a namespace and the current URL/URI identifiers is that a UUID refers not to a *location* but to pure *information*.

4 The Network of U-forms

Universally identified information provides fresh opportunities and challenges for building information systems. This section briefly describes some of these topics, including relations and indexing in the abstract information space, and repositories and shepherding in the physical device space.

4.1 Relations

Relations between u-forms are indispensable to the practical application of the system. Simply, a relation is a u-form attribute value consisting of a sequence of UUIDs.[5] The relation is the basic mechanism by which u-forms may refer to other u-forms. The notion of a relation is very similar to that of a pointer in standard programming languages, or a URI in the Semantic Web [1]. However, rather than depending on the address of the storage location as the means of reference, relations depend on UUIDs, and are thus completely location-independent. Relations can refer to u-forms whose locations are unknown, provided there is a suitable architecture for requesting and retrieving them.

For example, in the Pittsburgh u-form in Table 1, the `country` and `state` attributes would not really be strings, they would be relations to other u-forms representing the USA and the State of Pennsylvania respectively. A human-readable encoding of the Pittsburgh u-form would look very much the same, but instead of displaying just the `country` and `state` attributes, it would display the `name` attributes of the targets of these relations. This technique is of course widely used: for example, in WordNet, relationships are stored not between the surface-forms of words, but between synsets that are represented internally by unique identifiers or *offsets* [4]. The difference between the systems is that UUIDs are not defined within any one dataset: they are universal.

Many different relations can be collected in a single u-form, and even a single attribute of a u-form. For example, some u-forms are *collections*, and have a `members` attribute whose value is a list of other UUIDs. In this way, the mechanics of Boolean set theory is easily implemented.

The universal identity of a u-form therefore enables the database to express all kinds of linked structures, from simple directed graphs, to taxonomies, to full semantic networks. By judiciously combining numerical values, string values, and relations, many formal structures such as vectors, lattices, graphs, and curvilinear coordinates can easily be expressed in u-forms. This enables complex models of real-world knowledge to be expressed and combined or "fused." For example,

[5] That an attribute of one concept may contain a relation to another concept is noted explicitly by Aristotle, *Categories*, Ch. 7.

we have already combined the shapes of countries and their regions, extracted their common coastlines to fuse with other geophysical features, and expressed a detailed lattice of geopolitical features that extends from the country level right down to individual municipalities, census blocks and even individual parcels of land (for regions where this data is available). At this detailed level, parcels of land can be cross-referenced to public transit information, also expressed in u-forms, enabling a reasoning agent to analyze the data and work out the simplest way for a human to travel from (for example) their home to a doctor's appointment. The common currency of u-forms enables the reasoning agents performing these tasks to follow design patterns that are extremely simple, compared with most traditional artificial reasoning agents.

4.2 Distributed Indexing

A more advanced use of relations and collections is to provide distributed indexes of u-forms. The details of our research in this area are complex and will be described in a separate paper, but the broad overview is as follows.

Since all information is by nature distributed and universally identified, there is no need for a central server or network to index all u-forms and serve all queries, nor is this desirable, because of the inevitable time-lag that this involves. The large internet search engines such as Google[6] and Yahoo[7] demonstrate this shortcoming — they are excellent data-silos for persistently located information on the World Wide Web, but as tools for enabling person-to-person information sharing and collaboration between small local devices, such indexes are far too slow to update.

Instead, since all information in our system is contained in distributed u-forms, we have built indexing structures are distributed in exactly the same fasion. On a simple level, a collection u-form behaves as a degenerate index: for example, I could keep a collection of the UUID's of all the u-forms of people in my address book, and then this collection will be as persistently available as the address-containing u-forms themselves.

To search for a particular person by name, it is possible to iterate through this collection linearly until the correct name is found, but obviously this process cannot be used for large datasets. In the case of large datasets, to build an index of (for example) the name attribute from all the u-forms in a dataset, we adopt structures similar to the indexes provided to large books such as the *Encyclopedia Britannica*. In other words, the top level u-forms in the index might consist of a collection of UUID's along with annotation to say which segment of the alphabet they provide an index to. These u-forms may recursively refer to other u-forms that split the alphabet into more specific segments, and so on, until the desired name is found. Many such search trees are a standard part of computer science (see for example [13]). The research challenge lies in working out which of these structures can be adapted properly to environments where

[6] `www.google.com`
[7] `search.yahoo.com`

the information objects are distributed, and hence may or may not be present in your current venue.

