Representing metadata constraints in Common Logic

Harry S. Delugach

Computer Science Department,
University of Alabama in Huntsville,
Huntsville, AL 35899, USA
Email: delugach@cs.uah.edu

Abstract: This paper presents a standards-based approach to specifying and reasoning about metadata constraints based on existing metadata registry standards, existing terminologies and Common Logic. The paper introduces Common Logic's basic principles and theory, and applies it to several examples where we might want to reason about metadata constraints. It is our expectation that these additional constraints will be developed by the makers of the metadata for a given data source, and will be included in a metadata registry for exchange with any subscriber to the metadata.

Keywords: Common Logic; metadata; constraints; standards.

Reference to this paper should be made as follows: Delugach, H.S. (2009) 'Representing metadata constraints in Common Logic', *Int. J. Metadata, Semantics and Ontology*, Vol. 4, No. 4, pp.277–286.

Biographical notes: Harry S. Delugach is an Associate Professor of Computer Science at the University of Alabama in Huntsville. He has over 20 years of teaching and research experience, as well as in knowledge-based systems, conceptual graphs, Common Logic and formal models in software engineering. He serves on several conference programme committees, including a senior role in the International Conference on Conceptual Structures (ICCS). He is the author of CharGer, an open-source conceptual graph visualisation package. He serves on the USA's technical advisory groups to ISO/IEC JTC1's SC32 subcommittee on data interchange, where he edited the Common Logic standard (ISO/IEC 24707:2007).

1 Introduction

This paper deals with formal representations and reasoning about metadata in order to perform 'smart' queries of data repositories. As data users become more and more sophisticated, they tend to expect more 'brains' from their data collections. 'I know the data is there, so why can't the system tell me what I want to know?' Two main obstacles to solving this problem are well known:

- Large numbers of database schemas, whose integration is typically very difficult.
- Standard approaches are not suitable for information sources that are not relational databases with available schemas.

As a result, most recent approaches are centred around some kind of domain model, as in the SIMS project by Arens et al. (1996). [For further discussion about the general problem of intelligent queries, see Arens et al. (1996), Sowa (2000) and Bertino et al. (2001).] But even these approaches have their limitations:

- Domain models are still tightly bound to the structure of the database (or more generally, information source) for their semantics.
- For most applications, there are different points of view from which to establish a domain model, some of them user-based (end-user, novice, expert, auditor etc.) and others based on different aspects of the domain (temporal, spatial, economic, terminological etc.). For an early discussion of viewpoints from a developer's perspective, see Delugach (1992, 1996).

This paper presents a standards-based alternative that lies in between the two kinds of approaches. While it does rely on database structure to some extent, it relies more heavily on the metadata of the information source, whose semantics can be captured by a general domain model using another standard, namely Common Logic

It is our expectation that these additional constraints will be developed by the makers of the metadata for a given data source, and that the constraints (in some dialect of Common Logic) will be included in a metadata registry for interchange with any subscriber to the metadata.

1.1 Why do we need metadata?

We usually start out a discussion of metadata by saying that it is 'data about data,' but most practitioners have something more specific in mind. In general, metadata is anything we want to say about our data other than its values and types. This includes such knowledge as:

- What are the possible values of a data element with respect to other concepts?
- What does each of the values mean?
- What concept is being represented by the data?
- How is that concept related to other concepts in the information source?

We need metadata (whether we call it that or not) in order to use the data to represent things that matter to us: to understand phenomena, to better serve customers, to establish organisational policies, and/or to keep a more accurate record of human activities. Data does not just exist in a repository – it is stored, used, and modified as a reflection of whatever objects or activities it is meant to represent. We do more than just represent the data; we also want to reason about it, drawing conclusions about those real world objects during the pursuit of normal human activity. It is therefore widely understood that we need the ability to formally reason about data. Metadata is one important aspect to that ability.

For definitions and terminology about metadata and conceptual modelling used in this paper, readers are directed to ISO/IEC 11179 [specifically parts 1, 3 and 4 (ISO/IEC, 2003; ISO/IEC, 2004a; ISO/IEC, 2004b)] and two related terminology standards (ISO, 2000a; ISO, 2000b).

1.2 What are metadata constraints?

For the purposes of this paper, we consider two kinds of knowledge commonly considered metadata constraints:

Domain constraint: any formally specified condition that further limits the values or contents of a data element beyond its database definition, where definition is considered in the sense of ISO/IEC (2004a).

Metadata relationship: any formally specified relationship between metadata items where the relationship is largely specified in terms of metadata item attributes.

In this paper, we will focus on metadata relationships, and show how they are represented to semantically enhance existing metadata and how these relationships can be used in answering practical queries. In order to follow a standards-based approach to representing the relationships, we will use Common Logic ISO/IEC (2007) as our semantic representation.

The Digital Library Federation identifies three types of metadata about digital resources; although aimed at libraries, these categories are relevant for our purposes as well:

descriptive metadata: information describing the intellectual content of the object; we will call this the *semantics* of the objects being described.

administrative metadata: information necessary to allow a repository to manage the object: this can include information on how it was obtained, its storage format etc.; we will consider this part of the *pragmatics* of the objects being described.

structural metadata: information that ties each object to others to make up logical units (for example information that relates individual images of pages from a book to the others that make up the book itself); this we will also consider part of the *pragmatics* of the objects being described.

This paper shows examples of all three types of metadata. Our purpose is to illustrate the need for reasoning methods that use metadata constraints, and to show how Common Logic would represent those constraints.

Note that this paper uses several examples from the pharmacologic domain. These are not meant to be exhaustive or complete with respect to a full medical knowledge base and are intentionally simplified for illustration purposes.

