Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

SATAC Knowledge Representation and Automated Reasoning with JC3IEDM

Task 4—Knowledge Representation Capabilities and Limitations of JC3IEDM and P-JC3IEDM

Hugues Demers
Jean-Rémi Duquet
Lockheed Martin Canada

Prépared by:
Lockheed Martin Canada
6111 ave. Royalmount,
Montréal, Québec, H4P 1K6

Project Manager: Éric Dorion 418-844-4000 Ext. 4257
Contract Number:W7701-061941/001/QCL
Contract Scientific Authority: Éric Dorion 418-844-4000 Ext. 4257

Defence R&D Canada – Valcartier
Contract Report
DRDC Valcartier CR 2008-255
September 2008

Canada

SATAC Knowledge Representation and Automated Reasoning with JC3IEDM

Task 4—Knowledge Representation Capabilities and Limitations of JC3IEDM and P-JC3IEDM

Hugues Demers
Jean-Rémi Duquet
Lockheed Martin Canada

Prepared by:

Lockheed Martin Canada
6111 ave. Royalmount, Montréal, Québec, H4P 1K6

Project Manager: Éric Dorion 418-844-4000 Ext. 4257
Contract Number: W7701-061941/001/QCL
Contract Scientific Authority: Éric Dorion 418-844-4000 Ext. 4257

Defence R&D Canada – Valcartier

Contract Report

DRDC Valcartier CR 2008-255

September 2008

Approved by

Éric Dorion
Scientific Authority

Approved for release by

Christian Carrier
Head/Document Review Panel

This document results from S&T work conducted under DRDC project 12of (Situation Analysis for the Tactical Army Commander).

Abstract

This report describes the knowledge representation languages that are used by the *Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM)* and the *Protégé-JC3IEDM (P-JC3IEDM)*. It then describes the semantic expressiveness of JC3IEDM and P-JC3IEDM and how these can be used to support automated reasoning with respect to reasoning examples that were developed in Task 3 of this contract. [1]

Résumé

Ce rapport décrit les langages de représentation de connaissance utilisés par le *Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM)* et le *Protégé-JC3IEDM (P-JC3IEDM)*. Il décrit ensuite l'expressivité sémantique de JC3IEDM et P-JC3IEDM et comment ils peuvent être utilisés pour supporter le raisonnement automatique pour les exemples de raisonnement développés dans la tâche 3 de ce contrat. [1]

This page intentionally left blank.

Executive summary

SATAC Knowledge Representation and Automated Reasoning with JC3IEDM

Hugues Demers, Jean-Rémi Duquet; DRDC Valcartier CR 2008-255; Defence R&D Canada – Valcartier; September 2008.

Background: The purpose of this document is to describe the findings of Task 4 of the *Knowledge representation and Automated Reasoning (KAR) with the JC3IEDM* contract (W7701-061941/001/QCL). This contract was realized under the auspices of the *Situation Analysis for the Tactical Army Commander (SATAC)* Advanced Research Project (ARP). The overarching goal of SATAC is to provide the Canadian Army with the means to build an automated reasoning capability to support Situation Analysis (SA), a process by which the human gains Situation Awareness (SAW). Within the project, the aim of the SATAC KAR contract is to evaluate the suitability of the JC3IEDM and P-JC3IEDM to support a knowledge-based inference mechanism in support to the Tactical Army Commander (TAC) decision-making process. This will be done through a combination of literature surveys, analyses using formal ontological engineering tools and approaches, and the realization of a prototype demonstrator of a knowledge-based Situation Analysis Support System (SASS).

Results: This report show how some military situation elements that were developed in Task 3 can be encoded in the JC3IEDM and P-JC3IEDM. The difficulties in doing so with the JC3IEDM are already known due to its relatively high normalization but the task proves to be even more difficult with the OWL version (P-JC3IEDM). The reports then proceeds with proposing some modifications to P-JC3IEDM in order to facilitate this task.

Significance: The discoveries of this report shows that encoding simple knowledge into P-JC3IEDM, as would be required by the Army staff over a deployed platform, is not a trivial task. Although it is also difficult with the JC3IEDM, it proves to be almost impossible for certain cases with the OWL version. We ascertain that this is due to the nature of P-JC3IEDM, being an automated translation of the JC3IEDM in its OWL counterpart. The expressiveness of the relational model and OWL are different, and could therefore explain our difficulties in conveying the same semantics in both representations. These results are of great importance for the Army in their consideration of using ontologies such as it is advocated by the semantic web community.

Future work: The results of this report will be versed in SATAC KAR Tasks 7 and 8 that aim to demonstrate how automated reasoning can be made possible with both JC3IEDM and P-JC3IEDM.

Sommaire

SATAC Knowledge Representation and Automated Reasoning with JC3IEDM

Hugues Demers, Jean-Rémi Duquet ; DRDC Valcartier CR 2008-255 ; R & D pour la défense Canada – Valcartier ; septembre 2008.

Contexte : L'objectif de ce document est de décrire les découvertes de la tâche 4 du contrat *Représentation de la connaissance et raisonnement automatisé avec JC3IEDM* (W7701-061941/001/QCL). Ce contrat a été réalisé sous les auspices du projet de recherche appliquée *Analyse de la Situation pour le Commandant d'Armée Tactique (ASCAT)*. Le but de ce projet est de donner les moyens à l'armée de Terre de se construire une capacité de raisonnement automatisé en support à l'analyse de la situation, processus par lequel la conscience de la situation est acquise. Sous ASCAT, ce contrat vise à évaluer la pertinence de JC3IEDM et P-JC3IEDM en support au développement d'un mécanisme d'inférence pour le processus de prise de décision du commandant d'armée tactique. Ceci sera effectué par la conduite de revues de littérature, d'analyses utilisant des outils et approches formelles d'ingénierie ontologique, et la réalisation d'un prototype de système de support à l'analyse de la situation basé sur les connaissances.