Indexing of this sort can also be used to create indexes whose keys are UUID's, as well as just keyword strings. This is crucially important to distributing not only information but authority. For example, suppose that a user wishes to place a comment on the u-form for Pittsburgh in Table 1, to say that it is a beautiful (or an ugly) city. Such user opinions should not affect the u-form for the city itself, however, since this is simply meant to contain factual information about Pittsburgh. The solution is that the user's comments should be placed in separate u-forms that contain relations to the object that the comment is intended to describe. Such comments can be found by means of a UUID index. A particular user or group of users can even create a special index of "everything said by this community about other u-forms," thus forming a kind of "content channel."

In this way, universal identity enables information to be distributed without losing its identity. It is not only the storage and the computation that is distributed, it is also the authorship and authority.

4.3 Repository Venues, Replication and Shepherding

The universal database system has already been successfully deployed on behalf of government, commercial, and non-profit organizations. The main underlying component is a scalable persistent data store, existing both as an embeddable component and as a network server, whose essential role is the storage, replication, and retrieval of u-forms. The external interface to such a repository server is extremely simple. The basic operations consist of the following:

- GET ATTRIBUTE — Given a UUID and an attribute name, the repository returns the previously stored value of that attribute.
- SET ATTRIBUTE — Given a UUID, an attribute name and a value, the given attribute is set to the given value.
- LIST ATTRIBUTES — Given a UUID, the repository returns a list of attributes that have values in the specified u-form.

Since the identity of the u-form (its UUID) refers directly to the information object, not to its physical location, the u-form abstraction enables copies of u-forms to exist simultaneously in many different repository venues. The act of creating a new instance of a u-form in a different venue is called *replication*. Replication is a great bonus for data liquidity, because it enables frequently requested u-forms to exist in many locations near to where they are needed, avoiding delays and heavy network traffic when making GET ATTRIBUTE requests.

Of course, just because two replicates are defined to be the same does not mean that they are the same. Indeed, it is impossible, even in principle, to guarantee in general that two mutable data objects are identical at any given time. The task of maintaining both the pervasive availability of u-forms, and

the constraint that any edit to a u-form should be replicated to every version of that u-form as quickly as possible, is carried out by automatic agents called *shepherds*. The shepherd's job is to implement various business rules concerning the appropriate replication policies to be applied to u-forms whose contents are inconsistent across the venues. Currently the shepherding agents have some degree of artificial intelligence — for example, they can detect and, in limited situations, resolve conflicts that occur when two versions of the same u-form contain different information.

The dual recognition that (a) one can use replication to approximate the ideal of having the same data simultaneously accessible in multiple venues simultaneously but that, (b) any such approximation will inevitably be imperfect, proves to be provocative: How can we make this approximation useful? What exactly is it an approximation of? What compromises will nature force on us, and how can we mitigate them?

In exploring these questions, we have built peer-to-peer networks in which all traffic between venues is handled solely by shepherds. The goal of the shepherds is to make the repository network converge as closely as possible to something we call the *GRIS Principle*, which is the assumption that a replicated u-form is the same in every venue. GRIS is a (facetiously named) project acronym for the "Grand Repository In the Sky," a persistent store of unbounded capacity that is capable of providing instantaneous, transactionally consistent read/write data access anywhere in the world. Though these capabilities are impossible to ensure in practice, the question "How closely are we approximating GRIS?" has proved its worth as a benchmark for comparing implementations.

The current repository network has been used in a number of contexts and is robust both from a stability and a scalability perspective. Detailed performance studies have been conducted and are available from the authors. There are many other challenges in approaching the GRIS ideal, many having to do with such services as efficient search, retrieving partial values of large attributes, and authentication/security. This material has been the topic of much of our recent research and will be reported in a separate paper.

5 Roles

We have already seen that, in contrast to relational tables, a u-form does not have to follow any predefined schema when it is created, just as an XML document does not have to have a predefined set of permissible tags. New attributes can be added at will (this practically defines what it means for a data-object to be extensible).

However, it is extremely important for attribute names to be chosen so that u-forms can be interpreted according to the intentions of their creator. For example, the Pittsburgh u-form in Table 1 uses the attributes `latitude` and `longitude` which our mapping applications recognize these names as global coordinates. If the creator of the u-form uses `lat` and `long` instead, a rendering agent that is looking for `latitude` and `longitude` will not work.