2 Background

This section provides an introduction to Common Logic and explains why we believe it is a useful representation to support reasoning with metadata. First we briefly illustrate some basic limitations of metadata for reasoning, and then introduce Common Logic which we will later use as a representation to overcome some of the limitations.

2.1 Limitations of metadata for reasoning

For the approach in this paper, we assume that there already exists a metadata model that associates specific data elements with particular concepts. That is, our domain constraints and metadata relationships will be expressed primarily at the conceptual domain level (or 'higher') for the purposes of reasoning.

Looking at a data repository's content using concepts largely frees data users from the burden of trying to figure out the details what is in a database; however, reasoning about those concepts using databases is still done mostly in an ad-hoc case-by-case manner, through custom programming as in the following illustration.

Assume we have on hand a database of medications as in Figure 1(a) with metadata in the form of definitions also available as English text. Assume we also have a hospital record of drugs administered with information such as in Figure 1(b), again with appropriate metadata.

To motivate our illustration, suppose we want an answer to the query: *How many times have analgesics been prescribed in this particular hospital?*

Since the term 'analgesic' does not actually appear in the hospital drug record, a very naïve (and wrong!) answer would be 'zero'. A knowledgeable domain specialist will know that *analgesic* is a general term comprising all pain medications; they can easily create a custom query that will search a drug catalogue for all known pain medications from a list. But suppose we next want to know how many time antibiotics have been prescribed in the same hospital? Another ad-hoc custom query will have to be created, even though the structure and process is exactly the same. This is precisely the kind of duplication that metadata constraints will help us avoid.

Figure 1 Database examples for metadata illustration

Drug	Class	Form	Dose
Acetaminophen	Analgesic	Capsule	200 mg
Aminoglycoside	Antibiotic	Liquid	15 mg
Aspirin	Analgesic	Pill	250 mg
. . .	. . .	. . .	

(a) *Drug catalogue*

Patient	Date	Medication	Dose	Approval
A1234	12/01/07	Acetaminophen	400 mg	ZYX
A9876	12/01/07	Aspirin	250 mg	ABC
. . .	. . .	. . .	. . .	. . .

(b) *Hospital drug record*

One important strength of metadata is its ability to capture semantics. Even in simple examples we encounter some of the difficulties with database semantics that metadata is meant to address. The data element named Drug represents a concept that has synonyms; in the case of the National Cancer Institute (NCI) in the USA, the preferred term is *medication*, even though many medical personnel will still use *drug* to refer to these objects, even in technical reports. That is why we speak of 'concepts' rather than 'terms' or 'data values' in performing reasoning. This is also the attitude taken by terminologists (see the standards ISO, 2000a; ISO, 2000b) as well as the metadata registration community (ISO/IEC, 2003; ISO/IEC, 2004a; ISO/IEC, 2004b).

Also evident is an important limitation of metadata in its usual sense. In order to answer the above query, some knowledge missing from the written description is needed – namely, that both aspirin and acetaminophen are kinds of analgesics and both should be considered as analgesics. While this seems 'obvious' to the reader, it is actually beyond the reasoning ability of most metadata systems, and hence provides a clear illustration of where we need more knowledge than represented by just the metadata.

The form of metadata varies, but this paper assumes content based on international standards (ISO/IEC, 2003; ISO/IEC, 2004a; ISO/IEC, 2004b). Such descriptions of metadata are already in wide use for some significant metadata repositories, such as that of the NCI (Coronado et al., 2004).

The next subsection shows how a knowledge representation (in this case, Common Logic) can provide a basis for the reasoning needed to answer this section's query (and others).

2.2 Common Logic

In order to fully understand the work described herein, a brief description of Common Logic is necessary; for the complete standard see ISO/IEC (2007). Common Logic (CL) is a standardisation of first-order logic whereby sentences in the logic can be exchanged between systems while preserving its semantics. CL's semantics are those of formal model theory (Hodges, 1997; Huth and Ryan, 2004), where symbols are assumed to have a consistent interpretation within any given model. CL specifies an abstract semantics, to which three distinct standardised dialects are each semantically equivalent.

The CL standard does not specify how to perform reasoning; however, since it provides a clear model theoretic interpretation to any CL formula, reasoning from a CL representation follows directly from well-known logic rules and model theory. One key feature of that theory is that symbols in the representation denote individuals and relationships in some universe of discourse. A primary metadata standard (ISO/IEC 11179) recognises the importance of a conceptual domain relative to particular data values, we can use concepts in the conceptual domain on which to base our Common Logic interpretations. For example, the data values for 'drug' and 'medicine' both are based on the same conceptual domain, so that they share one universe of discourse and therefore the symbol 'acetaminophen' means the same individual for both of them.

Strictly speaking, since CL itself has an abstract syntax and semantics, we would be more accurate expressing our conceptual models using that abstract syntax. This sometimes becomes cumbersome, so it is easier for us to illustrate CL with examples written in one of the concrete dialects defined in the CL annexes. This paper will therefore use examples written in the Common Logic Interchange Format (CLIF) defined in Annex A of ISO/IEC (2007). We could represent the examples using any one of the three dialects and mix them up any way we would like – this would further demonstrate the power and interoperability of CL, but would probably confuse most readers.

The next section outlines the basic approach for representing relationships and for creating metadata constraint models at the conceptual level.