Résultats : Ce rapport montre comment certains éléments de situation militaires développés dans la tâche 3 peuvent être encodés dans le JC3IEDM et P-JC3IEDM. Bien que les difficultés associées à la réalisation de cette tâche avec le JC3IEDM sont bien connues, dû à la normalisation relativement grande de JC3IEDM, il est plus difficile encore de le faire avec la version OWL (P-JC3IEDM). Certaines modifications à P-JC3IEDM sont donc proposées pour faciliter cet état de choses.

Importance : Les découvertes de ce rapport montrent qu'un encodage de connaissances simple dans P-JC3IEDM, tel qu'il serait requis par le personnel de l'armée sur une plateforme déployée, n'est pas une tâche triviale. Même si ce n'est pas non plus facile avec JC3IEDM, le faire avec la version OWL est presque impossible dans certains cas. Nous posons que ceci est dû à la nature de P-JC3IEDM, étant une traduction automatisée de JC3IEDM en OWL. L'expressivité du modèle relationnel et celle de OWL sont différentes ce qui pourrait expliquer nos difficultés à rendre sémantiquement équivalents les deux représentations. Ces résultats sont d'une grande importance pour l'armée de Terre dans sa considération d'utiliser les ontologies tel que proposé par la communauté du web sémantique.

Perspectives : Les résultats de ce rapport seront versés dans les tâches 7 et 8 visant à démontrer comment le raisonnement automatisé peut être rendu possible avec JC3IEDM et P-JC3IEDM.

Table of contents

List of figures

1 Introduction

The purpose of this document is to describe the results of Task 4 of the Situation Analysis for the Tactical Army Commander (SATAC) Knowledge representation and Automated Reasoning (KAR) project, "Knowledge Representation Capabilities and Limitations of JC3IEDM and P-JC3IEDM". The purpose of Task 4 is to study and document the strengths and weaknesses of JC3IEDM and P-JC3IEDM in representing situations and situation analysis in general. It also investigates the abilities and deficiencies of JC3IEDM and P-JC3IEDM to represent the elements of the Full-Spectrum Operation (FSO) scenario of Task 3. Finally, this Task formalizes and encodes the reasoning examples produced in Task 3 both in JC3IEDM and P-JC3IEDM formalisms by producing a domain expert knowledge base.

1.1 Document Overview

This document is organized as follows. Section 2 gives an overview of the Knowledge Representation (KR) languages used in this document. Knowledge representation using JC3IEDM is discussed in section 3 and using P-JC3IEDM in section 4. Sections 5 and 6 present case studies for reasoning with JC3IEDM and P-JC3IEDM, respectively. Section 7 shows how specific reasoning arguments from the Task 3 document [1] can be encoded into JC3IEDM and P-JC3IEDM. Finally, conclusions are drawn in section 8.

The notation used in this document is as follows. The names of classes and properties used in P-JC3IEDM are preserved. Varying fonts are used to differentiate OWL-DL elements from XML code and from class individuals. OWL classes and object properties are typed as sans serif letters (uppercase or lowercase, depending on the actual P-JC3IEDM notation) like this OBJECT-ITEM and object-item-status-hostility-code. OWL-DL XML examples and statements are typed as typewriter letters like this `owl:Class`. Individual members of OWL classes are denoted as slanted lowercase letters like this *aircraft-instance*. Note that the latter is not italic, which appears like this *aircraft-instance*.

OWL-DL statements can appears as XML code like this:

```
<owl:Class rdf:ID="Fixed_wing__manned">
  <rdfs:subClassOf rdf:resource="#AIRCRAFT-TYPE"/>
</owl:Class>
```

However, most OWL-DL statements appear like this:

Class(Fixed_wing__manned subClassOf(AIRCRAFT-TYPE))

meaning that Fixed_wing__manned is a subclass of AIRCRAFT-TYPE. The previous statement can also be expressed as:

Class(Fixed_wing__manned)

subClassOf(Fixed_wing__manned AIRCRAFT-TYPE)

The subClassOf statement is given as a predicate, which can be read as: Fixed_wing__-manned is a subclass of AIRCRAFT-TYPE. In the same manner, a property will be expressed as:

hasParent(*Alex John*)

which can be read as *Alex* has parent *John*. Individuals are declared as:

Individual(*Alex* type(PERSON))

meaning that *Alex* is an individual member of class PERSON.

The ontologies that are JC3IEDM and P-JC3IEDM are examined to see how they represent different concepts. The focus of the analysis will be on concepts related to reasoning and those pertinent to the FSO scenario described in Task 3 [1]. Both JC3IEDM and P-JC3IEDM have the same conceptual model, which describes the high level view of information in terms of generalized concepts such as actions, organisations, materiel, etc. They are both a specification of a shared conceptualization but they differ in their way of formalizing concepts.[1] JC3IEDM formalizes concepts using an Entity-Relationship (E-R) model and P-JC3IEDM formalizes them using the Description Logic flavor of the Web Ontology Language, known as OWL-DL. Each is expressed using its own knowledge representation language, and their expressive power differs.

When knowledge needs to be expressed, one chooses a representation language, thereby committing oneself to the limitations of the expressiveness of that language, particularly the ontological commitment that the language makes [2]. Any representation language assumes something about the world; propositional logic assumes that there are facts that are either true or false, whereas first-order logic sees the world as composed of objects with relations that hold or do not hold. Thus, not everything may be expressed using a particular language. For example, the E-R model cannot express the fact that certain relationships are actually symmetric, that if *Alex* is-brother-of *Bob*, then necessarily *Bob* is-brother-of *Alex*. OWL on the other hand can assert this fact explicitly using the characteristic SymmetricProperty. Its being possible to state a fact with one language does not preclude its being easier to express with another.

