DOCUMENT NUMBER (USE DOCUMENT NUMBER STYLE)
MITRE TECHNICAL REPORT (USE DOCUMENT NUMBER STYLE)

MITRE

Developing a Common Interchange Model and Format for Representing Knowledge Synthesized from HLT Analytic Results

Ransom Winder, Joseph Jubinski, Michael Smith
{rwinder,jjubinski,smithmj}@mitre.org

Approved for Public Release; Distribution Unlimited. Case Number 14-3980

This technical data was produced for the U.S. Government under Contract No. W15P7T-13-C-A802, and is subject to the Rights in Technical Data-Noncommerical Items clause at DFARS 252.227-7013 (FEB 2012)

McLean, VA

Abstract

In the Human Language Technology (HLT) domain, analytic results extracted from raw document sources are captured in varied models and formats due to the depth of what can be revealed and the diversity of interpretation. However, some common model and format must be followed to allow for multiple analytics to operate together in workflows and enable both the communication between analytics and the fusion of parallel or complementary results. This data integration problem is exacerbated when placing an emphasis on extracting knowledge from text, as the data model must be both adaptable and extensible to handle current and emerging content extraction capabilities and technologies. This paper describes a common interchange format and model designed to coordinate the extracted information from raw document sources in order to generate knowledge. The approach described adheres to the principles of adaptability and extensibility. It also provides the means to represent the annotation data that act as the reference for the knowledge and maintain provenance about these analytic results. While the data model and format described were designed for the HLT domain, the process used to develop them can be applied to other domains as well (e.g., image processing, signal processing).

Keywords: Human Language Technology, Ontology, Content Extraction

Table of Contents

1 Introduction 1
2 Background 2
3 Methods 5
3.1 Design Philosophy 5
3.2 Model Layers: Knowledge, Annotation, and Provenance 6
3.3 Common Format: Statement Basics 6
3.3.1 Classes – Entities, Designators, and Time 7
3.3.2 ObjectProperty – Relationships 7
3.3.3 Frame – Events 8
3.3.4 Designator Example – Phone and Email 8
3.3.5 Custom Types 9
3.3.6 Annotation Layer – Mentions 9
3.3.7 Document Seeds and Connections 11
3.3.8 Provenance 11
3.3.9 Visualizing the Common Interchange Format 13
3.4 Common Interchange Model 15
3.4.1 Coarse Divisions: Graph, ReferenceElement, and KnowledgeElement 16
3.4.1.1 Graph and Subclasses 16
3.4.1.2 ReferenceElement and Subclasses 16
3.4.1.3 KnowledgeElement and Subclasses 17
3.4.2 Object Properties 19
3.4.2.1 hasAnnotation 19
3.4.2.2 hasEntityRelationship 20
3.4.2.3 hasEntityProperty 20
3.4.2.4 hasEntityDocumentRelationship 20
3.4.2.5 hasDesignatorRelationship 21
3.4.2.6 hasTimestamp 21
3.4.2.7 hasTableObject 21
3.4.2.8 hasFrameRole 21
3.4.3 Data Properties 21
4 Discussion 23
5 References 25

Appendix A 27

This page intentionally left blank.

1 Introduction

The Human Language Technology (HLT) domain encompasses a wide variety of analytic techniques that can be applied across raw document sources to extract or identify content artifacts or provide analytic results. While the results of the implementations of these techniques are manifold and useful, they nevertheless exhibit some significant limitations. Due to the complexity and scope of HLT, there is no universal model and representation for extracted content. Further, there are many extant taxonomies for analytic results that have different definitions for their expected content, and they do not easily align. Nonetheless, these taxonomies represent a necessary starting place for building a knowledge model that can be lifted from extracted content and associated annotations. This initial heterogeneity presents a significant technical challenge.

In complex HLT systems there is often a need to synthesize results from multiple discrete analytics arranged in workflows addressing higher-order problems. A Common Interchange Model (CIM) can act as a lingua franca between different analytics, aiding integration of analytic results expressed in varied models and formats. Having a CIM allows analytic results to be mapped consistently and also gives analytics relying on minimally processed data the capability to reconstitute their required input in a lossless fashion from the CIM. Given the universality that a CIM implies, a natural question arises as to whether or not analytic results can be elevated into direct assertions of knowledge for a world model of interest.

This paper describes a model and format that addresses this question for an HLT architecture designed to populate knowledge bases with facts extracted from the content of raw documents (i.e., largely unstructured text). In addition to generating knowledge from these analytic results, it is crucial to preserve the original annotation and provenance regarding the analytic processes in order to provide traceability and confidence in the system results, while maintaining a clear definition of what constitutes knowledge co-referenced across textual mentions and elevated from document-centric annotation. This ensures knowledge can be isolated from the annotation for downstream processes that operate only on assertions of knowledge outside of the context of specific mentions. A further design constraint levied on this architecture and the CIM is that it be created with free and open-source elements to make it easily reproducible and low-cost.

Described below is the relevant background, including both a survey of potential and related data models and an overview of the analytic architecture supported by the proposed CIM and its companion Common Interchange Format (CIF). Then the development, structure, and implementation decisions with respect to the common model are presented. Finally, there is a discussion of the future planned work.

2 Background

The MOSAIC architecture, described in greater detail in (Winder et al., 2011; Winder et al., 2013), is founded on a philosophy of adaptable and extensible design that is interoperable with new and existing analytics. It provides a framework that loosely couples these discrete analytics and organizes them into workflows so they can function in concert and have a common store for their data. MOSAIC is application-agnostic and its subcomponents (an executive, a data bus, the analytic layer) can be substituted, exchanged, or modified as necessary. Separation of documents and analytic artifacts from the execution was key to this loosely coupled design. Because it was intended to support a wide variety of analytics, the lack of cohesion with respect to the native analytic data models and formats requires attention. Analytics are often developed independently without the intent to integrate, essentially creating a variety of representations that needs to be resolved before allowing communication between analytics or fusion of their results. Thus, there is an essential need for a common interchange model and format, whether it is derived from an existing format with sufficient model coverage and representational flexibility or is newly generated to serve this purpose.

This naturally leads to an examination of existing formats and data models from which to choose appropriate candidates for the roles of format and model. Given the need for a representation that is compatible with some ultimate knowledge base that can express first-order logic, the most significant option examined in service of both roles was the Web Ontology Language (OWL)[1] family of languages for knowledge representation, which are the current accepted web standard for ontological development. Our attention was primarily given to OWL 2 (Motik et al., 2009). In addition to providing definitions for base level classes, object properties, data properties, and individuals, it is also compatible with other Semantic Web standards, such as XML RDF[2] (Resource Description Framework), which more closely align to what has been termed format here. RDF is founded on a linking structure of statements that represent a relationship between two individuals, where the statements are usually referred to as triples, providing a simple rule for structuring information or knowledge, as appropriate. There are alternatives to these representations such as N3 (Notation 3),[3] which has the advantage of being more concise and human interpretable than XML serializations of RDF. While these formats are typically founded on triple relationships, applying a name to the relationship to create named graphs, or quads, is also used in extensions to the RDF syntax such as TRIX[4] or TRIG[5] to allow for a more convenient and compact reification syntax and a reduction in the number of tuples employed. Other languages suitable for representing the modeling expressiveness desired include Common Logic (CL) and F-Logic (frame logic) (Kifer et al., 1995). As with N3, there is an emphasis with F-Logic of making readable statements with a simple syntax.

One must populate the data model with specific classes and properties reflecting the domain of what must be captured. Specific individuals and assertions needed for common referents by

[1] http://www.w3.org/TR/owl2-xml-serialization/#Example_Ontology_.28Informative.29
[2] http://www.w3.org/RDF/
[3] http://www.w3.org/TeamSubmission/n3/
[4] http://www.w3.org/2004/03/trix/
[5] http://www4.wiwiss.fu-berlin.de/bizer/TriG/

analytics, such as geographic place names or well-known people, may be added as referents to a MOSAIC seed Knowledge Base (KB). The content of the CIM's ontology and accompanying seed KB may be generated newly for each use of MOSAIC, but one can leverage previous efforts to make comprehensive knowledge representations.

Serving as an example of an extensive formal ontology, the Suggested Upper Merged Ontology (SUMO) (Niles & Pease, 2001; Pease et al., 2002) had its genesis in merging together multiple ontologies of concepts that are common across knowledge domains, or upper-level ontologies. It has evolved from this starting state. This breadth of coverage makes it deserving of significant attention when considering that the ultimate content of the knowledge base should not be unduly bound to any particular knowledge domain. FrameNet (Baker et al., 1998) is another example; one more driven from a linguistic background. FrameNet is organized on the idea of semantic frames that express the concepts conveyed by sentences, where frames express particular classes of events and relationships and also capture their particular participants' roles.

Examining models of knowledge that have a basis in what HLT analytics extract from raw documents, one openly available model is the Automatic Content Extraction (ACE) model and its ACE Pilot Format (APF).[6] This offers a representation and taxonomy for various entities, relations, and events considered important and recognizable to automated content extraction systems. The ACE model embodies some SUMO and FrameNet elements, making it feasible to bridge these models' representations. The ACE competitions, which ran for most years between 2003 and 2008, sought to evaluate state-of-the-art systems for content extraction against a common output set (Doddington et al., 2004). The relevance of APF's output format and model to content extraction makes it a convenient candidate upon which to develop a primary knowledge base, especially when it will be populated with knowledge discovered by HLT analytics.