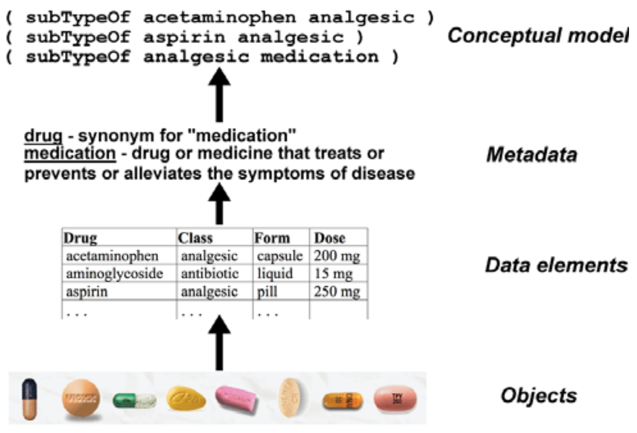
3 Our approach

This section describes the overall approach to both representing and reasoning about metadata using Common Logic. We explain how metadata constraints and relationships fit into the modelling scheme, then we describe our approach for creating and representing a conceptual model from its corresponding metadata. We then present some basic notions about reasoning.

3.1 Representing metadata relationships

The scheme in Figure 2 suggests the general structure of modelling metadata relationships and domain constraints. The diagram illustrates how we are interested in relationships that exist beyond the metadata descriptions themselves. The picture is best understood from the bottom up. At the bottom are shown the objects in a domain of interest; namely, medicines. (We will discuss later whether there are other levels or points of view from which we can consider these drugs.) Information about these medications is represented by a database as in Figure 1(a).

Figure 2 Knowledge structure for metadata relationships (see online version for colours)



```
( subTypeOf acetaminophen analgesic )
( subTypeOf aspirin analgesic )
( subTypeOf analgesic medication )
```

drug - synonym for "medication"
medication - drug or medicine that treats or prevents or alleviates the symptoms of disease

Drug	Class	Form	Dose
acetaminophen	analgesic	capsule	200 mg
aminoglycoside	antibiotic	liquid	15 mg
aspirin	analgesic	pill	250 mg
...	...	...	

As we noted in the previous section, while metadata does provide some semantics to help understand the content along with the conceptual domain of a database, we require something beyond metadata that will provide formal semantics at the level of understanding relationships in the conceptual domain. Common Logic is one representation for conceptual models that can provide some of these semantics.

This paper describes an approach in which CL formulae are included along with metadata to model a conceptual domain. The metadata itself forms a model whose understanding does not require any additional knowledge, in the sense that practitioners with no CL expertise already know how to use metadata effectively. It can also be argued that practitioners already 'know' some of what we are about to describe; again, we do not claim to be adding any new knowledge, only that we are adding existing informal knowledge in a formal way suitable for automation and formal reasoning.

Common Logic supports all of first-order logic with quantification. We can therefore describe basic logic reasoning and deduction rules. We begin illustrating Common Logic by showing a simple rule. One typical relationship between classes is the notion of one category being a subtype (specialisation, subclass, subconcept, etc.) of another.

To express the rule that the subtype relationship is transitive, we can state the formula in CL.1. This formula says that if s1 is a subtype of s2 and s2 is a subtype of s3, then s1 is a subtype of s3.

```
(forall ( (s1 s2 s3) (implies
  (and (subType s1 s2) (subType s2 s3))          CL.1
    ( subType s1 s3 )  )
```

To express the rule that an individual of a given type is also an individual of any of its supertypes, see CL.2.

```
(forall ((t1 t2 x) (implies
  (and (type t1 x) (subType t1 t2))              CL.2
  (type t2 x)))
```

We will use these two formulas later.

3.2 *Associating metadata with conceptual models*

The above discussion gives some constraints that a conceptual model would contain, but still lacks content – namely, specifications of actual instances of things such as types, objects, individuals etc. That is, we must develop a model of a database's conceptual domain and then populate it with objects of interest. This requires an active system that is able to use a conceptual model to acquire individual attributes (from a database or other information source) about each of the objects. This section describes one practical strategy for populating such a conceptual model. Some of this work appears in Delugach (2003), using conceptual graphs, which is one of the dialects to express CL formulae.

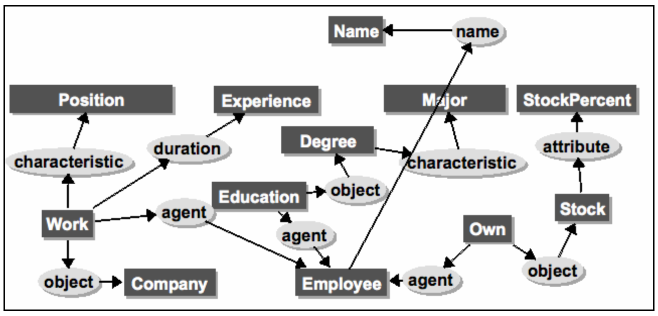
We use a typical source and model such as those in Figure 3, where Figure 3(a) is the data source and Figure 3(b) shows the conceptual model (in conceptual graph form, one of the dialects of CL). This figure is a clear illustration of the knowledge that must be included in order to support reasoning: some 'obvious' concepts are missing from the data source, such as the fact that these are people working for a particular organisation, the fact that each record represents an employee etc. Most of the relationships among the data are implicit ones, easily assumed by (human) users of the source, but un-represented for purposes of automated retrieval and processing.

Figure 3(c) shows how active agents (here represented by the actor **lookup** in a conceptual graph) are used to populate the conceptual model with instance attributes from the data source. The populated model is thereby usable for inference and reasoning.