Two KR languages will be examined: the entity-relationship model as implemented by SQL[2] and description logic as implemented by OWL 1.0. SQL is the language used in most database management systems and is the expression of two formalisms:

1. relational algebra which consists of set operators like union and special operators like join
2. relational calculus which is based on first-order logic.

OWL exists in three flavors: OWL-Lite, OWL-DL and OWL-Full [2]. OWL-DL is a description logic corresponding to $\mathcal{SHOIN}(\mathbf{D})$ and a subset of first-order logic. Both SQL and OWL-DL will be briefly presented in the following section.

[1] An ontology is a formal specification of a shared conceptualization [2].

[2] The actual implementation of SQL does not strictly follow the relational model as proposed by Codd, see discussion in [2].

Task 3 provides a number of reasoning arguments that are formulated as logical implications. These implications can be coded as rules to derive knowledge. As such, rules play the important role of being the first choice to express derived knowledge. Implications and rules are described in the following section.

2 Knowledge Representation Languages Overview

2.1 Basic Elements of Rules and Logical Implications

Deductive reasoning arguments are language constructs having a set of premises and a conclusion whereby if the premises are true, it becomes impossible for the conclusion to be false. This construct is usually written in a formal logical language using an implication, the classic example being:

All men are mortal
Socrates is a man
Therefore, Socrates is mortal.

This can be translated in first-order logic as

$$\frac{\forall x \;\; \mathsf{Man}(x) \Rightarrow \mathsf{Mortal}(x), \quad \mathsf{Man}(Socrates)}{\mathsf{Mortal}(Socrates)}$$

where the logical implication is given first, followed by the fact from which the conclusion follows.

A rule and an implication are closely related. An implication is an expression of a logical formula language, most often first-order logic, that possess a truth-value, while a rule does not possess a truth-value per se but has the role of generating derived sentences. In most logic, however, there is a close relationship between a derivation rule and the corresponding implication formula; they both have the same models [3]. In the following formulas, a rule will be denoted by $\rightarrow$ and a logical implication by $\Rightarrow$.

A rule consists of an antecedent (also called the body) and a consequent (the head). The informal meaning here is that whenever the conditions specified by the antecedent hold, the conditions of the consequent must also hold. The syntax is

$$\text{antecedent} \rightarrow \text{consequent}$$

where both terms are conjunctions of atoms (possibly zero) written $P_1 \wedge \ldots \wedge P_n$. It is sometimes called an if-then rule, an implication, or a Horn-like rule. They are not true Horn clauses since the consequent is allowed as a conjunction, whereas Horn clauses are not; only one atom is allowed. The most common human readable syntax [4] for rules places a question mark (e.g. $?x$) before a variable. The famous uncle rule would be written:

$$\mathsf{hasParent}(?a, ?b) \wedge \mathsf{hasBrother}(?b, ?c) \rightarrow \mathsf{hasUncle}(?a, ?c)$$

The semantic of the word *rule* is heavily overloaded in the field of databases and AI. It is useful here to differentiate between different kinds of rules [5, 3, 6]:

Constructive rules They specify how to derive new data from data already available. They are called *views* in databases and are related to logical implications in AI. Views typically involve data selection and grouping.

Active rules They specify how a knowledge base or database can be modified depending on its current state. They are of two forms: *if condition then action* and *on event if condition then action*. Rules of the first form are called production rules and the second forms are Event-Condition-Action or ECA rules. They are also known as reactive rules or triggers.

Normative rules More commonly known as integrity constraints, they express conditions that data must fulfill.

Parsia et al. in [7] state that OWL is a very expressive language, but they also admit that even seasoned logicians find description logic perplexing, misleading and simply confusing when it comes to what they can or cannot express. That is why so many new approaches focus on merging rules with description logics, the most famous being the Semantic Web Rule Language (SWRL) which is roughly the union of Horn logic and OWL. SWRL has the full power of OWL-DL but at the price of decidability, in particular for ontology consistency inference. OWL is able to express most reasoning arguments that seem to be expressible only by rules as long as the ontology is suitably modified. There is even a technique exposed in [7], called the rolling-up technique, by which some rules are directly represented in OWL by transforming them into class axioms.

The rolling-up technique is a procedure to generate description logic class expressions that contain the constraints placed on a single variable $?x$. The class expression is called the rolled-up class for $?x$. In the case of rules, both the antecedent and the consequent are treated as conjunctive queries and the addition of the assertion that antecedent class expression is a subclass of consequent class expression ensures the intended rule semantics. For example, the following rule describes a Military as a person who is assigned to a military organisation:

$$\mathsf{Person}(?x) \wedge \mathsf{isAssignedTo}(?x, ?y) \wedge \mathsf{MilitaryOrganisation}(?y) \rightarrow \mathsf{Military}(?x).$$

The rolling-up technique makes sure that each side of the rule becomes a class expression and that the antecedent becomes a subclass of the consequent:

$$\mathsf{Person} \sqcap \exists \mathsf{isAssignedTo}.\mathsf{MilitaryOrganisation} \sqsubseteq \mathsf{Military}.$$

Thus the antecedent is now the set described by things that are a Person that have, amongst other things, the property isAssignedTo with some value that are member of class MilitaryOrganisation. The rolling-up technique states that the resulting set should be a subset of Military. Type inference, however, will not classify under Military any Person with such property because the subclass relation stipulates only necessary conditions whereas sufficient conditions are needed for reasoning to occur. In order to infer that someone is a military, the equivalent class property of OWL must be used. The equivalent class expression is:

$$\mathsf{Person} \sqcap \exists \mathsf{isAssignedTo}.\mathsf{MilitaryOrganisation} \equiv \mathsf{Military}.$$

There are conditions that the rule must satisfy for the relation between the two resulting class expressions (the antecedent and the consequent) to have its intended semantics. These conditions are placed on the number of variables shared between the consequent and the antecedent:

- if zero variables are shared, then the rule can be expressed in OWL as long as one individual is shared.
- if one variable is shared, then the rule can be expressed in OWL.
- if two or more variables are shared, then the rule cannot be expressed directly in OWL.