There are at least two ways we could handle this situation:

- Introduce a definitive list of universal attribute names.
- Enable information about intended interpretation to be carried by individual u-forms.

The first of these options, though simple and tempting in the short run, is not future-proof and would make the management of the attribute space far too critical and cumbersome.

Instead, we take the second option, encouraging the creation and use of metadata-carrying u-forms called *roles*. A role is simply a u-form that asserts that, if a particular attribute is present, the creator of the u-form intends that the value of this attribute should be used in a particular way. To use a particular role, a user simply adds a relation to this role's u-form in its *roles* attribute.

For example, the real Pittsburgh u-form includes a relation to the *Geopolitical Entity Role* in its list of roles. This role asserts that the value of the `name` attribute should be a human-readable string, the value of the `country` attribute should be a relation to the u-form representing the country in which the city is located, and that the `latitude` and `longitude` attributes should be interpreted as curvilinear coordinates describing a location on the planet Earth. Other attributes in this role include the `fips_code` issued by the US Federal Information Processing System.

Latitude and longitude are in fact covered by a more general schema, the *Role for Geo-Reference using Global Coordinate System*, and the *Geopolitical Entity Role* inherits all of the attributes covered by this role. In this way, one role can extend the namespace defined by another. Several other roles (e.g., those used to describe natural rather than man-made geographic phenomena) also inherit from the *Role for Geo-Reference using Global Coordinate System*, and a mapping application that only knows about this general role can still correctly render a u-form that plays any of these roles. At the top of this role-structure is a role for *Entity*, which just describes the attribute's `name`, `label`, and `description` (`label` is used by interface devices with limited space — for example, the `label` attribute of the u-form for Pennsylvania contains the abbreviation *PA*). All u-forms that represent some object in the real world can have a meaningful `name` and `description`, and thus every role played by real-world objects will almost certainly inherit from the role *Entity*. (Inheritance is transitive, so this is not necessarily a direct inheritance.)

5.1 Types of Roles

Most of the roles that are in wide current use can be grouped into three broad categories:

- **Phenomenal** roles usually list the attributes used by u-forms representing objects in the real world. At some level of inheritance, phenomenal roles generally inherit from the role for *Entity*.

- **Adjectival** roles list further attributes that may extensibly be applied to phenomenal u-forms. For example, many different kinds of objects have an *address* or *price*, and these can be added to phenomenal u-forms without having to add these attributes to each phenomenal role.
- **Assertional** roles list relationships between several phenomenal u-forms. For example, a u-form playing the role *Service Offering* asserts that an organization is providing a service (a kind of activity) at a given venue between certain hours.

There is an informal correspondence between phenomenal, adjectival, and assertional roles and between nouns, adjectives, and verbs in natural language. (Like parts of speech, the boundaries between the role categories are blurred in a few instances.) The system of roles is partly influenced by the idea of case roles and case grammar, developed by Fillmore [14] as a semantically robust alternative to the idea that syntactic parsing must precede semantic interpretation. In the same way, a u-form may often omit some of the attributes defined by its roles, and an artificial agent will still be able to proceed intelligently with partial information. For example, a geo-political entity with no `country` attribute can still be entered into a spatial index and rendered on a map using just its `latitude` and `longitude`.

5.2 The Difference Between Roles and Ontologies

Because of the structure of inheritance and the human-readable choices of role and attribute names, it is tempting to think of the role system as a kind of ontology. It is important to remember that this is not the case. All roles do is ensure that the creator of a u-form and its eventual user have an agreed vocabulary for attribute names. This enables roles to be reused without making ontological claims, but merely to facilitate interpretation.

This lack of ontological claim has an important corollary: a u-form may often have more than one schema, which is accomplished by relating the u-form to more than one role. Although these roles may not syntactically contradict each other by assigning conflicting meanings to the same attribute name, the various roles that a u-form plays may be quite distinct. These roles may complement or elaborate upon each other (such as adding an annotation role to a u-form representing a text document); they may be independent of each other; or they may even semantically contradict each other. The ability to add unanticipated roles to existing u-forms is a prime source of extensibility and introspection within the architecture. It becomes possible to write applications that can process data objects that they only partially understand, even to the point of modifying those objects without risk of compromising other applications' continued ability to interpret and modify other attributes of the u-form. The ultimate effect of this is that roles can be applied and even defined at retrieval time, rather than when the database is created. The importance of this fact in enabling the creation of evolvable systems is difficult to overstate. U-forms can systematically take

on new roles, and therefore constitute possibly the first ever knowledge representation system to embrace the systematic ambiguity that is now known to be responsible for the reuse of word-forms in natural language, as described by Generative Lexicon theory [15].