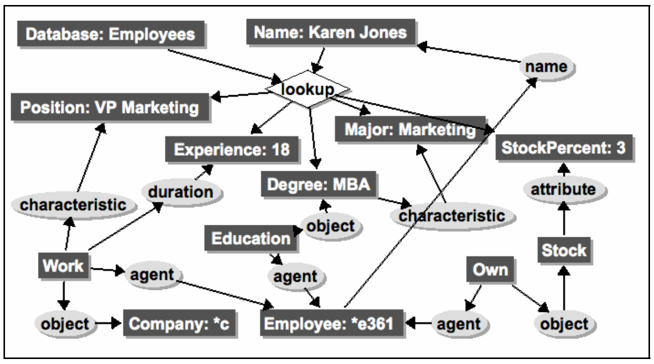
Figure 3 Data source and conceptual model

Name	Position	Yrs Experience	Degree	Major	Percent Stock
Karen Jones	VP Marketing	18	MBA	Marketing	3
Kevin Smith	VP Technology	12	MSE	Engineering	4
Keith Williams	VP Finance	15	BS	Accounting	3
É	É	É	É	É	É

(a)



(b)



(c)

Note that the **Employee** concept has the identifier ***ed361**. This is an arbitrary symbol, assigned by the populating process, in order to denote this particular employee (the one named 'Karen Jones'). Each record of the data source will represent a different employee, each one having its own arbitrary (but unique) symbol that denotes it for reasoning purposes. Populating the model for each employee will therefore result in a new copy of the graph shown in Figure 3(c), but with its distinct attribute values denoted in the concepts. The populated conceptual model will consist of many replicated graphs, all structurally identical because the relationships among the record fields are the same for every employee.

Such a conceptual model would of course be very large. A reasonable optimisation would be to defer populating the model with instances until an instance is needed; e.g., as the antecedent to an implication. For some general searches, however, all instances would populate the model. For this paper such considerations will be avoided; we assume that an automated agent will have been developed with sufficient optimisation strategies to make this approach feasible.

There may be additional constraints to add to the conceptual model; e.g., relationships between a manager and the members of their departments, team relationships among groups of employees, and so on. If these happen to be contained in other data sources (a plausible assumption) then a similar procedure can be used to further populate the model with those occurrences.

There is one last point to make. For practical systems, we are not really interested in the *record* of this employee – we are interested in the *actual* employee, whose information is captured in the data source. Associating the symbol ***ed361** with this actual employee is of key importance for practical systems, but is beyond the scope of the formal modelling process.

This section described how conceptual models can be created which capture knowledge about objects of interest, as well as their semantics. Once this is accomplished, we have the capability to reason about these models, as shown in the rest of the paper.

3.3 Reasoning with Common Logic

The CL standard itself is neutral with respect to reasoning, in the sense that it does not require or describe any particular reasoning strategy or algorithms. This is in keeping with its primary purpose which is to serve as an interchange representation for logical formulae. It is up to knowledge users how they intend to reason with them, of course, while still requiring that they adhere to the model theoretic foundation of the representations.

As a result, reasoning with CL is usually performed with conventional first-order logic theorem proving rules which are obviously not in the scope of this paper; see Huth and Ryan (2004), Hayes (1985), Kalish et al. (1980), Israel (1983), Sowa (1993) for some current techniques, and Roberts (1973), Whitehead and Russell (1927) for some historical background.

One novel feature that is added to logic representations is a facility for interacting with agents. This feature is included in conceptual graphs (Sowa, 1984), the basis for Annex B of ISO/IEC (2007), through the use of actor nodes in an active knowledge system as described in Delugach (2003, 2006, 2008). These can provide 'triggers' so that certain real-world activities can invoke automated reasoning and retrieval processes within a knowledge system. The extended example in Section 0 gives an illustration of how this works in a practical system.

4 Examples

The purpose of this section (and the main purpose of the paper) is to illustrate the usefulness of a formal semantic representation for metadata constraints. It is our aim that the examples clearly show the kind of knowledge that we humans would like to automatically apply in solving problems and understanding human activity.

The reasoning shown in these illustrations is not esoteric or complicated. On the contrary, these simple examples are intended to show that, with respect to existing metadata representations, our current reasoning ability is quite limited; even apparently simple deductions are beyond our current capabilities. Common Logic is one representation that makes a step toward supporting such inferences.

We will show some Common Logic descriptions of the metadata required for these examples, but the main point is that they all require a knowledge representation that captures semantics. We do not argue that Common Logic is the only way to accomplish this, just that it is sufficiently expressive for these purposes.

4.1 Generalisation from an ontology

The previous sections' examples already illustrated common way of organising information by assigning a type to a given object and then arrange the types in a hierarchy. In the model of metadata relationships for medications, we would also include CL.3.

```
( subType acetaminophen analgesic )
( subType aspirin analgesic )                    CL.3
( subType analgesic medication )
```

Note that in one sense, the example is a *metadata relationship* involving *descriptive metadata*, in the sense that we are talking about a shared meaning between the concepts **acetaminophen** and **aspirin**; namely, that they share a common super-concept **analgesic**. It could also be considered *structural metadata*, since it involves the generalisation hierarchy structure.

Common Logic does not support types directly, but there are two common conventions; unfortunately, modellers must choose one of them. One is simply to use monadic predicates; e.g. **(aspirin d1)** associates the type aspirin with the name **d1**. Another way is to explicitly denote the type relationship by saying **(type aspirin d1)**. Either of these could be used to denote that **d1**'s type is aspirin. Since CL's semantics imply that **d1** is a symbol for some individual in the domain being described, it would only be useful if it also appeared somewhere else in a model.

To model the hospital drug records from Figure 1(a), we would specify CL.4.

```
(type drug-administration d1)
(recipient d1 A1234) (type patient A1234)
(date-administered d1 12/01/77)
(type aspirin d1)  (dose d1 (400 mg) )
                                                CL.4
(type drug-administration d2)
(recipient d2 A1234) (type patient A9876)
(date-administered d2 12/01/77)
(type acetaminophen d2) (dose d2 (250 mg) )
```

Note that each instance of administering a drug has its own symbol **d1**, **d2** and so on. This is how data must be specified in logic terms, because each time a drug is administered and a record produced, there is an identifiable individual that we must denote.