APF includes a method for annotation that points back into the document and identifies from where its entities, relations, and events are derived. It is not the only existing format that attempts to capture HLT annotation. Existing architectures for tightly-integrated components often have their own mandated internal data format. One example is the Unstructured Information Management Architecture (UIMA) that makes use of Common Analysis System (CAS) (Gotz & Suhre, 2004) data structures, which store the subject of analysis (such as a text document), the analytic results of annotation and indices to these results, and a schema for interpreting the structure of analytic results. Another example format is GrAF (Ide & Suderman, 2007), which merges linguistic annotation based on sources such as the Penn TreeBank, PropBank, and others into a single, traversable graph of annotation, and has been demonstrated to be capable of consuming mappings from other formats (Ide & Suderman, 2009). In addition to formats driven by competitions and academia, most automated content extraction systems have their own unique, and often proprietary, formats. It bears stating plainly that neither CAS nor GrAF are domain or ontological models, but rather simple data models that allow for arbitrarily complex structures without any means to express model constraints or axioms. They lack the semantic expressiveness that OWL-based models offer, and while there are semantics, they do not include

[6] An overview of ACE and guidelines for APF: https://www.ldc.upenn.edu/collaborations/past-projects/ace

consistency checkers or reasoners, which can help in making a KB's schema and content interpretable, consistent, and queryable.

Producing an interchange model emphasizing issues in HLT and Natural Language Processing (NLP) remains an active area of research. A recent contribution of note is the NLP Interchange Format (NIF) (Hellman et al., 2013), which addresses challenges in comparing or combining NLP tools not inherently designed to interoperate. NIF is principally defined by a core ontology and supporting ontologies for common NLP terms and concepts, and it is accessible by way of REST services. The result enables integration of NLP applications that are heterogeneous, distributed, and loosely coupled. The emphasis of NIF's core ontology is document annotation and delineation of natural language structures in text, but it can be enhanced with existing knowledge ontologies, such as DBPedia, to expand the robust underlying NLP model. The discussion below explores how NIF and our own model relate.

3 Methods

3.1 Design Philosophy

The philosophy behind the CIM and its expression in CIF is driven by certain fundamental goals for its intended use. Paramount among these is the need to both include and maintain a strict separation between the three interconnected categories of 1) extracted or derived knowledge, 2) original annotation, and 3) provenance regarding the origin of these other two categories with respect to their source documents. The model must be extensible and adaptable as it is unreasonable to expect a static model will encompass everything analytics will identify in documents, especially as new strides are made in analytic development. The redundancy of information in many annotation formats (where extents and types were often expressed more than once) has been minimized here, encouraging conciseness. Additionally, as recipients of the final knowledge need it to be increasingly fine-grained, the importance of the extensibility of the model is once again emphasized. This need for extensibility implies the model should be as simple and regular as possible. A common model must endeavor to achieve uniqueness in its semantics and format, which is a danger as it stands in opposition to the multiple raw HLT analytic formats that may have completely different approaches to representing the same concept, entity type, relationship type, or annotation to the underlying document.

This last point is the most salient when considering selection of an existing model. There is no uniformity among HLT analytic output and a great deal of ambiguity or differences when their models are compared to one another. While rich existing models can form the basis for the common data model, one cannot be chosen and used as is by itself. We must create a model which can span, to the greatest degree possible, the analytics' different representations.

Other crucial considerations were the type of knowledge we wished to accumulate and what knowledge the analytics were able to produce, both of which could evolve as new analytics were developed or new types of knowledge became relevant for the final use cases. Therefore it was important to choose a model that was easily extensible to accommodate new requirements and capabilities, for instance the need to reason over knowledge and axioms. Ultimately, we chose OWL 2 to model the knowledge and annotation schema of CIM.

When selecting a language to express our format, the difficulty in directly appropriating an existing technology is somewhat lessened as there are several extensible and flexible languages available that match the driving philosophy here. The language chosen was TRIG, which is N3 using named triples, or quads, to encode efficiently individual statements that make up the knowledge, annotation, and provenance. Fundamentally, this named triple format consists of four part statements that break down into *name*, *subject*, *predicate*, and *object*. This simple format can support a rich model of allowable classes, object properties (which capture relationships), and data properties. What is key here are the named triples as they allow for statements to made about statements, which permits us to express not only relationships between defined classes and individuals but also express properties of the statements. We found quads are a more convenient format to express these properties than explicit reification with triples, though both are functionally equivalent. While there is also an implementation of our CIM in F-Logic, we choose to demonstrate examples using N3.

3.2 Model Layers: Knowledge, Annotation, and Provenance

We first define the difference we have suggested between the knowledge, annotation, and provenance layers. *Knowledge* refers to individuals and relationships that express the content found in the raw source document but are in themselves divorced from specific reference or context to that document. *Annotation* refers to the locations in the raw source document from where the associated knowledge is mentioned. *Provenance* refers to preserved information about the knowledge and annotation statements, namely when and by what analytic system these statements were produced along with confidence values that express an ideally normalized measure of accuracy of the statement.

The content of the knowledge and annotation in the data model in terms of classes and relationships took inspiration from both SUMO and APF. Within the knowledge layer, the subject-predicate-object relationship interconnects knowledge elements and this interconnection can be reasoned over. Data properties (or properties that connect individuals of classes to literal values) preserve the strings drawn from the original text. Annotation is similarly self-connected, but its data properties outline character offsets in the original document where the basis for knowledge can be located.

Predicates bridging annotation-to-knowledge, provenance-to-knowledge, or provenance-to-annotation are connected to the *names* of the statements rather than the elements. That is to say, the names of the graphs will be the subjects and objects of these predicates, indicating that statements within the knowledge and annotation layer are related to one another. This pattern of reification makes a clear line of separation between statements that are internal to knowledge, annotation, and provenance layers and the properties that connect these layers. Notably, this facilitates the isolation of knowledge in use-cases where annotation or provenance are not deemed necessary.

An annotation statement is connected to the knowledge statement for which it represents a mention. Usually, the annotation statement is linked to a statement defining the instance's class. When a knowledge statement is directly based on an annotation, these two statements are connected instead. A typical example are statements that define an entity's name, which is preserved in the knowledge and directly linked to its annotation. As part of our operating use case, every knowledge and annotation statement can have provenance properties attached. Provenance properties track the history of the statement's generation by the analytics that produce them. This removes the need to create parallel and redundant annotation structures for mentions of each type of knowledge as, for example, is done in the APF format for entities, relations, and events.

3.3 Common Format: Statement Basics

The following are examples that provide a representative description of how knowledge and annotation information relate in the common interchange model and our standard format. Even though the smallest unit of knowledge is a named triple consisting of a name, subject, predicate, and object, this is used to represent what we consider the fundamental knowledge objects: entities, frames, and designators. There is also a special class of entities that represent time. Entities are the abstract and physical things or concepts that exist. Frames represent specific

existing contexts involving multiple participants that serve some role. Designators includes names and identifying terms for entities. Each of these classes are connected to one another through object properties, which specify direct relationships between individuals, the types of designators an entity has, or an entity's role in a frame.

Unique classes or individuals are signified by node identifiers. These typically are long strings that have a detailed namespace. In the examples below, we shorten for the sake of readable examples by using the colon to indicate a previously defined namespace. Uniquely identifying strings are also shortened for readability.

3.3.1 Classes – Entities, Designators, and Time

Entities in the CIM that are equivalent to individuals require a definition statement that specifies the class.

```
<KSGP1_1> { <jane> a :Person . }
```

Entities are treated as separate from their designators. The *Designator* class includes uniquely identifying terms such as addresses, URLs, and certain numbers (e.g. *MotorVehicleLicenseNumber*) but is open to any preserved alternate names of the entity. For certain important entity classes (e.g. person, region, organization, device, email account, phone account) there is typically at least one designator object. The inclusion of different classes of designators aligned to different important entity classes was made to allow for special relationships that only apply to those types of designators. An example is provided:

```
<KSGPN1_1> { <janename> a :PersonAlternateName . }
<KSGPN1_2> { <janename> :hasString "Jane Jones" . }
<KSGPN1_3> { <jane> :hasPersonAlternateName <janename> . }
```

Note that the name has a data property that indicates the string that encodes it, often lifted from the text by an HLT analytic. *AlternateName* is the default type, but the *Alias* or *CanonicalName* classes can also be used when appropriate. These are much more specific types, and should only be used when the relationship between the entity and the name reflects the corresponding semantics. An alias is a name that is intentionally intended to obfuscate the entity's actual identity and the canonical name is the fullest form of the actual name of the entity. Names that are not known to be one of the more precise entity classes can be linked using the *hasName* object property or using *hasAlternateName* or *hasCanonicalName* as appropriate.

Time entities in our CIM have special properties that make temporal references. These are data properties based on the TIMEX2 (Ferro et al., 2005) specification and include: *hasAnchorDirection*, *hasAnchorValue*, *hasValue*, *hasModifier*, *hasNonspecific*, *hasSet*, and *hasTimeComment*. This choice was made in favor of extractor representations and the expressiveness of TIMEX2, which can capture the ambiguity and nuances of natural language. Ideally, analytics would ultimately interpret and ground these results to a more semantically formal ontology for time references, with TIMEX2 only indicating the extractor references.

3.3.2 ObjectProperty – Relationships

Extracted relationships typically map to simple entity-to-entity object properties in the model.

```
<KSGR1_1> { <jane> :memberOf <organization1> . }
```

Relationships can also have timestamps that modify them, saying when the relationship held.

```
<KSGR1_2> { <KSGR1_1> :timeAtEnd <KSGT1_1> . }
```

It is worth noting that this is an unfortunate occasion where knowledge is forced to attach to the name of the graph, something assiduously avoided elsewhere when possible. Ideally, relationships would instead be represented as an individual connected by way of object properties to their participants including the subject, object, and any related time information, but legacy issues—and the convenience of representing the end knowledge base form directly in the CIM—prevented the universal adoption of this convention.