These conditions stem from the fact that classes are essentially one-place predicates that describe the condition placed on a single variable; the predicates assert the membership of the individual to that class.

The rolling-up technique is a procedure for turning rules into complex class expressions and axioms that require significant restrictions on the structure of the rules. Thus rules that can be turned into class expressions are nothing more than syntactic sugar; they are a simpler way of expressing knowledge. And in that sense, these rules are useful in their own right. For further details of the procedure, the reader is referred to [7].

2.2 Basic Elements of SQL

JC3IEDM's logical model is expressed in an E-R diagram, and thus the primary implementation which the physical description is about is on a Relational Database Management System (RDBMS). Since most RDBMS implement an SQL interface to access their data, it is appropriate here to introduce the basic elements of SQL.

SQL stands for Structured Query Language and its primary use is querying on tables in a database with strong data manipulation statements to insert, update and delete elements of a table. There exist, however, many features that supply SQL with additional functionalities, e.g. triggers and views, that give it the flavor of a reasoning engine. The following gives an overview of SQL with a strong emphasis on elements related to inference. Note that the examples given work for the MySQL[3] RDBMS but that the syntax for making the same queries might change for another RDBMS.

2.2.1 SQL Syntax and Semantics

The most common commands of SQL are:

select Used to retrieve rows from one or more tables. For example, selecting all rows from a table named employee would be performed with:
`SELECT * FROM employee;`
where the `*` means all rows.

[3] MySQL is a trademark of MySQL AB

insert Used to insert new rows into a table. Inserting a new row into table employee with values John, Smith and 1002 would be performed with:
`INSERT INTO employee VALUES ('John', 'Smith', 1002);`

update Used to update columns of existing rows with new values. Updating the name of an employee would be performed with:
`UPDATE employee SET surname='Doe';`

delete Used to delete rows from a table.

These commands form the basis of what is sometimes called the SQL data manipulation language. The most interesting part of SQL for this study, however, is its ability to combine information and to infer knowledge. Combining information is most commonly known as a join. Some authors have said that joins (also known as inner joins) put the *relational* in relational databases [8] meaning that a join enables one to match a row from one table with a row from another table. Thus joining table is a matter of specifying equality in columns from two tables (refer to Tables 1 and 2):

```
SELECT department.name, employee.name
FROM employee, department
WHERE employee.id = department.manager
```

Table 1: department Table

ID	name	manager	no. employees
1	Accounting	100	10
2	Finance	101	5
6	Production	112	145
4	R&D	102	4

Table 2: employee Table

ID	name	employee number	department
100	John Smith	5005	1
101	John Doe	5006	2
112	Alex Tremblay	5067	6
102	Jane Smith	5090	4
105	Bob Pearson	5091	4

The join expressed above yields Table 3 which shows the manager's name for each department.

Joins are a way of aggregating information from two or more tables. However, there is another way of doing that using views. In SQL, a view is a virtual table based on the result-set of a SELECT statement. A view is akin to an ordinary table in the sense that in can be queried, but rows cannot be inserted or deleted. Each time a view is queried, the whole virtual table is rebuilt in order to obtain the latest values from its constituent tables.

Table 3: Query Results Based on a Join of Tables 1 and 2

department.name	employee.name
Accounting	John Smith
Finance	John Doe
R&D	Jane Smith
Production	Alex Tremblay

A view is conceptually the same as an implication; the existence of a row in a view means that the SELECT statement evaluated to true. For instance, the following implication says that all departments with more than 100 employees is a big department:

$$\forall x \ \mathsf{GreaterThan}(\mathsf{NumberOfEmployee}(x), 100) \Rightarrow \mathsf{BigDepartment}(x).$$

This implication can be turned into a view using the following SQL construct:

```
CREATE VIEW big_department AS SELECT department.name
  FROM department
  WHERE department.number_of_employee > 100
```

The resulting view will have one column with the name of any department with more than 100 employees, in this case Production. Although, this is a very simple example, views can gather information from multiple tables. Whenever the implication evaluates to true, the view will reflect that fact by presenting the corresponding row. Views, however, just present data; they do not actively modify the database when an implication evaluates to true. Triggers are needed to modify data.

A trigger is a piece of SQL code that is activated on the occurrence of certain specific events. For example, a trigger could be activated upon insertion of new data into a table. The events that act as triggers are generally INSERT, UPDATE and DELETE. To increment the number of employees in the department table, one could trigger on the insertion of a new record in the employee table like so:

```
CREATE TRIGGER employee_count_trigger AFTER INSERT ON employee
FOR EACH row
  UPDATE department SET department.number_of_employee =
      ( department.number_of_employee + 1 )
    WHERE department.id = NEW.department;
```

The above trigger increments by one the department's number of employees after a new row is inserted into table employee. This trigger is extremely simple and was chosen to exemplify the fact that rules are often used to write back to the database rather than merely showing information like views do. The trigger could be summarized with:

if a new employee is hired in a specific department

then increment the number of employees in the corresponding department.