Finally, it should be noted that the definition and use of u-forms are not necessarily dependent upon the use of roles, just as it is not necessarily dependent on our implementation using repositories and shepherding. While roles have provided invaluable metadata for interpreting u-forms, they are at a higher semantic layer than the construction of the u-form datatype. Because of this semantic layering, it would be quite possible for a new user to enter the Information Commons and use a completely different system for negotiating metadata, and because of the careful semantic layering of the knowledge representation system, all of the data we have already imported and fused could still be of great use to them. The persistent identity conveyed by the UUID will still enable all of the work in creating u-forms to be highly valuable even if grave flaws are later discovered in the metadata schema. We believe that this evolvable design signals an important advance over the traditional uses of both relational tables and ontologies.

6 Applications and Results

For such a large knowledge-engineering undertaking as this, the test of success lies not only in its theoretical soundness, but in its usefulness in practice. To this end, we have imported many different datasets to serve the needs of several projects. These include geographical, demographic, environmental, and literary information sources, and the current network of repositories (including for profit, non-profit and educational users) contains over 300 million u-forms. Millions of these u-forms are fused together in a coherent model of the real world in information space. The value of having a common currency for information, and common metadata to guide its interpretation, becomes much clearer when dealing with such large quantities of data: for example, the same mapping tools can be used to zoom in on and visualize far more types of data than is possible with conventional GIS systems. This world-modeling effort will be described in full in another paper: here we will describe a regional collaborative project that gives a snapshot of the power of the universal database concept.

6.1 Allegheny County Human Services Project

Allegheny County in Western Pennsylvania (which includes the Pittsburgh metropolitan area) has thousands of providers of human services, ranging from adoption counseling to rehabilitation clinics to Boy Scout troops. Two major organizations, the Allegheny County Department of Human Services (DHS)[8] and United Way of Allegheny County (UWAC),[9] both maintain large relational

[8] `http://www.county.allegheny.pa.us/dhs/`
[9] `http://www.uwac.org/`

databases of organizations, facilities, and services provided in the region. In order to integrate these previously incompatible information sources, a project was undertaken by Three Rivers Connect (3RC),[10] a Pittsburgh non-profit foundation, to represent these datasets using u-forms in the Information Commons.

Initially, there was considerable skepticism from the two main data providers, both of whom felt that their data could only suffer from being included in any other information system, since their databases were carefully designed to contain information specific to their needs. This is a very typical stance, since many data providers have invested much time and expertise in their systems, and have good reason to fear tampering. The project gradually calmed these fears and produced a common solution, taking the following steps:

- General roles for *Facility*, *Organization* and *Service Offering* were defined. Between these roles and other linked concepts such as *Activity* and *Municipality*, most of the important data from both databases could be faithfully represented.
- Using the extensibility of u-forms and the inheritance structure between roles, the remaining attributes (such as DHS- and UWAC- specific codes for different services and organizations) could be easily accommodated by introducing more specific roles that inherit from the basic *Service Offering* and *Organization* roles.
- The common architecture now enabled data fusion of items that were in both databases — for example, two facilities with the same name and very similar latitudes and longitudes would be amalgamated into the same u-form.
- The contributors began to see unexpected advantages from this process — for example, they could be alerted when the *telephone* attribute of an organization differed from the version in their database, and they began to request lists of these differences so they could update their original data where necessary.

These points exhibit many of the benefits of the universal database concept. Because there is a single extensible u-form representing a real-world object, information accrues to this u-form, enabling all users to see the most definitive, up-to-date information. Similarly, the system of roles encourages attribute names to be chosen from a common, but evolving, vocabulary that can expand at will to include more specific attributes from individual datasets.