3.3.3 Frame – Events

Frames represent larger contexts in which entities participate, including events. Frames share many features with entities in that they have a definition statement, but its defined knowledge element acts as a hub of roles involving other entities. This frame structure allows for a rich depiction of contexts or events, while specifying a data model allows for precise ontological typing and sets of valid object properties that constitute the roles of participating entities. An example frame is specified below:

```
<KSGF1_1> { <meet1> a :MeetFrame . }
<KSGF1_2> { <meet1> :hasAgent <jane> . }
<KSGF1_3> { <meet1> :hasAgent <bill> . }
<KSGF1_4> { <meet1> :hasPlace <region495> . }
```

3.3.4 Designator Example – Phone and Email

Typically important pieces of identifying information for entities with agency (in particular persons and organizations) include phone numbers and email addresses. It is worth making a small digression into specifics here to describe how these cases are handled. Phone numbers and email addresses are considered designators for phone accounts and email accounts respectively. Therefore they are indirectly connected with associated persons and organizations via the accounts which the persons or organizations use. Accounts can have multiple addresses or numbers as well as multiple users, so it makes sense to include them in the model. Example of this is specified below for email (although phone numbers and accounts follow the same pattern):

```
<KSGEL1_1> { <emailaccount1> a :EmailAccount. }
<KSGEL1_2> { <jane> :usesEmailAccount <emailaccount1>. }
<KSGEL1_3> { <emailaddress1> a :EmailAddress. }
<KSGEL1_4> { <emailaccount1> :hasEmailAddress <emailaddress1>. }
<KSGEL1_5> { <emailaddress1> :hasString “janejones@mail.com” }
```

This reveals that often there will be additional pieces of knowledge to generate in order to accurately capture certain relationships. HLT analytics typically work at a surface semantic level and directly assert a relationship between the apparent subject and object appearing in the text, rather than map directly to the desired representation of the knowledge base to be populated (e.g. it is more common to find a mention attaching a person to an email address rather than to an email account that has an email address). Any attempt to model information accurately needs to

maintain awareness of these leaps that are common in language, but do not reflect the actual relationships between the entities involved.

3.3.5 Custom Types

If there is no appropriate class in the common model for an entity identified by an analytic, the class will default *Entity*. A "custom type" can then be applied to this as a data property. While this is allowable in the model, it is intended as a temporary accommodation; ideally if such an individual needs to be represented, it should be explicitly added to the common model.

```
<KSGE2_1> { <ent32> a :Entity . }
<KSGE2_2> { <ent32> :hasCustomEntityType "SewingMachine" . }
```

Similarly, if there is no appropriate object property for a relationship in the common model, the default object property is *hasEntityRelationship*.

```
<KSGR2_1> { <jane> :hasEntityRelationship <organization2> . }
<KSGR2_2> { <KSGR2_1> :hasCustomRelationshipType "BuysFrom" . }
```

Finally, if there is no appropriate class for a frame in the common model, the default class is *Frame*.

```
<KSGF2_1> { <frm73> a :Frame . }
<KSGF2_2> { <frm73> :hasCustomFrameType "DanceCompetition" . }
```

Each of these effectively acts as a placeholder for entities, relationships, or frames that will eventually receive an appropriate class or object property in the schema.

3.3.6 Annotation Layer – Mentions

Entities extracted by analytics typically have some reference to their original document acting as supporting evidence. Mentions can be examples of both names of the entity or referring nominals or pronouns. In the case of names, the mentions should be attached to the statements connecting the entity and the appropriate designator, as the mention refers directly to this. Since nominals and pronouns are not preserved at the knowledge layers, the knowledge statement graphs for these should refer to the general entity definition statement's name.

A typical mention structure will include at least one text extent that is defined by start and end offsets in the source text. This is the span of this specific entity textual mention. Multiple text extents can be attached to an *EntityMention*, one representing the head (or the core text) of the mention while the other is the full extent of the entity's mention in the text.

This format for representing annotation is derived from how APF represents markers to indicate where its extraction appears in the raw document. An example of an entity mention (for a name) follows:

```
<RSGP1_1>  { <janeMention> a :EntityMention . }
<RSGP1_2>  { <KSGPN1_3> :hasMention <RSGP1_1> . }
<RSGP1_3>  { <janeMention> :hasHead :headExtent> . }
<RSGP1_4>  { <headExtent> a :TextExtent . }
<RSGP1_5>  { <headExtent> :hasOffsetStart "140" . }
```

```
<RSGP1_6>  { <headExtent> :hasOffsetEnd “150” . }
<RSGP1_7>  { <janeMention> :hasExtent <extExtent> . }
<RSGP1_8>  { <extExtent> a :TextExtent . }
<RSGP1_9>  { <extExtent> :hasOffsetStart “140” . }
<RSGP1_10> { <extExtent> :hasOffsetEnd “150” . }
```

An example of an entity mention (for a nominal) follows:

```
<RSGP1_11> { <janeNominal> a :EntityMention . }
<RSGP1_12> { <KSGP1_1> :hasMention <RSGP1_11> . }
<RSGP1_13> { <janeNominal> :hasHead <headExtent> . }
<RSGP1_14> { <headExtent> a :TextExtent . }
<RSGP1_15> { <headExtent> :hasOffsetStart “310” . }
<RSGP1_16> { <headExtent> :hasOffsetEnd “315” . }
<RSGP1_17> { <janeNominal> :hasExtent <extExtent> . }
<RSGP1_18> { <extExtent> a :TextExtent . }
<RSGP1_19> { <extExtent> :hasOffsetStart “290” . }
<RSGP1_20> { <extExtent> :hasOffsetEnd “339” . }
```

Relationships can also have mentions in the source text. They are captured by using a generic predicate (*hasEntityMentionRelationship*) between the entity mentions involved in the mention. This need not be marked as it can be recovered by examining the related knowledge. The full extent of the relationship is preserved at the mention level. Timestamps can modify these mentions, saying when the specific mention held. This is needed because relationships can have multiple mentions, and therefore which mention is the source of a timestamp must be maintained to preserve the full annotation. Examples of these follow:

```
<RSGRM1_1>  { <janeMention>
              :hasEntityMentionRelationship
              <orgMention1> . }
<RSGRM1_2>  { <KSGR1_1> :hasMention <RSGRM1_1> . }
<RSGRM1_3>  { <RSGRM1_1> :timeAtEnd <KSGTM1_1> . }
<RSGRM1_4>  { <relExtent> a :TextExtent . }
<RSGRM1_5>  { <RSGRM1_1> :hasExtent <RSGRM1_4> . }
<RSGRM1_6>  { <relExtent> :hasOffsetStart “444” . }
<RSGRM1_7>  { <relExtent> :hasOffsetEnd “467” . }
```

Frames can also have reference annotations. An example of frame mention follows:

```
<RSGF1_1>  { <meetMention> a :FrameMention . }
<RSGF1_2>  { <KSGF1_1> :hasMention <RSGF1_1> . }
```

This mention connects to the appropriate entity mentions with the same role types as appear in the knowledge layer frame.

3.3.7 Document Seeds and Connections

A seed *TextDocument* object representing the source document and connected to its original text serves as a single reference point to indicate that all entities, frames, and relationships appear in the same document. It also may have properties representing a zone of metadata and a zone of content; many source documents contain both.

```
<TxtGraph_1>  { <Txt_1> a :TextDocument . }
<TxtGraph_2>  { <Txt_1> :hasTextDocumentBody "…" . }
<TxtGraph_3>  { <Txt_1> :hasDocumentFilePath "…" . }
<TxtGraph_4>  { <Txt_1> :hasDocumentFileName "…" . }
<TxtGraph_5>  { <TDB_1> a :TextDocumentBody . }
<TxtGraph_6>  { <TDB_1> :hasString "…FULL TEXT…" . }
<TxtGraph_7>  { <MDZ_1> a :MetadataZone. }
<TxtGraph_8>  { <MDZ_1> :hasOffsetStart "0" . }
<TxtGraph_9>  { <MDZ_1> :hasOffsetEnd "293" . }
<TxtGraph_10> { <CNZ_1> a :ContentZone . }
<TxtGraph_11> { <CNZ_1> :hasOffsetStart "294" . }
<TxtGraph_12> { <CNZ_1> :hasOffsetEnd "34353" . }
<TxtGraph_13> { <Txt_1> :hasMetadataZone <MDZ_1> . }
<TxtGraph_14> { <Txt_1> :hasContentZone <CNZ_1> . }
```

An example triple that connects a knowledge statement graph to the document follows. These statements are typically applied to the names of definitions of entities and frames or the names of relationship statements.

```
<DocLink_1> { <KSGP1_1> :appearsInDocument <Txt_1> . }
```

3.3.8 Provenance

As stated above, provenance is attached to the names of statement graphs. There is no limitation on which statement graphs can have provenance, but typically provenance is applied to definitions of entities and frames, object properties or role triples, and statements that connect an entity, object property, or frame to a reference mention. Current provenance includes confidence values (values between 0 and 1), labels of which analytic produced a triple, labels of which version of an analytic produced a triple, and the date a triple was produced.

```
<SGP1_1> { <KSGP1_1> :hasConfidence "1.0" . }
<SGP1_2> { <KSGP1_1> :producedByAnalytic "Serif" . }
<SGP1_3> { <KSGP1_1> :producedByAnalyticVersion "2.1" . }
<SGP1_4> { <KSGP1_1> :producedOnDate "12/12/2010" . }
```

The provenance information is quite weighty as practically all knowledge and annotation statements are accompanied by four standard provenance statements. Fig. 1A depicts the rough ratio of statement types broken into knowledge statements, annotation statements, and provenance statements. The vast majority of these are provenance, at a ratio of four provenance statements to one statement of another kind. The majority of the remaining statements are

devoted to preserving annotation information. Extracted and generated knowledge amounts to only 5.5% of all statements.