Triggers and views can be related to rules and logical implications, respectively. The difference is admittedly subtle, but nonetheless interesting to make. Rules taken as generators of derived sentences can be seen as triggers, whereas implications are expressions of a logical language that assign a truth-value to the conclusion. In a similar manner, a view assigns a truth-value to the conclusion by presenting (truth-value is true) or not (truth-value is false) a row.

2.2.2 Reasoning with SQL

Reasoning with an RDBMS is most commonly done using production rules. RDBMSs that provide such a facility are generically termed active databases. These databases implement an active rule engine, which monitors events caused by database transactions and schedules rules according to given policies. This contrasts with the passive behavior of conventional database systems.

With active databases, a large fraction of the semantic that is normally encoded within external applications can be expressed with rules, giving external applications a new independence called knowledge independence [6] in which applications are freed from the need to express knowledge using their own rules. Instead these rules are coded directly in the database schema. This has the consequence of automatically allowing all users for which the database is replicated to share the rules. Knowledge is modified by changing the rules and not the external applications.

Reasoning with an RDBMS involves using SQL triggers and possibly views to produce new knowledge. These facilities are certainly not as sophisticated as those provided by OWL for example. They are powerful enough, however, to allow a wide range of reasoning to take place. One problem with rules remains the prediction of their collective behavior. One aspect of the problem is termination, which is not always guaranteed for a given set of rules. As with many other rule-based expert systems, SQL triggers may interact in subtle and unexpected ways and the database may never produce a stable state.

SQL is loosely based on relational algebra which, according to [6] cannot express many important queries. For this reason, more powerful languages have been developed that subsume relational algebra, the most popular one being Datalog.

Datalog is a syntactic subset of Prolog and is used as a query and rule language. In a Datalog representation, the database is viewed as a set of facts, one fact for each row in the corresponding table of the relational database, where the name of the relation becomes the predicate name of the fact. Going back to the example of the previous section, a big department could be defined as:

$$\text{BigDepartment}(\mathit{Name}) \leftarrow \text{Department}(\text{–}, \mathit{Name}, \text{–}, \mathit{NEmployee}),\ \ \mathit{NEmployee} > 100$$

where – stands for an anonymous variable. The head of the rule is BigDepartment and the body of the rule is composed of two *goals* separated by a comma that stands for

a logical conjunct. Other variables are emphasized and must fulfill the requirements of the body for the rule to fire. The BigDepartment predicate is called a derived predicate. Derived predicates are normally found only in the heads of rules and form the intensional database. The intensional database is formed by virtual tables; data derived from rules. The extensional database is built from the base predicates, which are defined by the database schema.

2.3 Basic Elements of OWL-DL

The following briefly introduces basic elements of OWL-DL that will be necessary in understanding the knowledge representation capabilities of P-JC3IEDM. The first part describes the syntax elements of the language and the second part gives examples of reasoning with OWL-DL. For more details the reader is referred to the official OWL documentation found in [9, 10, 11].

2.3.1 OWL-DL Syntax and Semantics

Most of the elements of an OWL ontology concern classes, properties and individuals, also called class instances. A class is interpreted as a set of individuals, and a property generally describes a relationship between two individuals. Classes are the basic blocks of an OWL ontology. Every entity in the OWL world is a member of the class `owl:Thing`. Thus each user-defined class is implicitly a subclass of `owl:Thing`. There are six ways of declaring classes using a constructor, as follows:

Named class The simplest way of declaring a class using `owl:Class`. Named classes can be arranged in a taxonomy with a subsumption relationship, using the subclass property `rdfs:subClassOf`

Intersection class Formed by combining two or more classes using the set operator `intersectionOf`

Union class Formed by integrating two or more classes using the set operator `owl:unionOf`

Complement class Specified by negating another class using the construct `owl:complementOf`

Restriction Describes a class of individuals based on the type and possibly the number of relationships that they participate in. It describes an *anonymous* class composed of all the individuals that satisfy the restriction. See below.

Enumerated class Specified by explicitly and exhaustively listing the individuals that are members of the enumerated class using the `owl:oneOf` construct.

For example, the named class Fixed_wing__manned would minimally be written in XML like this:

```
<owl:Class rdf:ID="Fixed_wing__manned">
  <rdfs:subClassOf rdf:resource="#AIRCRAFT-TYPE"/>
</owl:Class>.
```

OWL provides class axioms that typically contain additional components that state necessary, or necessary and sufficient characteristics of a class. Three axioms exist:

subClassOf Allows one to say that a particular class is a subset of another class. This axiom describes the necessary conditions for an individual to be a member of that class. It is also referred to as a *partial* description.

equivalentClass Allows one to say that a class has exactly the same description as another one. This axiom describes the necessary *and* sufficient conditions that one individual be a member of the class. It is also referred to as a *complete* description.

disjointWith Allows one to say that a class has no members in common with those of another class.

The second axiom, equivalentClass, is very important for reasoning with an ontology. Because it describes necessary and sufficient conditions, as soon as an individual satisfies these particular conditions, it can be classified as a member of the class. Satisfying only necessary conditions, provided by the subClassOf axiom, does not lead to sophisticated reasoning except to say that a member of a subclass is also a member of the parent class.

In addition to classes, an ontology must be able to describe their members. In OWL these are called individuals or a class's instances. An individual is minimally introduced by declaring it to be a member of a class. The following declares individual CP-140_Aurora to be a member of class Fixed_wing__manned:

```
<Fixed_wing__manned rdf:ID="CP-140_Aurora"/>
```

2.3.1.1 Properties

Properties assert general facts about the members of classes and specific facts about individuals. A property is a binary relation that can have a cardinality of many-to-many. Two types of properties are distinguished:

Datatype properties Relations between instances of classes and literals (or datatypes), i.e. primitive types like integer, string and float.

Object properties Relations between instances of two classes.