Interfaces were built that enable a health professional to explore the datasets, navigating by geographic region, names of programs, organizations, and facilities, and via the United Way taxonomy of service offerings. This information can then be viewed in a number of ways that emphasize different aspects of the data, ranging from an in-depth analysis of which services are provided by which organizations at specific facilities (see Figure 1), to a map view that is fused with public transportation data so that the person being referred can easily plan a trip to the facility in question. This interface is currently available to regional

[10] `http://www.3rc.org/`

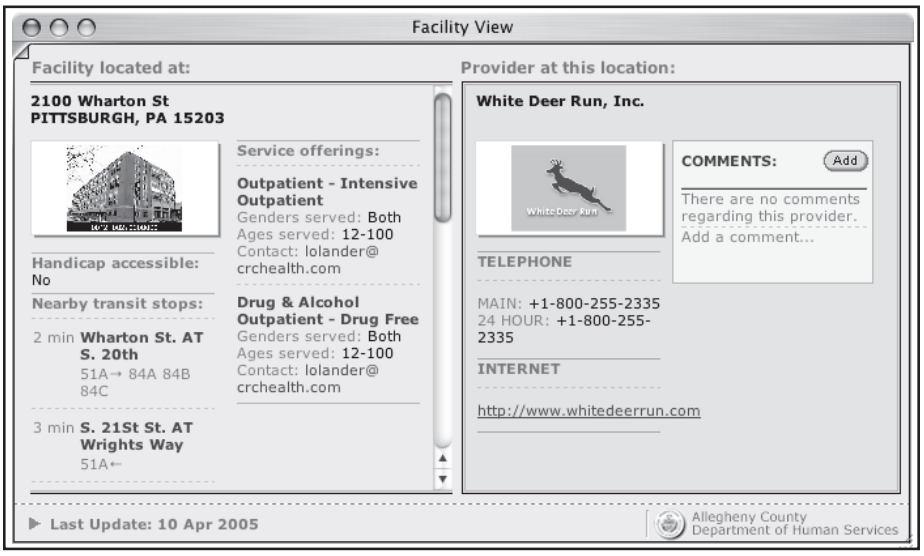

Fig. 1. A view of a u-form mediated by the *facility* role. The images and text in this interface are all attributes of u-forms; the existence of an information-centric interface encourages users of the system to reuse the same roles and attribute names, so that their information will appear in a predictable part of the encoder.

health professionals, and will shortly be released to the public over the Internet. A demonstration will be given in our presentation.

7 Conclusion

We have described a universal data architecture based upon the concept of a u-form, a data container whose contents corresponds to a real-world object rather than an architecturally confined data-table. The persistent universal identity of this data-object enables other u-forms to unambiguously refer to this object from any location, so that data can be structured into semantic networks and fused into definitive versions using the role system to describe extensible meta-data schema. The information architecture is supported by a mature peer-to-peer network of repositories, and has been used in large-scale projects. We have demonstrated that the data architecture can satisfy the scalable needs of knowledge representation, from simple (though pervasive) hypertext applications to a complex domain requiring the fusion of demographic, geographic, transportation and health information.

References

1. Minola, F., Miller, E.: RDF primer. (2004)

2. Volk, M., Ripplinger, B., Vintar, v., Buitelaar, P., Raileanu, D., Sacaleanu, B.: Semantic annotation for concept-based cross-language medical information retrieval. International Journal of Medical Informatics (2002)
3. Lenat, D.B., Guha, R.V.: Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project. Addison-Wesley (1990)
4. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge MA (1998)
5. Handschuh, S., Staab, S., eds.: Annotation for the Semantic Web. Ios Pr Inc (2003)
6. Dertouzos, M.: The Unfinished Revolution. Harper Collins (2001)
7. Boyer, C.B., Merzbach, U.C.: A History of Mathematics. 2nd edn. Wiley (1991)
8. Wolf, M., Wicksteed, C.: Date and time format. Technical report, World Wide Web Consortium (1997)
9. Widdows, D.: Geometry and Meaning. CSLI publications, Stanford, California (2004)
10. Daigle, L., van Gulik, D., Ianella, R., Faltstrom, P.: Urn namespace definition mechanisms. Technical report, The Internet Society (1999)
11. Lucas, P.: Mobile devices and mobile data: Issues of identity and reference. Human Computer Interaction **16** (2001) 323–336
12. Leach, P., Mealling, M., R.Salz: A UUID URN namespace. Technical report, The Internet Society (2004) Current draft, awaiting approval.
13. Knuth, D.: The Art of Computer Programming. Volume III, Sorting and Searching. Addison-Wesley (1973)
14. Fillmore, C.J.: The Case for Case. Eric Document Service (1968)
15. Pustejovsky, J.: The Generative Lexicon. MIT press, Cambridge, MA (1995)