Because of its weightiness, an alternative format for the provenance has been created that collapses the information into a single provenance statement that concatenates all four statements object values into a single string that can be parsed. This changes the typical distribution to 50% provenance, 36.25% annotation and 13.75% knowledge as shown in Fig. 1B.

While bearing a resemblance to the provenance statements, the *appearsInDocument* object property has a somewhat narrower coverage of statements, intended to attach entities, object properties, and frames by their definition to the source document, but not the broader spectrum of statements.

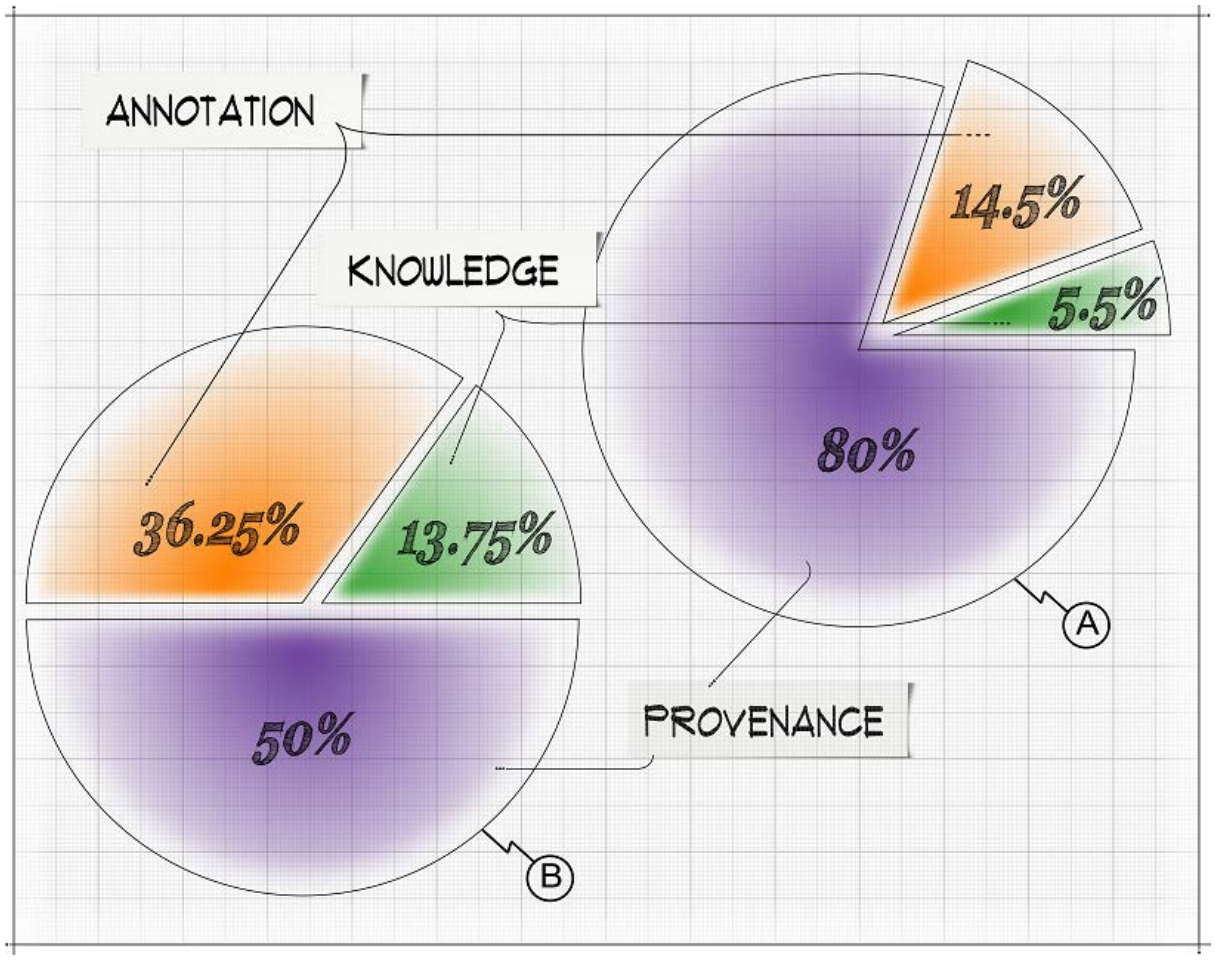

Fig. 1. Rough categorization percentages for the common interchange format. In (A) annotation and knowledge statements typically have four associated provenance statements each, which explains the 4-to-1 ratio between provenance and knowledge and annotation combined. In (B) a collapsed provenance statement reduces the overall ratio and amount of provenance.

3.3.9 Visualizing the Common Interchange Format

While highly flexible and simple in fundamental units, the named graphs can easily become confusing as they interconnect to form a complex network of properties attached to node identifiers and graph names. Fig. 2 depicts how these elements typically interconnect, using a simplified subset of named graphs.

In this example, derived from an ACE competition newswire document, there is an *Person* individual that has two alternate names as well as a business connection (captured using the *hasBusinessConnection* object property) to another *Person*. This *Person* individual has two other nominal mentions in a source document (only one of these mentions is fully expanded in the figure due to space restrictions). These mentions have heads and extents, where head is the shortest meaningful textual reference to the *Person* individual and extent is the full noun phrase representing the *Person* individual. The head and extent in turn refer to the character offsets that indicate where the mentions occur in the source document.

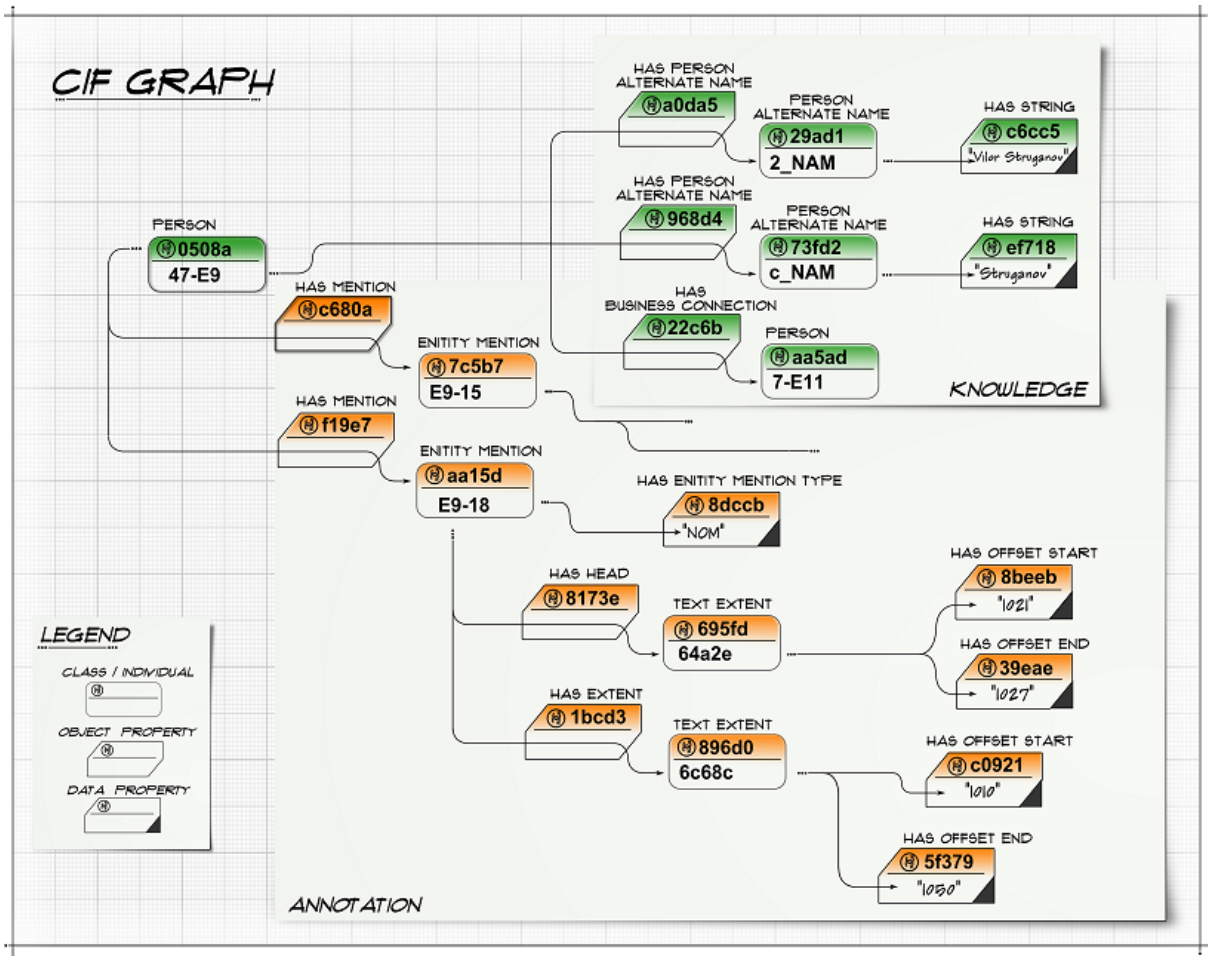

Fig. 2. Visual representation of the common interchange format (excluding the numerous provenance statements). Knowledge individuals and object properties are color-coded green and reference individuals and object properties are color-coded orange. Each individual is represented as having a class type, a graph name, and the individual's name. Each object property also has a type and a graph name, and the lines passing through them indicate the subject and object directionality of these properties. Data properties are similar to object properties, but include their object as a literal string.