It is possible to specify global property *characteristics*, which provides a mechanism for enhanced reasoning about a property. They are termed global because the characteristics apply to all instances of the property. The most important characteristics are briefly described here.

domain A domain of a property limits the individuals to which the property can be applied. If a property relates an individual A to another individual B, and the property has a class as its domain, then individual A must belong to that class.

range The range of a property limits the individuals that the property may have as its value. If a property relates an individual A to another individual B, and the property has a class as its range, then individual B must belong to the range class.

subPropertyOf Defines a property as a sub property of another. If property P_2 is a sub property of P_1, then anything with property P_1 with value x also has property P_2 with value x.

inverseOf One property may be stated to be the inverse of another property. If property P_1 is stated to be the inverse of property P_2, then if x is related to y by the P_2 property, y is related to x by the P_1 property.

TransitiveProperty Properties may be stated to be transitive. If a property is transitive, then if the pair (x, y) is an instance of the transitive property P, and the pair (y, z) is an instance of P, then the pair (x, z) is also an instance of P.

SymmetricProperty Properties may be stated to be symmetric. If a property is symmetric, then if the pair (x, y) is an instance of the symmetric property P, then the pair (y, x) is also an instance of P.

FunctionalProperty Properties may be stated to have unique values. If a property is a FunctionalProperty, then it has no more than one value for each individual (it may have no values for an individual).

2.3.1.2 Restrictions

In addition to characteristics, properties may have local restrictions. These are constraints on the range of a property local to their containing class definition.

allValuesFrom This restriction is stated on a property with respect to a class. It means that this property for this particular class has a local range restriction associated with it. Thus if an instance of the class is related by the property to a second individual, the second individual can be inferred to be an instance of the local range restriction class.

someValuesFrom This restriction is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type.

hasValue This restriction is stated on a property with respect to a specific individual. It requires that the property have at least this individual as its value.

minCardinality, maxCardinality Restricts the number of values that a property can have.

2.3.1.3 Individual's Identity

In OWL, different individual's names can be used to refer to the same things in the world. For example, the same person can be referred to in many different ways. Unless an explicit statement is made that two references refer to the same or to different individuals, OWL assumes either situation is possible. Thus, OWL does not make a unique name assumption. Three constructs exist for stating facts about the identity of individuals:

sameAs is used to state that two references refer to the same individual.

differentFrom is used to state that two references refer to different individuals.

AllDifferent is used to state that individuals from a set are mutually distinct.

2.3.2 Reasoning with OWL-DL

The following demonstrates some of the reasoning capabilities of OWL-DL. Reasoning examples are given for each case.

2.3.2.1 Type Inference Through Domain and Range Constraints

The following asserts two individuals of unknown type, *Alpha* and *Beta* members of owl:Thing, the all-encompassing class. *Alpha* has the property has-for-support-a-specific with value *Beta*. The property has-for-support-a-specific is asserted with a domain of MILITARY-ORGANISATION-TYPE and a range of UNIT-TYPE. The property is-constituted-to-support is also asserted as being the inverse of has-for-support-a-specific.

Individual(*Alpha* type(owl:Thing) value(has-for-support-a-specific *Beta*))

Individual(*Beta* type(owl:Thing))

ObjectProperty(has-for-support-a-specific domain(MILITARY-ORGANISATION-TYPE) range(UNIT-TYPE))

ObjectProperty(is-constituted-to-support inverseOf(has-for-support-a-specific))

By asserting that *Alpha* has for support a specific *Beta*, it is possible to infer that *Beta* is constituted to support *Alpha* and that *Alpha* is a MILITARY-ORGANISATION-TYPE and *Beta* a UNIT-TYPE.

2.3.2.2 Type Inheritance Through subClassOf

Asserting that

subClassOf(Fixed_wing__manned AIRCRAFT-TYPE)

Individual(*CF-18* type(Fixed_wing__manned))

implies that

Individual(*CF-18* type(AIRCRAFT-TYPE)).

In other words, a *CF-18* is a member of class Fixed_wing__manned which in turn is a subclass of AIRCRAFT-TYPE. Thus a *CF-18* is also a member of AIRCRAFT-TYPE.

2.3.2.3 Transitivity of subClassOf

If a UNIT is an ORGANISATION and an ORGANISATION is an OBJECT-ITEM, than a UNIT is an OBJECT-ITEM. Or,

subClassOf(UNIT ORGANISATION)

subClassOf(ORGANISATION OBJECT-ITEM)

implies that

subClassOf(UNIT OBJECT-ITEM).

2.3.2.4 Enforcing OWL-DL Property Characteristics

OWL-DL introduces property characteristics (see section 2.3.1.1) that are not only syntactical and semantical language constructs, but are also enforced by the inference engine. Thus, symmetric, transitive, inverse of and functional property characteristics imply some reasoning mechanisms. For example, the following

TransitiveProperty(is-part-of)

Individual(*tire* is-part-of(*wheel*))

Individual(*wheel* is-part-of(*car*))

implies the fact

is-part-of(*tire car*).

2.3.2.5 Inferring Disjointedness from a Class's Complement or Through Inheritance

The assertion that two classes are disjoint (i.e. two individuals cannot be members of both at the same time) can be inferred from one parent's complementOf characteristic or through inheritance. For example,

complementOf(Matter Antimatter)

subClassOf(Proton Matter)

implies the fact

disjointWith(Proton Antimatter)

and

disjointWith(FACILITY ORGANISATION)

subClassOf(ROAD FACILITY)

implies the fact

disjointWith(ROAD ORGANISATION).

2.3.2.6 Necessary and Sufficient Reasoning

Whenever a class is described by necessary and sufficient conditions, it is possible to infer that an individual belongs to that class if it satisfies the conditions. For example,

Class(Driver subClassOf(Person))

equivalentClass(Driver intersectionOf(Person restriction(drives someValuesFrom(Vehicle)))).