Another way to use an ontology for reasoning with metadata is to aggregate individual concepts based on their supertypes. A good example is to consider a doctor whose patient has an infection, and an antibiotic is indicated. A database will have data about different drugs and their properties; the metadata will reflect them in a data value-independent way. The doctor may be familiar with some of the individual drugs, but wants to choose the best one for the patient. Metadata ontologies allow a querying system to gather all possible subtypes of *antibiotic drug* and present their properties to the doctor for further decision making.

4.2 Constraints using structural metadata

CL can also express constraints among metadata that represent knowledge that lies beyond the actual content; in this case, we show how the origin of the data affects how we might reason about it.

Suppose we have the model CL.5 from a database, where sample **s1** has assigned two different values from two different chemical tests.

```
(sample-pH s1
(assign test-12-pH
  (value 6 ('5 Jul 2007' Harry litmus ) )
                                                CL.5
(assign test-12-pH
  (value 5.5 ('7 Jul 2007' Jane ph-meter) )
)
```

In registering metadata for other users, we would like to specify our own axioms for what value to use when there may be more than one assigned value. CL.6 shows how such an axiom might be framed in CL semantics:

```
(forall (x val a1 a2 a3) (implies
(assign a1 (value val (a2 a3 ph-meter) ) )      CL.6
    val ) )
```

The above axiom states that if any value's source is a pH meter test, then that is the value to be used. We could also provide rules that tell what do if you only have the litmus test.

4.3 An extended example

This section describes a more complete example of the kind of powerful inferences that can be supported through a knowledge-based representation of concepts to support reasoning with metadata.[1] The domain of the example is a simple one, yet it demonstrates the amount of knowledge that must be brought to bear in using data from multiple data sources, and therefore shows the power of metadata constraints so that we can perform these kinds of inferences automatically.

The example situation is where a doctor prescribes some medication to a patient. Will the patient have an allergic reaction? In most real-world environments, this question is answered in an ad-hoc way, by comparing a patient's stated medical history with a list of drugs. Using automated reasoning with semantically based metadata, powerful queries can be automatically constructed and answered.

The example scenario assumes the existence of an automated reasoning agent that will traverse the models making inferences and satisfying (or not) logical assertions along the way. The initial logic rule for the agent's reasoning is shown in CL.7, so the agent attempts to satisfy the consequent **(allergic d person)**. The rule just says that if there's a prescription for a medication and the recipient of the prescription has an adverse reaction to the drug, then the person is allergic.

```
(forall (p person d) (implies
    ( (type prescription p) and
    (object p d) and
    (type medication d) and                     CL.7
    (recipient p person) and
    (adverse-reaction d person) )
        (allergic d person)
      ) )
```

The function **adverse-reaction** (specified in CL.11 below) is obtained from metadata related to medical terminology. All of the specifications in this section are shown as they would be obtained either automatically from the metadata or else provided by the information source providers with the metadata. These would already have been developed and would be available to an automated system before the scenario begins. In the interest of clarity, however, each model will be introduced at the point it would be accessed.

The scenario begins with a prescription:

John A. Doe A0123456   500 mg Prevpac bid

One way to describe the prescription is by the conceptual model in CL.8, which has been populated as described above in Section 0.

```
(type prescription p1)
(recipient p1 person1 )
(type patient person1 )
(name person1 'John A. Doe')                    CL.8
(patientID person1 A0123456)
(object p1 d1)
(type Prevpac d1)  (dose d1 (500 mg) )
(frequency d1 bid)
```

The model in CL.8 denotes the following. This particular instance of a prescription is denoted by the symbol **`p1`**. The recipient of the prescription is denoted by the symbol **`person1`**. The recipient is of type patient, has the name 'John A. Doe' and the patient ID A0123456. The object (i.e. the thing being prescribed) in **`p1`** has the symbol **`d1`** and is of type Prevpac; its dose is 500 mg and frequency is 'bid' (twice daily).

Since not all prescriptions are for drugs (e.g. some could also be for a medical device), the triggered agent first looks for what type prescription this is ('Prevpac'), and then consults a drug ontology (partially shown in CL.9) to determine its type. Additional information beyond this example is shown to remind the reader that such ontologies contain a rich and large set of knowledge.

```
(subType acetaminophen analgesic )
(subType aspirin analgesic )
(subType analgesic medication )
(subType amoxicillin penicillin-antibiotic)
(subType penicillin-antibiotic antibiotic)
(subtype antibiotic medication)
(subType nafcillin penicillin-antibiotic)
(subType clarithromycin
       macrolide-antibiotic)
(subType macrolide-antibiotic antibiotic)      CL.9
(subType macrolide-antibiotic
       protein-synthesis-inhibitor)
(subType protein-synthesis-inhibitor
       medication)
(subType lansoprazole proton-pump-inhibitor)
(subType proton-pump-inhibitor
       anti-ulcer-agent)
(subType anti-ulcer-agent adjuvant)
(subType adjuvant medication)
```

The agent first tests the logical query **`(type medication Prevpac)`** for its truth or falsity with respect to the existing conceptual models. Note that Prevpac does not appear anywhere in CL.9, since these are generic-name drugs, and it happens that Prevpac is a trade name. The automated agent looks up Prevpac in a composition list, as shown in CL.10. This model would be generated automatically from the metadata of a database containing drug components.

```
(part Amoxil amoxicillin)
(part Co-amoxiclav
   amoxicillin Clavulanic-acid)
(part Prevpac                                  CL.10
   amoxicillin clarithromycin lansoprazole)
(part Tylenol acetaminophen)
(part Panadol paracetamol)
```

Discovering that Prevpac is actually a combination of three things, amoxicillin, clarithromycin and lansoprazole (but still not identified as medications), the agent applies the type rule shown in CL.2 to each of the components. Each can therefore be identified as a medication from CL.9, so the agent now continues its reasoning steps from CL.7 to determine whether the particular patient (denoted as **`p1`** earlier) is allergic to the particular drug **`d1`**.