The legend defines each of the diagram elements that are used to convey statements in the named graphs of CIM as figures. Each shape represents an OWL statement. The class name or property name of each statement is shown directly above its shape. The graph name of the statement is shown in the upper half of a shape (here a truncated UUID). For instance class declarations, the lower half of the shape shows its individual's name. For data properties, the literal value of the data property is presented in the lower half. A black triangle in the bottom right further identifies the data properties, as no further linkage is possible. In the case of object properties, the statement ties to individuals together and the conceptual link passes through the object property shape from the subject of the property to its object.

Statements that are part of the knowledge layer are shown in green, while statements that are part of the annotation layer are shown in orange. Provenance statements are not shown in Fig. 2 due

to size constraints, but bear in mind that each statement shown would have four provenance statements that describe the confidence, the analytic that produced the statement, the version of that analytic, and the time the statement was created. Fig. 3 includes provenance data properties for two of the statements to serve as examples.

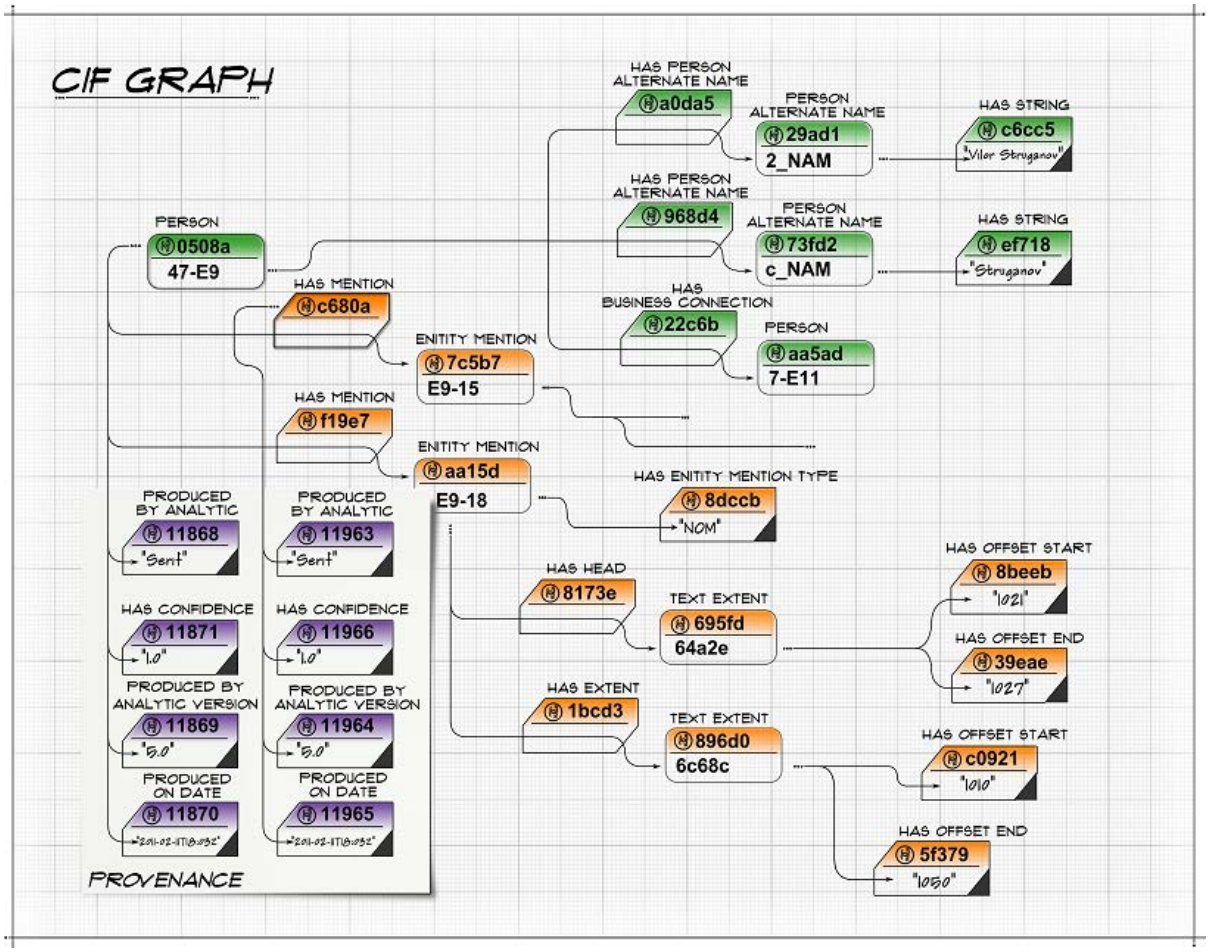

Fig. 3. Examples of the four typical provenance statements (the analytic that produced the statement, the version of that analytic, the date the statement was produced, and the confidence value) for two of the statements depicted originally in Fig. 2. Note that all knowledge and reference statements have similar provenance data properties.

Appendix A lists the named graph statements used to generate Fig. 2.

3.4 Common Interchange Model

This section goes into further detail about the common interchange model (CIM) and how it is structured, defining the classes and object properties that make up its knowledge base so that analytic and adapter developers understand how to map their results consistently. This section will be subdivided into different tiers that represent the different types of information, with more detailed descriptions of subclasses and subproperties described after the higher-level distinctions are introduced.

It should be noted that this is an evolving model that is largely driven by two considerations: 1) the types of knowledge our available analytics are capable of extracting from text, and 2) the types of knowledge which should be captured for our use cases. Fortunately, it is still relatively easy to effect significant changes in the model. Many elements in the model are still under refinement or have not been fully populated, some acting as placeholders for future deeper modeling implementations. However, as analytics and interests mature, the model will mature as well.

3.4.1 Coarse Divisions: Graph, ReferenceElement, and KnowledgeElement

Classes in the CIM hierarchy represent statement names, subjects, and objects of statements. These are organized into a hierarchy that provides structure that includes inheritance (e.g. subclasses inherit the properties of their superclass).

The highest level division in the CIM is between *Graph*, *ReferenceElement*, and *KnowledgeElement*. *Graph* is the superclass for classes that act as the names of the statements in our named graph format. This is a functional category that contains no content, but allows for easier bridging of content. *ReferenceElement* is the superclass for the classes that carry the content that directly relates knowledge to the source document annotation. *KnowledgeElement* is the superclass for the bulk of the CIM's content, representing all classes of entities, designators, etc. that are used to populate the arguments of assertions to define what analytics discover in the documents, though this category is not necessarily limited to analytic results on documents being its basis.

3.4.1.1 Graph and Subclasses

The Graph class consists of three major subclasses: *KnowledgeStatementGraph*, *ReferenceStatementGraph*, and *TabularGraph*. Each of these is intended to represent the name of a different kind of named graph statement. *KnowledgeStatementGraph* is the class for statement names that reside at the knowledge layer of the CIM. *ReferenceStatementGraph* is the class for statement names that reside at the document annotation, or reference, layer of the CIM. *TabularGraph* is the class for statement names that contain tabular information collected from analytics that describe the source document. The first two of these classes represent the fundamental division in the CIM: 1) the knowledge layer, where assertions act as definitions or facts about classes and individuals of classes, and 2) the reference layer, where assertions point into the text of the source document such that the basis for asserting facts is preserved and can be examined. The last of these classes, *TabularGraph*, handles the special analytics that do not extract knowledge from the document, but instead provide analytic results describing features of the document itself.

3.4.1.2 ReferenceElement and Subclasses

The ReferenceElement class has three major subclasses: *Document*, *Metadata*, and *Annotation*. The *Document* class is at present split into three subclasses, representing different types of documents that are the source document from which knowledge and annotations are generated. These include *TextDocument*, *AudioDocument*, and *ImageDocument*. These represent documents that are textual, audio, or images, respectively. Metadata breaks down into subclasses of *AnalyticMetadata* and *DocumentMetadata*. Both of these are currently placeholders in advance

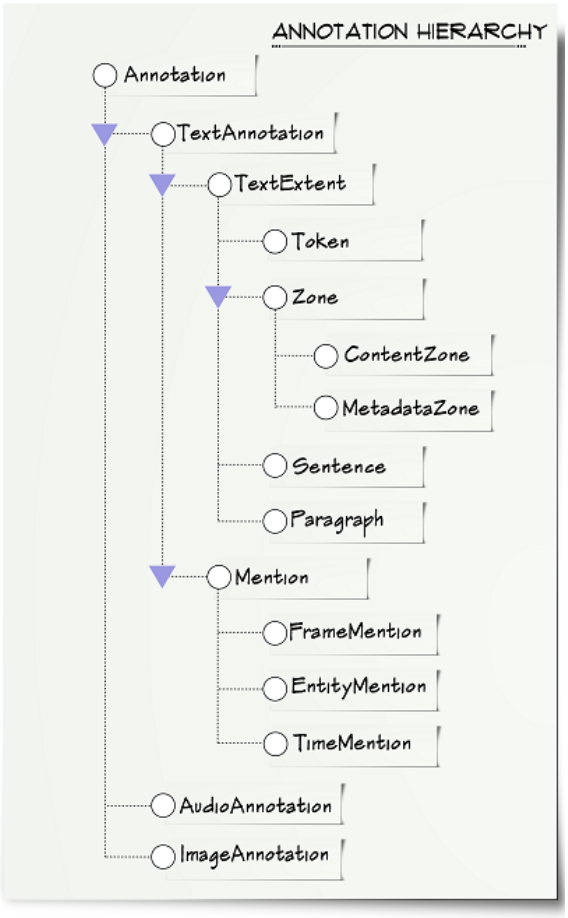

Fig. 4. Annotation hierarchy in the CIM

of a more formalized way of representing metadata as classes. Note that metadata here does not refer to the provenance described earlier.