The first description of Driver asserts that it is necessary for a Driver to also be a Person. A Person is not necessarily a Driver however. The second description asserts that it is necessary *and* sufficient for a Person that has, amongst other things, the property drives some value of a Vehicle to be a Driver. Whenever such a Person is found it can be classified as a Driver. This kind of reasoning will play an important role in the encoding of inference premises in P-JC3IEDM since it provides a powerful means of describing class membership in terms of properties individuals must have.

The reasoning examples given above do not constitute an exhaustive list; other reasoning mechanisms exist. They give, however, an idea of how reasoning is done in OWL-DL.

3 Knowledge Representation in JC3IEDM

JC3IEDM is a product of the analysis of a wide spectrum of Allied information exchange requirements and as such can express a great variety of facts about military situations. It models the information that combined joint component commanders need to exchange. The following will not focus on the representation power of JC3IEDM regarding most elements of military significance. This study assumes that JC3IEDM is able to represent these elements. The focus will rather be on the analysis of the representation of *reasoning* arguments.

In order to express reasoning arguments, a knowledge representation language needs to be able to speak about objects of the world and to put these objects in relation to one another. For example, to express the knowledge contained in a premise of the form

if a group has perpetrated an ambush

then it is hostile,

a KR language needs to encode the fact that a group has perpetrated an ambush, the fact that a group is hostile and, perhaps more importantly in this case, the fact that one implies the other. It is common to differentiate an object language and a metalanguage where the latter is used to talk about the former. To encode the above premise, a metalanguage needs the necessary construct to express an implication between various objects. The JC3IEDM model does not have rich enough constructs to speak about itself in this way. This is understandable because this model did not mean for such knowledge to be encoded. It is used to represent facts and not reasoning premises. The most common language in the field of RDBMS that has the necessary constructs is SQL. According to Sowa [12], "SQL supports full FOL [...] and makes first-order logic, as expressed in SQL, the most widely

used version of logic in the world." SQL is rich enough to express knowledge in the form of rules. For these reasons, it will be used in section 5 to build an example of a relatively simple reasoning argument.

4 Knowledge Representation in P-JC3IEDM

As discussed in [2], JC3IEDM is described by three models:

- The conceptual data model generalizes concepts and presents them in a high level fashion.
- The logical data model represents all of the information and is based upon breaking down high level concepts into specific information that is regularly used. It specifies the way data is structured with an E-R model and supporting documentation.
- The physical data model provides the detailed specifications necessary to implement the schema of a database.

P-JC3IEDM was not produced like JC3IEDM's logical model was. In producing the latter, the encoding of the ontology (the formalization) was done from a conceptual model. P-JC3IEDM, however, was produced from JC3IEDM's logical entity-relationship diagram, hence all the limitations of this model were translated into OWL-DL. The logical model of JC3IEDM already has the constraints and shortcomings of the E-R model and to correctly express JC3IEDM in another KR language, it would be necessary to start from the conceptual rather than from the logical model [2]. One reason for this is that E-R logical models encompass associative entities that are used solely for the purpose of representing many-to-many relationships. Other KR mechanisms may have built-in capabilities to represent such relationships, hence porting JC3IEDM into such mechanisms might require special treatment of the associative entities.

Since the automatic translation of P-JC3IEDM did not lose any entities, attributes or relationships, it is fair to expect that most military situations expressible by JC3IEDM are also expressible by P-JC3IEDM. There exist some differences in how P-JC3IEDM expresses knowledge, like the fact that some attributes are no longer mandatory (see section 4.2.4), but this does not necessarily reduce the expressive power of P-JC3IEDM. These differences will be explored in section 4.2. The emphasis, however, for the remaining analysis will be on how P-JC3IEDM can support reasoning.

The ontology that is P-JC3IEDM contains:

- named classes
- enumerated classes
- properties defined with:
  - a domain

- a range
- sometimes a cardinality
- sometimes as the inverse of another property (when JC3IEDM's reverse verb-phrase exists).

There are no rich class constructors like intersection, union and complement. There are no equivalent and disjoint class axioms.

An inference premise is a reasoning example stated as an *if-then* rule such as:

if a person is military

then he is not a civilian.

Expressing the knowledge contained in an inference premise consists of encoding the ontology in such a way that the inference engine will be able to correctly conclude. Not all knowledge representation languages allow the expression of all inference premises. OWL is able to express the previous inference premise; however, it fails to express a lot of them. Notably it cannot express premises that involve properties. Many of the limitations of OWL stem from the fact that it does not provide a language rich enough for reasoning over properties. OWL includes a rich set of class constructors but is much weaker regarding property constructors [4].

4.1 Automatic Translation of P-JC3IEDM

The translation methodology employed by Matheus and Ulicny [13] is summarized here (for more details see [2]). Using a set of XSLT scripts, an automated procedure was developed for translating the XML description of JC3IEDM into OWL-DL according to the following general rules:

1. An entity and a category-code attribute are translated to `owl:Class`.
2. The entity's attributes (other than category-code) are translated to `owl:ObjectProperty` or `owl:DatatypeProperty`.
3. The entity's relationships are translated to `owl:ObjectProperty` having a unique identifier `rdf:ID`.
4. The attributes' values (JC3IEDM's domains) are translated to enumeration classes.

It is stressed here that the category-code attribute of each entity is translated not as an `owl:ObjectProperty` like other attributes but rather as an `owl:Class`. This makes the entity a superclass of each class translated from category-codes. It extends the taxonomy tree down to category-codes, the underlying assumption being that category-codes are concept specifications. This choice has important consequences that will be discussed below.