Satisfying the (sub)formula **`(adverse-reaction d person)`** in CL.7 is the crux of this example. For our purposes, we will restrict our modelling to a physician's explicit diagnosis of an adverse reaction, so the reasoning will focus mostly on the drug part of the formula.

Assuming that the patient's records are accessible through a standards-compliant information source, the automated agent may make use of standard codes, such as ICD-9-CM, which is a recent version of the International Classification of Diseases, used by the World Health Organization (WHO) for its data collection and reporting functions. Many hospitals and medical practitioners use the codes in their everyday recording of patient information.

We will assume that this patient's records are available, including the ICD codes. The agent has been given a simple model for determining past adverse reactions: If the patient record contains a diagnosis of ICD-9-CM code 995.2, this denotes 'unspecified adverse effect of drug, medicinal and biologic substances'. Note that this code does not specifically mean 'allergy' – any adverse reaction will be included under this code by the reasoning agent.

The SNOMED terminology for disorders has a concept 'Allergic reaction to drug' (code 416093006) which is mapped to the ICD as 995.29, meaning that it is a subtype of the 'unspecified adverse effect'. This concept has the relationship 'causative agent' to a drug or medication.

Such codes must be provided manually (hard-coded) by a domain specialist when the agent was originally specified; having the agent determine it through reasoning over all the terms in the ICD is beyond the scope of the current work, but is part of our long-term goal. Since the adverse reactions are concepts, a richer domain model of medication and patient behaviour could provide real meaning to 'adverse reaction', but for the present, it is simply one kind of response to a causative agent. In fact, substituting the code for some other response would allow the same agent to determine whether a given medication would cause that other response.

CL.11 shows the basis for determining if there is an adverse reaction and what caused it. This is a *domain constraint* in the sense described at the beginning of the paper, because it formally specifies what is an adverse reaction for the agent's formula in CL.7.

```
(forall (drug prev patient) (implies
    ( (record patient r ) and
    (part r (diagnosis d 995.2)) and
    (part d (causative-agent prev) ) and      CL.11
    (similar prev drug ) )
        (adverse-reaction drug patient)
        ) )
```

The formula in CL.11 depends on a function named 'similar'. Research on similarity measures is abundant (e.g. Delugach, 1993; Landauer et al., 1998; Maedche and Staab, 2002). Particular strategies for measuring similarity among metadata concepts are still under investigation, but we could start by adopting a simple measure based on the subtype hierarchy: the closer together are the two drugs in the ontology (i.e. the fewer subtype relationships needing to be traversed between them), the more similar they are; if the similarity is below some threshold, the function will return true. This is obviously simplistic when dealing with a large number of drugs, but for the moment we will assume that `(similar nafcillin amoxicillin)` returns true, since both drugs are ontological 'siblings' with respect to penicillin-antibiotic, as can be inferred by the agent from CL.9.

```
(subType nafcillin penicillin-antibiotic)
(subType amoxicillin penicillin-antibiotic)
```

Note that this model specifies that any drug related to the one which caused the previous reaction will also cause the same reaction. That is, if someone had a reaction to nafcillin, then it is assumed there would be a reaction to amoxicillin. In practice, a richer conceptual model would be needed to more accurately determine this, if indeed such determination can be made without further tests or additional patient information.

Another simplification of the example's model is that the composition model does not specify how much of each ingredient makes up the whole. The agent therefore does not perform any reasoning about the amount of the dose or the form of the drug, so that a 1 mg portion of amoxicillin is considered the same as a 1000 mg portion.

This extended example shows how an automated reasoning agent derives the results we seek, by proving (or not) formulae specified by metadata constraints. A failure to prove the formula means that (within the given information of course) the condition does not hold.

5 Discussion

This section discusses some of the limitations and problems identified with the formal logic and semantics approach.

5.1 Declarative style issues

Aside from the use of multiple dialects, there are a variety of ways in which the same logical formula can be represented, each with exactly the same expressive power. For example, to show that a door is part of a house, we could write either of these (as well as others):

```
(part door house)
(part house door)
```

Which one to choose is a matter of convention. Common Logic does not prescribe any particular order, and in fact one of the strengths of the standard is that specifiers can use whatever style is convenient. Of course, interpreting the first one while assuming the style of the second one (i.e. interpreting it to mean that a house is part of a door) is clearly an error.

Declarative style can mean other things as well, e.g. we can either say `(subtype a b)` or `(supertype b a)`. For efficiency in some systems, it may also be useful to include both of these. No logic formalism by itself can eliminate the need for modellers to make these choices.

Specifiers of metadata constraints must be explicit about the styles they use and ensure that any interpreters (either human or automated) use the same interpretation. When systems use differing styles, Common Logic's abstract semantics allow specifying a translation between the two, as long as interpreters are required to use them.

5.2 Mechanisms of database access

The approach and examples described herein combine both logical reasoning and active querying of existing databases. Since we have focused on the logical reasoning part of this process, we do not discuss details related to the active querying. This must be addressed in future work, because the number of possible information sources is expected to be quite large and it is therefore not feasible to assume that all of them have been previously identified and integrated into a single logical system. They must be found and modelled as they are needed.

A key ingredient to accessing existing databases is the ability to provide interpretations for symbols in a formal logical model, in the sense of mapping to whatever 'real world' objects and relationships are under consideration. Given these interpretations, it will be possible for queries to instantiate information from external sources and perform reasoning on these instantiated symbols.

5.3 Knowledge-based system limitations

There are other major issues in the approach described that should be already clear to the reader who is familiar with knowledge-based systems. Indeed, these issues are ever-present in developing and validating such an approach, as summarised in textbooks such as Brachman and Levesque (2004). We briefly include them here as reminders.