Fig. 4 depicts the class hierarchy of *Annotation*. This is divided into media-specific categories. Currently, only the *TextAnnotation*, which represents annotations on textual sources, has a deeper representation, where its subclasses are *TextExtent*, which represents sub-spans of the original document, and *Mention*, which represents the presence of a reference to a knowledge object in the original document and is divided into the major knowledge element types, including *EntityMention*, *FrameMention*, and *TimeMention*. *TextExtent* breaks down into the major interesting subdivisions of text, including *Sentence* and *Paragraph* as determined by grammar, format, and punctuation, as well as *Zone* and *Token* as determined by white-space and, typically, XML tags.

All reference graphs consist of some combination of these elements. As described in the format, it is expected that a *hasMention* object property will connect the statement names of a knowledge element and its corresponding reference element, which is an individual of a subclass of *Mention*. This individual will then be connected to individuals of *TextExtent* by way of the *hasHead* (for the core reference text) or *hasExtent* (for the full extent) object properties. These in turn point to character offsets in the document to show their boundaries by way of the data properties *hasOffsetStart* and *hasOffsetEnd*.

3.4.1.3 KnowledgeElement and Subclasses

The bulk of the common interchange model's complexity is found in the subclasses of *KnowledgeElement*, whose highest level subclasses are shown in Fig. 5. These subclasses are divided into five main categories: *Entity*, *Frame*, *Designator*, *TabularElement*, and *MiscDataType*. *MiscDataType* contains information about the common model itself as well as certain types that defy categorization in the present hierarchy, such as Penn Treebank bracket tags for syntax. *TabularElement* is currently a structure used to support storage of tabular results. A *Table* individual acts as the structure representing a table of data with potentially many attached *DataEntry* individuals. Each *DataEntry* individual has both a string primitive to store its value and a *DataEntryLabel* individual to serve as its tabular label.

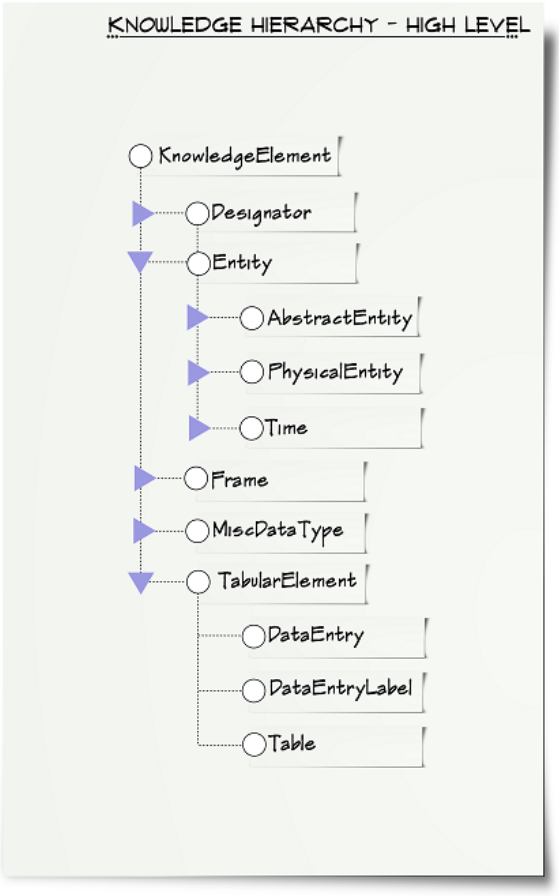

Fig. 5. Knowledge hierarchy in the CIM

Entity is the most complex of these categories as most objects can be classified broadly as an Entity, falling into further classifications of *PhysicalEntity*, *AbstractEntity*, or *Time*. Of particular importance among the subclasses of *PhysicalEntity* is the *Agent* class, which indicates any *PhysicalEntity* that can take action, as these are usually among the more interesting entities. *Designator* contains various subclasses of names and identifiers for the different entity objects. A *Frame* contains various types of contexts which can have entity participants.

A deeper discussion of the *Entity* subtree could be quite extensive. As suggested earlier, the structure of this subtree is heavily influenced by the Suggested Upper Merged Ontology (SUMO) and the type content by the Automatic Content Extraction Pilot Format (APF). The basic hierarchy here is split into entities that have physicality (*PhysicalEntity*), those that are abstract or concepts (*AbstractEntity*), and those that are subclasses of time (*Time*).

The subtree of the Designator class includes all designators for the key namable *Entity* classes (in particular *Person*, *Organization*, *Region*, and *Device*), an *Address* class, and an *Identifier* class. Address breaks down into *PhysicalAddress* for addresses in real-world space and *LogicalAddress* for addresses in cyber-space. Identifier breaks down into specific uniquely identifying terms, such as *EmailAddress* and *PhoneNumber*. The other subclasses are for each of the major *Entity* types described and they are split into identifier, alternate names, and canonical name. Most names should map into the alternate category with canonical names being reserved only for cases where there is a high certainty of the name defining the *Entity*. Additionally, *PersonDesignator* has subclasses for *PersonAlias* and *PersonTitle*, which represents attachments to the person's name based on birth, education, gender, or occupational role.

The *Frame* class encompasses many different contexts in which *Entity* individuals will participate and be interrelated to one another. This allows for a hub from which to represent these complex contextual relationships. There is some hierarchical organization to its subtree, but it is mostly flat as the *Frame* subclasses are intended to be anchors to which the contextual relationships are attached.

3.4.2 Object Properties

At the highest level, object properties in the CIM hierarchy represent the predicates of statements. Like classes, these have a hierarchical structure that includes inheritance of properties, where subproperties have a greater specificity of their superproperties in terms of domain and range. The following sections describe each of these high-level properties and their subhierarchies in greater detail.

- *hasAnnotation*: hierarchy of predicates that relate different reference level classes.
- *hasEntityRelationship*: hierarchy of predicates that connect entities to other entities.
- *hasEntityProperty*: hierarchy of predicates that connect entities to properties or attributes they have.
- *hasDesignatorRelationship*: hierarchy of predicates that connect designators to other knowledge elements.
- *hasFrameRole*: hierarchy of predicates that describe a relationship an object entity has in the specified frame of context.
- *hasTimestamp*: hierarchy of predicates that connect knowledge or reference statements for relationships or frame classes to time entities that describe when the relationship held or the frame context is temporally constrained.
- *hasTableObject*: hierarchy of predicates that interrelates tabular data classes to one another.
- *hasEntityDocumentConnection*: hierarchy of predicates that connect an entity to a source document.

Certain object properties are not hierarchical, but do not fit into any of these particular categories, the most important of these being the properties that accommodate the preservation of annotation information. One such example is *hasMention*, the key predicate for connecting knowledge and reference elements in the common interchange model. The subject of this object property is a *KnowledgeStatementGraph* while the object is a *ReferenceStatementGraph*. This connects entities, relationships, and frames to their mentions. Another important example is *hasEntityMentionRelationship*, a reference layer predicate connecting two entity mentions captured in a relationship that appears in the knowledge layer.

3.4.2.1 hasAnnotation

Currently, *hasAnnotation* has three subproperties: *hasHead*, *hasExtent*, and *hasAnchor*. The property *hasHead* is used to connect an *EntityMention* to a *TextAnnotation* individual that represents core text that represents the entity. According to the APF definition from which this is derived, this would be the final word in the extent in the case of a nominal mention and would be the full proper name in the case of a named mention. The property *hasExtent* connects a *Mention* to a *TextAnnotation* that represents the entire noun phrase of an entity as well as the full extent necessary to capture a relationship or the context of a frame. Further, *hasAnchor* will connect a *Frame* to a *TextAnnotation* that represents the key word or phrase from which the frame's context is being generated.

3.4.2.2 hasEntityRelationship

Subproperties of *hasEntityRelationship* represent a relationship between two distinct entities. Although the top level property is purely structural and generic, the intent of the subproperties is to minimize ambiguity. Certain key subproperties and their subtrees are described here to give a picture of the scope of coverage.

For relationships that indicate the subject is part of the object, whether geographically, organizationally, or physically (e.g., a subcomponent of a *Device* or a snippet of a *Text* object) then *isPartOf* or one of its more specific subproperties is used. More temporary relationships that indicate location and proximity make use of the *hasPhysicalLocationRelationship* with subproperties that indicate when subjects are at or near *Region* objects. To indicate a relationship between an *Agent* and some *Artifact* (i.e., a human-created item), any number of subproperties of *hasAgentArtifactRelationship* can be used to indicate the *Agent* is the manufacturer, owner, user, inventor, sender, or recipient of the *Artifact*. Encompassing a varied set of relationships of membership and affiliation that entities can have, *hasAffiliation* has several diverse subproperties that include citizenship, ethnicity, religion, residence, and language capability.

Some of the most important relationships are subproperties of *hasPersonRelationship*, which covers relationships that feature largely *Person* to *Person* relationships. Example relationships here cover blood or marriage familial connections, business connections, personal relationships, and relationships by different communication methods.

Some relationships defy the application of a specific type, but it is still useful to know that there is at least a contextual connection between the participants. For a generic relationship such as this between two knowledge elements, indicating they appeared in close proximity within a document, *hasContextualRelationship* is used.

3.4.2.3 hasEntityProperty

Subproperties *hasEntityProperty* represent a relationship where object is perceived to be a property of the subject. Subjects of these subproperties can be quite general in their expected types or quite specific, often an *Agent* or *Person* individual. A wide variety of properties are covered such as identifiers (e.g., names, unique identifying numbers, addresses, titles), statuses (e.g., employment status, educational level, legal and marital status) and physical features (e.g., age, height, weight, eye and hair color).

3.4.2.4 hasEntityDocumentRelationship

This category of object properties connect entities to source documents, and they break down into three main subproperties that include *sourceReferencedIn*, a relationship to attach a *Text* subject to the *Document* in which it appears, *isAuthorOf*, an authorship relationship with an *Agent* subject and a *Document* object, and *appearsInDocument*, a relationship that connects *KnowledgeElement*, *ReferenceElement*, or *Graph* subjects to a *Document* object. These different relationships could alternatively viewed as examples of provenance, but this representation was chosen given the sometimes immediate relationship between the content of the document and what produced it.

3.4.2.5 hasDesignatorRelationship

This category of object properties covers relationships that have a *Designator* as the subject. Its subproperties represent *Designator* individuals that have been transliterated into other character sets, for example *hasLatinTransliteration, hasChineseTransliteration,* and *hasCyrillicTransliteration*.

3.4.2.6 hasTimestamp

The subclasses of *hasTimestamp* bridge a *Frame* or a relationship object property—where the graph of the relation is used as the subject—to a *Time* object. These relationships are directly derived from the eight timestamp definitions that appear in the relation and event timestamping guide for the ACE model.[7]

- *timeAtEnd*: indicates the statement occurs or holds at the end of a *Time*.
- *timeAtBeginning*: indicates the statement occurs or holds at the beginning of a *Time*.
- *timeEnding*: indicates the statement ends at the object *Time*.
- *timeStarting*: indicates the statement begins at the object *Time*.
- *timeWithin*: indicates the statement occurs or holds within the object *Time* duration.
- *timeHolding*: indicates the statement occurs or holds throughout the object *Time* duration.
- *timeBefore*: indicates the statement occurs or holds before the object *Time*.
- *timeAfter*: indicates the statement occurs or holds after the object *Time*.

3.4.2.7 hasTableObject

This is the category of object properties that specify relationships between tabular data entries. Its two subclasses are *hasDataEntry***,** which connects a *Table* object to a *DataEntry*, and *hasDataEntryLabel***,** which connects a *DataEntry* to a *DataEntryLabel*.

3.4.2.8 hasFrameRole

This category contains all the object properties that relate the different types of *Frame* classes to the *Entity* individuals that participate in their contexts. This is a fairly complex unstructured list, and the types appearing here were directly influenced by event roles in the APF and the specified participant roles in the event candidates in the METEOR, an analytic system for recognizing and reasoning over certain event classes (Taylor et al, 2009). A full description of each of these would be very detailed, therefore it is recommended at present to review the documentation of those two sources for further description on these as the definitions of the object properties here adhere to their documented meanings.

3.4.3 Data Properties

Data properties, as the term is used in this work, are properties where the value is a literal, often a string. In the CIM, there is not considerable hierarchical organization to the data properties,

[7] http://projects.ldc.upenn.edu/ace/docs/English-TimestampingGuidelines_v3.pdf

which are the properties that relate individuals to literals, in particular string literals in this case. The most important data properties are listed below:

- *hasString*: attaches a string literal to any *Graph*, *KnowledgeElement*, or *ReferenceElement*. This is the default manner by which content that is not hard-typed in the model's representation is attached, typically in the knowledge layer to subclasses of *Designator*.
- *hasDataEntryValue*: for *DataEntry* individual subjects, attaches the entry's value and places it in the tabular structure.
- *hasConfidence*: attaches a provenance confidence value to a statement that represents how likely it is that the statement holds.
- *producedByAnalytic*: attaches provenance to a statement that specifies what analytic produced the statement.
- *producedByAnalyticVersion*: attaches provenance to a statement that specifies what version of an analytic produced the statement.
- *producedOnDate*: attaches provenance to a statement that specifies the date on which the statement was produced.
- *hasCustomEntityType*: attaches an unmodeled entity type to an *Entity* individual.
- *hasCustomFrameType*: attaches an unmodeled frame type to a *Frame* individual.
- *hasCustomRelationshipType*: attaches an unmodeled relationship type to a generic *hasRelationship* statement.

Remaining data properties are typically intended to preserve analytic annotation information that is not modeled as hard types in the CIM, such as *hasAnchorValue* and *hasAnchorDirection* for Timex-based data or *hasModality* for ACE relations.

4 Discussion

This paper has presented ongoing work to develop a common interchange model, CIM, implemented in a flexible and extensible format, CIF. The aim is to represent the varied results generated across diverse HLT analytics performing content extraction from raw unstructured text. The ability to represent both knowledge and annotation drawn from these sources was a key goal of this development and led to the genesis of the CIM out of existing efforts in ontology (SUMO) and analytic artifact representation (APF). Using OWL to represent the model and TRIG named triples as a format allowed for the extensibility and flexibility sought while at the same time maintaining a known and established structure for data representation. A longer overview and description of the implementation of MOSAIC, which makes use of the CIM, featuring specific use cases can be found in (Winder et al, 2011; Winder et al, 2013). At the time this document was written, 28 different analytics, each with its own raw analytic artifact model and format, have been mapped into the CIM. This number is expected to grow as new analytics need to have their results legible by a knowledge base designed to store and make inference over the knowledge captured using the CIM.

There is still a wide variety of future work required before we resolve all representational issues for the CIM, and it is unlikely that the CIM will converge to a point where there is no longer a need or desire to add new classes, object properties, and data properties, as the space of what HLT analytics are capable of producing is always in flux and expanding.

The development of CIM and CIF began in 2010, and since that time significant solutions have emerged in the space of HLT and NLP, such as the NIF described in the background. The striking similarity of intent behind the NIF and our CIM and CIF suggests the NIF could provide an alternative robust underlying NLP model for our work. Alternatively, the CIM and CIF's knowledge layer could serve in the capacity of background knowledge that boosts applications natively or adapted into NIF's format. Ultimately, the community decides how to make use of provided capabilities, but there is potential for these models to benefit and enrich one another.

Paramount among the long-term issues with respect to the CIM that should be addressed is creating a more cohesive and correct model of time. While TIMEX2 is a rich linguistic model of time, the current model treats time statements as data properties rather than attempting to map them to the specific time and date classes present in the CIM (e.g. *Day*, *Month*, *Year*). Generating this unified model of time is such a significant effort that it will be a long-running undertaking. In representing time, other existing methods apart from TIMEX2 that could be used include TimeML, which can also specify temporal expressions as related to events that appear in natural language (Pustejovsky et al., 2003).

There are also some core difficulties in maintaining the integrity of the knowledge layer representation when it comes to time, especially when timestamps are attached to relationship statements. Because relationships are represented in the model as object properties, the only way to attach a timestamp to the relationship is by using the name as the subject, which violates how knowledge and annotation and provenance should distinguish themselves. This requires a change in either how relationships are represented (perhaps by instantiating them as individuals of a class that use object properties of *hasSubject* and *hasObject* to indicate their binary participants)

or redefining what relationships are (perhaps by considering any relationship that can have a temporal property to be more equivalent to an event or frame).

One of the other considerations that has emerged is the efficiency concern. The overriding goal at the outset was to emphasize adaptability and extensibility, but as the model is populated and requires querying and reasoning, whether or not the representation is capable of handling these requests efficiently must be investigated.

There are also many minor concerns to be handled in future iterations of the model as well, such as an elimination of redundancy in the types of object properties and classes (e.g., is it necessary to have a *hasPersonAlternateName* relationship over *hasName* when its subject and object are typed as *Person* and *AlternateName*, respectively?), a richer hierarchical representation (e.g., should the frames which represent events be subclasses of the class of event they represent?), and a re-characterization of what constitutes knowledge, annotation, and provenance in some cases (e.g., should an entity's appearance in a document be treated as part of provenance?).

Additionally, as the common model for the HLT domain is refined, the same process of developing a CIM and choosing a CIF can be applied to other domains (e.g., image/video processing, signal processing) where the ambiguities and restrictions witnessed here may or may not hold but a uniform representation is required.