Synonyms: The issue of synonyms in naming schemes is well known and a number of solutions have been effective. For example, the same popular analgesic *para*-acetylaminophenol is called *paracetamol* internationally but it is referred to as *acetaminophen* in the USA. In the case of pure synonyms, where two or more terms refer to the identical concept, it is straightforward (though less efficient) to use lists of synonyms. Note it is not feasible to always expect a single standard nomenclature in a system or even in a single database – there are still many information domains where multiple standards exist, and sometimes an updated version of a standard produces synonyms or requires mappings between terms. In some cases, we can envision that using metadata constraints based on concepts instead of terms will tend to 'smooth out' differences in terminology.

Validation: As in any knowledge-based system, especially where human analysts are involved, validating the constraints is a crucial part of the process. Internal consistency among a set of constraints can be established by various theorem-proving techniques, at a computational cost that is well documented, as summarised in Brachman and Levesque (2004) and Huth and Ryan (2004).

Cost vs. benefit analysis: The benefit of using metadata constraints and relationships has been shown in the previous section. Users can make queries on conceptual relationships, which are then instantiated by database access based on its metadata. But acquiring and validating metadata constraints involves time and effort on the part of knowledge modellers and domain experts. That cost must be balanced against the potential benefits: an ever-present issue in knowledge-based systems.

It may be argued that there are some broad application areas for knowledge-intensive systems (national security, nuclear materials, epidemiology) where the benefit can be considered great enough to balance even substantial costs. For such obviously critical systems, however, a collection of special-purpose or customised approaches might be just as (or more) effective. For the vast majority of databases, cost will be an important consideration.

Granularity: A set of well-known problems arises from data that are at distinctly differing levels of detail. For example, one database may deal with drugs in terms of their general class (e.g. NSAID, analgesic) whereas another may deal with drugs in terms of their point of origin, method of preparation and manufacturer.

Knowledge modelling: Finally, a key issue is left un-addressed, not just because it is beyond the scope of this work, but also because it is centred in the very fact that there are many more useful queries that we would want to be automatically performed besides the one shown in the extended example. Yes, automatically determining potential allergic reactions is a valuable thing to do. But an allergy is just one possible reaction to a drug – there may be counter indications (conditions under which the drug may be dangerous rather than beneficial). There may also be situations where a less expensive drug should be sought if it can be as effective as the more expensive one. Each of these considerations would require different data sources and therefore access to different concepts and metadata.

6 Related work

The field of metadata and ontology is a wide one, as evidenced by the range of topics discussed in this journal. We briefly mention other approaches to this problem and summarise their relationship to the current work.

One of the most important modelling representations used today is that of UML (Booch et al., 2005), whose associated Object Constraint Language (OCL) (Warmer and Kleppe, 1998) is particularly relevant to this work. OCL would seem a reasonable candidate for expressing metadata constraints and indeed has been used in some limited ways already (e.g. Demuth et al., 2001). The reason for choosing Common Logic is both its foundation in formal logic and its solid relationship to model theory, both of which are somewhat lacking in OCL [see Vaziri and Jackson (2000) for a good overview]. In fact, the OMG's Ontology Definition Metamodel uses Common Logic as one of its formalisms.

There are some other widely used ontological representations, e.g. RDF (Brickley and Guha, 2004) and OWL (Antoniou and van Harmelen, 2004; Smith et al., 2004), that also provide some useful features related to the semantics of constraints with respect to metadata. Since these languages are primarily intended for ontology interchange, they clearly lack some of the expressiveness desired for knowledge representation and reasoning. Indeed, one criticism of full first-order logic as a practical representation centres on its computational complexity which the less expressive formalisms seek to avoid (Baader et al., 2004).

7 Conclusion and future work

We have shown how Common Logic can be a valuable way to represent and reason about metadata, and therefore can be used to develop 'smart' queries of information sources. This paper shows only a first step toward full utilisation of CL's capabilities. As a new standard, Common Logic has yet to gain widespread use in the metadata and information systems community.

While CL is certainly not the only representation that can support these capabilities, it is clearly a candidate for further study; future work will develop automated tool support and lead to even more interesting applications.

Acknowledgements

Members of ISO/IEC/SC32/WG2 have been very helpful in understanding these ideas. Bruce Bargmeyer and Kevin Keck helped in relating Common Logic to the existing metadata standards. Denise Warzel clearly explained the purpose of the NCI terminology browsers and how to use them.

References

Antoniou, G. and van Harmelen, F. (2004) 'Web Ontology Language: OWL', in Staab, S. and Studer, R. (Eds): *Handbook on Ontologies*, Springer, Berlin, pp.67–92.

Arens, Y., Knoblock, C.A. and Shen, W-M. (1996) 'Query reformulation for dynamic information integration', *Journal of Intelligent Information Systems*, Vol. 6, pp.99–130.

Baader, F., Horrocks, I. and Sattler, U. (2004) 'Description logics', in Staab, S. and Studer, R. (Eds): *Handbook on Ontologies*, Springer, Berlin, pp.3–28.

Bertino, E., Zarri, G.P. and Catania, B. (2001) *Intelligent Database Systems*, Addison-Wesley.

Booch, G., Rumbaugh, J. and Jacobsen, I. (2005) *Unified Modeling Language User Guide (2/e)*, Addison-Wesley, Reading, MA USA.

Brachman, R.J. and Levesque, H.J. (2004) *Knowledge Representation and Reasoning*, Morgan Kaufmann, San Francisco.

Brickley, D. and Guha, R.V. (2004) 'RDF Vocabulary Description Language 1.0: RDF Schema', 10 February 2004.