5 References

1. Baker, C., Fillmore, C, & Lowe, J. (1998). The Berkeley FrameNet project. *COLING '98 Proceedings of the 17th International Conference on Computational Linguistics*. Volume 1, (pp. 86-90).
2. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., & Weischedel, R. (2004). The automatic content extraction (ACE) program tasks, data, and evaluation. *Proceedings of LREC*. (pp. 837-840).
3. Ferro, L., Gerber, L., Mani, I., Sundheim, B., & Wilson, G. (2005). TIDES—2005 Standard for the Annotation of Temporal Expressions. *Technical Report*. MITRE. http://timex2.mitre.org/annotation_guidelines/2005_timex2_standard_v1.1.pdf
4. Gotz, T. & Suhre, O. (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal*. 43(3): 476-489.
5. Hellmann, S., Lehmann, J., Auer, S., & Brümmer, M. (2013). Integrating NLP using linked data. In *The Semantic Web–ISWC 2013* (pp. 98-113). Springer Berlin Heidelberg.
6. Ide, N. & Suderman, K. (2007). GrAF: A graph-based format for linguistic annotation. *Proceedings of the Linguistic Annotation Workshop*. (pp. 1-8).
7. Ide, N. & Suderman, K. (2009). Bridging the Gaps: Interoperability for GrAF, GATE, and UIMA. *Proceedings of the Third Linguistic Annotation Workshop*. (pp. 27-34).
8. Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*. 42(4): 741-843.
9. Motik, N., Patel-Schneider, P, & Parsia, B. (2009). OWL 2 web ontology language structural specification and functional-style syntax. *Technical Report*. W3C. http://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20090914/all.pdf
10. Niles, I. & Peace, A. (2001). Towards a standard upper ontology. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems.
11. Peace, A., Niles, I., & Li, J. (2002). The suggested upper merged ontology: a large ontology for the semantic web and its applications. *Working Notes of the AAAI 2002 Workshop on Ontologies and the Semantic Web*.
12. Pustejovsky, J., Castano, J., Ingria, R., Sauri, R., Gaizauskas, R., Setzer, A., & Katz, G. (2003). TimeML: robust specification of event and temporal expressions in text. *IWCS-5, 5th International Workshop on Computational Semantics*.
13. Taylor, M., Carlson, L., Fontaine, S., & Poisson, S. (2009). Searching Semantic Resources for Complex Selectional Restrictions to Support Lexical Acquisition. In *Advances in Semantic Processing, 2009. SEMAPRO'09. Third International Conference on* (pp. 92-97). IEEE.
14. Winder, R., Jubinski, J., Prange, J., & Giles, N. (2013). MOSAIC: a cohesive method for orchestrating discrete analytics in a distributed model. In *Natural Language Processing and Information Systems* (pp. 260-265). Springer Berlin Heidelberg.
15. Winder, R., Giles, N., & Jubinski, J. (2011). Implementation Recommendations for MOSAIC: A Workflow Architecture for Analytic Enrichment. Analysis and

Recommendations for the Implementation of a Cohesive Method for Orchestrating Analytics in a Distributed Model. *Report No. MTR-MNI-000-012*. MITRE, McLean VA.

Appendix A

The following are the statements used to generate Fig. 2. The full unique knowledge element names are preserved here, while the figure shows only the final five characters of any node identifier due to size restrictions.

```
<http://kbs.mitre.org/caf3f740-498e-4e9c-bdf1-1200ae5aa5ad> {
<http://kbs.mitre.org/d569dc45-c44a-457d-8a83-c65a629713be_AFP_ENG_20030701.0247-E11>
          a       ns0:Person .}
<http://kbs.mitre.org/c1fda123-7a2e-4d2a-a2e9-665a2a90508a> {
<http://kbs.mitre.org/0bfb0d84-3b97-4cf7-ad50-01a15f706b6a_AFP_ENG_20030701.0247-E9>
          a       ns0:Person .}
<http://kbs.mitre.org/e0616368-a420-4395-a9fe-1a3104022c6b> {
<http://kbs.mitre.org/0bfb0d84-3b97-4cf7-ad50-01a15f706b6a_AFP_ENG_20030701.0247-E9>
          ns0:hasBusinessConnection
<http://kbs.mitre.org/d569dc45-c44a-457d-8a83-c65a629713be_AFP_ENG_20030701.0247-E11> .}
<http://kbs.mitre.org/9e2b496c-dee1-40ea-8d5e-cb3b152a0da5> {
<http://kbs.mitre.org/0bfb0d84-3b97-4cf7-ad50-01a15f706b6a_AFP_ENG_20030701.0247-E9>
          ns0:hasPersonAlternateName
                  <http://kbs.mitre.org/29677300-226d-4211-9755-202458ded142_NAM> .}
<http://kbs.mitre.org/a24c2258-dc15-483b-9171-79b1f7729ad1> {
<http://kbs.mitre.org/29677300-226d-4211-9755-202458ded142_NAM>
          a       ns0:PersonAlternateName .}
<http://kbs.mitre.org/e6ff7a8c-46fd-4ab1-afc2-66f3da0c6cc5> {
<http://kbs.mitre.org/29677300-226d-4211-9755-202458ded142_NAM>
          ns0:hasString "Vilor Strugan♀v" .}
<http://kbs.mitre.org/007c44ad-de70-4645-840c-ec31aeb968d4> {
<http://kbs.mitre.org/0bfb0d84-3b97-4cf7-ad50-01a15f706b6a_AFP_ENG_20030701.0247-E9>
          ns0:hasPersonAlternateName
                  <http://kbs.mitre.org/cbb1866d-0460-4aef-a296-b1a31f89916c_NAM> .}
<http://kbs.mitre.org/edeef9f0-eab7-4e5e-907b-e7915fc73fd2> {
<http://kbs.mitre.org/cbb1866d-0460-4aef-a296-b1a31f89916c_NAM>
          a       ns0:PersonAlternateName .}
<http://kbs.mitre.org/75f007f6-6cb1-4e67-a251-2888f5fef718> {
<http://kbs.mitre.org/cbb1866d-0460-4aef-a296-b1a31f89916c_NAM>
          ns0:hasString "Strugan♀v" .}
<http://kbs.mitre.org/cf4c2f67-0dc2-4192-9240-b78a761c680a> {
<http://kbs.mitre.org/c1fda123-7a2e-4d2a-a2e9-665a2a90508a>
          ns0:hasMention <http://kbs.mitre.org/b620dd12-17a4-4c77 8782-5eee2d07c5b7> .}
<http://kbs.mitre.org/b620dd12-17a4-4c77-8782-5eee2d07c5b7> {
<http://kbs.mitre.org/9da998b2-998d-45b9-929e-0226b842100d_AFP_ENG_20030701.0247-E9-15>
          a       ns0:EntityMention .}
<http://kbs.mitre.org/4df37c6e-9133-464a-bc65-95387dcf19e7> {
<http://kbs.mitre.org/c1fda123-7a2e-4d2a-a2e9-665a2a90508a>
          ns0:hasMention <http://kbs.mitre.org/4e810189-ec06-4ae3-a868-e9e69d4aa15d> .}
<http://kbs.mitre.org/4e810189-ec06-4ae3-a868-e9e69d4aa15d> {
<http://kbs.mitre.org/405f7a65-3303-4b75-9e87-8a202e6dfc74_AFP_ENG_20030701.0247-E9-18>
          a       ns0:EntityMention .}
<http://kbs.mitre.org/f856f602-c842-4728-a5aa-17a0a858dccb> {
<http://kbs.mitre.org/405f7a65-3303-4b75-9e87-8a202e6dfc74_AFP_ENG_20030701.0247-E9-18>
          ns0:hasEntityMentionType
                  "NOM" .}
<http://kbs.mitre.org/c12ff6f9-f97b-4aa6-a216-3731c5efa3b8> {
<http://kbs.mitre.org/405f7a65-3303-4b75-9e87-8a202e6dfc74_AFP_ENG_20030701.0247-E9-18>
          ns0:hasEntityMentionRelationship
<http://kbs.mitre.org/c7103f39-bc96-4e99-9f86-1d3ef4e33ea2_AFP_ENG_20030701.0247-E11-19> .}
<http://kbs.mitre.org/5acdd82a-b151-4da1-aa18-66b149f8173e> {
<http://kbs.mitre.org/405f7a65-3303-4b75-9e87-8a202e6dfc74_AFP_ENG_20030701.0247-E9-18>
```

```
          ns0:hasHead <http://kbs.mitre.org/ea84d876-dfa8-4b11-9434-c14fc4364a2e> .}
<http://kbs.mitre.org/988197e0-4ef2-48e4-b4d2-b1b3681695fd> {
<http://kbs.mitre.org/ea84d876-dfa8-4b11-9434-c14fc4364a2e>
          a       ns0:TextExtent .}
<http://kbs.mitre.org/739251a8-a473-4cf4-8a11-c9ad8578beeb> {
<http://kbs.mitre.org/ea84d876-dfa8-4b11-9434-c14fc4364a2e>
          ns0:hasOffsetStart "1021" .}
<http://kbs.mitre.org/818c471a-1970-404c-89a5-9aa98e039eae> {
<http://kbs.mitre.org/ea84d876-dfa8-4b11-9434-c14fc4364a2e>
          ns0:hasOffsetEnd "1027" .}
<http://kbs.mitre.org/1758fb16-b84e-419c-a514-c9769011bcd3> {
<http://kbs.mitre.org/405f7a65-3303-4b75-9e87-8a202e6dfc74_AFP_ENG_20030701.0247-E9-18>
          ns0:hasExtent <http://kbs.mitre.org/31fdf3d2-8125-426a-9287-7f4c7e26c68c> .}
<http://kbs.mitre.org/5c723e4f-c345-49e4-897e-4795f46896d0> {
<http://kbs.mitre.org/31fdf3d2-8125-426a-9287-7f4c7e26c68c>
          a       ns0:TextExtent .}
<http://kbs.mitre.org/e639404e-0a1f-4316-a59b-1d2e900c0921> {
<http://kbs.mitre.org/31fdf3d2-8125-426a-9287-7f4c7e26c68c>
          ns0:hasOffsetStart "1010" .}
<http://kbs.mitre.org/db924345-1a45-4948-9bbd-462e0395f379> {
<http://kbs.mitre.org/31fdf3d2-8125-426a-9287-7f4c7e26c68c>
          ns0:hasOffsetEnd "1050" .}
```