Common Logic Standard portal – http://common-logic.org

Coronado, S.D., Haber, M.W., Sioutos, N., Tuttle, M.S. and Wright, L.W. (2004) 'NCI thesaurus: using science-based terminology to integrate cancer research results', in Fieschi, M., Coiera, E. and Li, Y-C.J. (Eds): *Proceedings of 11th World Congress on Medical Informatics*, IOS Press, San Francisco, CA, USA, pp.33–37.

Delugach, H.S. (1992) 'Analyzing multiple views of software requirements', in Eklund, P., Nagle, T., Nagle, J. and Gerholz, L. (Eds): *Conceptual Structures: Current Research and Practice*, Ellis Horwood, pp.391–410.

Delugach, H.S. (1993) 'An exploration into semantic distance', in Pfeiffer, H.D. and Nagle, T.E. (Eds): *Conceptual Structures: Theory and Implementation*, Vol. 754, Springer-Verlag, pp.119–124.

Delugach, H.S. (1996) 'An approach to conceptual feedback in multiple viewed software requirements modeling', *Viewpoints 96: International Workshop on Multiple Perspectives in Software Development*, San Francisco, pp.242–246.

Delugach, H.S. (2003) 'Towards building active knowledge systems with conceptual graphs', in de Moor, A., Lex, W. and Ganter, B. (Eds): *Conceptual Structures for Knowledge Creation and Communication: 11th International Conference on Conceptual Structures (ICCS 2003)*, Vol. LNAI 2746, Springer-Verlag, Berlin, pp.296–308.

Delugach, H.S. (2006) 'Active knowledge systems for the pragmatic web', in Schoop, M., de Moor, A. and Dietz, J. (Eds): *Pragmatic Web: Proceedings of the First International Conference on the Pragmatic Web*, Vol. P-39, Gesellschaft für Informatik, Stuttgart, Germany, pp.67–80.

Delugach, H.S. (2008) 'Active knowledge systems using conceptual graphs', in Hitzler, P. and Schärfe, H. (Eds): *Conceptual Structures in Practice*, Chapman and Hall/CRC Press.

Demuth, B., Hussmann, H. and Loecher, S. (2001) 'OCL as a specification language for business rules in database applications', in *4th International Conference on the Unified Modeling Language: Modeling Languages, Concepts and Tools*, pp.104–117.

Digital Library Federation – http://www.diglib.org/dlfhomepage.htm

Hayes, P.J. (1985) 'The logic of frames', in Brachman, R.J. and Levesque, H.J. (Eds): *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA.

Hodges, W. (1997) *A Shorter Model Theory*, Cambridge University Press, Cambridge, UK.

Huth, M. and Ryan, M. (2004) *Logic in Computer Science: Modelling and Reasoning About Systems*, Cambridge University Press, Cambridge, UK.

ISO (2000a) 'ISO 704:2000 – Terminology work – Principles and methods', International Organization for Standardization, Geneva, Switzerland.

ISO (2000b) 'ISO 1087-1:2000 – Terminology work – Vocabulary. Part 1: Theory and application', International Organization for Standardization, Geneva, Switzerland.

ISO/IEC (2003) 'ISO/IEC 11179-3:2004 – Information technology – Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes', International Organization for Standardization, Geneva, Switzerland.

ISO/IEC (2004a) 'ISO/IEC 11179-4:2004 – Information technology – Metadata registries (MDR) – Part 4: Formulation of data definitions', International Organization for Standardization, Geneva, Switzerland.

ISO/IEC (2004b) 'ISO/IEC 11179-1:2004 – Information technology – Metadata registries (MDR) – Part 1: Framework', International Organization for Standardization, Geneva, Switzerland.

ISO/IEC (2007) 'ISO/IEC 24707:2007 – Information technology – Common Logic (CL) – A framework for a family of logic-based languages', International Organization for Standardization, Geneva, Switzerland.

Israel, D.J. (1983) 'The role of logic in knowledge representation', *Computer*, Vol. 16, pp.37–41.

Kalish, D., Montague, R. and Mar, G. (1980) *Logic: Techniques of Formal Reasoning*, Harcourt Brace Jovanovich, New York.

Landauer, T.K., Foltz, P.W. and Laham, D. (1998) 'Introduction to latent semantic analysis', *Discourse Processes*, Vol. 25, pp.259–284.

Maedche, A. and Staab, S. (2002) 'Measuring similarity between ontologies', in *Proceedings of European Conference on Knowledge Acquisition and Management – EKAW-2002*, Springer, Madrid, Spain, pp.251–263.

National Cancer Institute Terminology Browser – http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do

OMG Ontology Definition Metamodel – http://www.omg.org/technology/documents/modeling_spec_catalog.htm

Roberts, D.D. (1973) *The Existential Graphs of Charles S. Peirce*, Mouton, The Hague.

Sowa, J.F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Mass.

Sowa, J.F. (1993) 'Relating diagrams to logic', in Sowa, J.F. (Ed.): *Conceptual Graphs for Knowledge Representation*, Springer-Verlag, Berlin, pp.1–35.

Sowa, J.F. (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole.

Smith, M.K., Welty, C. and McGuinness, D.L. (2004) 'OWL Web Ontology Language Guide', Vol. 2006, W3C.

Vaziri, M. and Jackson, D. (2000) 'Some shortcomings of OCL, the object constraint language of UML', in *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 34'00)*, IEEE Computer Society.

Warmer, J. and Kleppe, A. (1998) *Object Constraint Language: Precise Modeling with UML*, Addison-Wesley.

Whitehead, A.N. and Russell, B. (1927) *Principia Mathematica*, 2nd ed., Cambridge University Press, Cambridge, UK.

Note

1 This example is taken from a talk given by Bruce Bargmeyer at the 9th Open Forum for Metadata Registries, Kobe, Japan, 2006.