The following are OWL-DL XML excerpts from P-JC3IEDM. The first is the class definition of the OBJECT-ITEM entity.

```
<owl:Class rdf:ID="OBJECT-ITEM">
  <rdfs:label rdf:datatype="&xsd;string">
    OBJECT-ITEM
  </rdfs:label>
  <rdfs:comment>
    An individually identified object that has military or
    civilian significance.
  </rdfs:comment>
</owl:Class>
```

This is the definition of the aircraft-type-model-code attribute translated as an object property.

```
<owl:ObjectProperty rdf:ID="aircraft-type-model-code">
  <rdfs:domain rdf:resource="#AIRCRAFT-TYPE"/>
  <rdfs:range rdf:resource="#AircraftTypeModelCode"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

The relationship is-the-subject-of that exists between an OBJECT-ITEM and an OBJECT-ITEM-ASSOCIATION is encoded by:

```
<owl:ObjectProperty rdf:ID="OI-is-the-object-of-OIA">
  <rdfs:domain rdf:resource="#OBJECT-ITEM"/>
  <rdfs:range rdf:resource="#OBJECT-ITEM-ASSOCIATION"/>
</owl:ObjectProperty>
```

Using the above method to automatically translate JC3IEDM ensures that all syntactical elements are translated into OWL-DL concepts. Restriction of the E-R model, however, imposes design peculiarities to JC3IEDM such as database normalization. The E-R model restrictions are not the same as those of OWL-DL, hence expressing an ontology in OWL-DL will not lead to the same design patterns. JC3IEDM design peculiarities have been automatically transposed into OWL-DL. JC3IEDM's concepts (the conceptual model on the ontology) are probably not expressed optimally in OWL-DL. The following section explores these issues.

4.2 Analysis of P-JC3IEDM's Translation

One of the goals of Matheus and Ulicny [13] in producing P-JC3IEDM was to automate the process and allow the rapid building of an OWL ontology from a new version of JC3IEDM. The product of such an automation is a representation of JC3IEDM's logical model to which rules and reasoning can be added. Unfortunately there are some shortcomings to this way of translating JC3IEDM. This section analyses from a knowledge representation

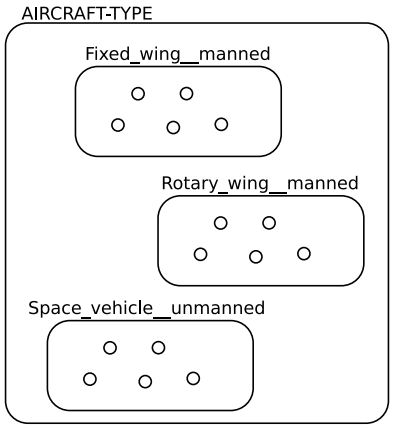

Figure 1: Some of the AIRCRAFT-TYPE subclasses.

perspective the shortcomings of P-JC3IEDM's automatic translation in order to understand the limitations of the reasoning made possible by this ontology. Comments from this section consist of the observations made while trying to represent knowledge in P-JC3IEDM. It is not an exhaustive list, but serves to illustrate the limitations of its expressivity.

4.2.1 Category-Code Translated as owl:Class

The choice was made to extend the taxonomy tree using the category-code of each entity. This way, each category-code is turned into an `owl:Class` defined as a subclass of the entity. For example, the AIRCRAFT-TYPE entity has the following category-code: *Fixed wing manned*, *Rotary wing manned*, *Space vehicle unmanned*, etc. These category-codes are kinds of AIRCRAFT-TYPE; they further define an AIRCRAFT-TYPE, making them suitable to be turned into subclasses (see Fig. 1).

There are cases however when the category-code does not further define the entity but rather describes it. For example, the REPORTING-DATA entity has the following category-codes: *Assumed*, *Erroneous*, *Inferred*, *Planned*, *Reported* and *Extrapolated*. The automatic translation has turned each category-code into a subclass of REPORTING-DATA. Other examples include the OBJECT-ITEM-ASSOCIATION's category-codes, some of which are: *Administers*, *Is brother of*, *Has as a member*, etc. In P-JC3IEDM, the category-code *Is brother of* has been turned into class Is_brother_of subclass of OBJECT-ITEM-ASSOCIATION. Thus, to represent the fact that *Alex* is brother of *Bob*, it is necessary to create an instance *brother-instance* of class Is_brother_of, which must be linked using the object properties is-the-object-of and is-the-subject-of[4] to *Alex* and *Bob* respectively, which are themselves instances of class PERSON. Thus, to express this knowledge using the ontology of P-JC3IEDM, the following must be added to the knowledge base:

Individual(*brother-instance* type(Is_brother_of))

is-the-object-of(*Alex*, *brother-instance*)

[4]Those object properties are declared to have a domain of OBJECT-ITEM, but because a PERSON is also an OBJECT-ITEM, they can be used to describe a PERSON.

is-the-subject-of(*Bob*, *brother-instance*).

Stating the above does not imply that *Bob* is brother of *Alex*. In order to assert that, the following must be added to the knowledge base:

is-the-object-of(*Bob*, brother-instance)

is-the-subject-of(*Alex*, brother-instance).

For example, expressing the same knowledge in an ontology where the category-code *Is brother of* is translated into an object property and not a class would be done by stating that

SymmetricProperty(is-brother-of)

is-brother-of(*Alex*, *Bob*).

Asserting that the property is symmetric allows the inference engine to deduce that *Bob* is brother of *Alex*, without having to explicitly add this information to the knowledge base. In this case, the category-code was translated into an object property having a domain and a range of OBJECT-ITEM.

Associative entities (e.g. JC3IEDM's entities ending with _ASSOCIATION) are used to resolve many-to-many relationships. They can be thought of as both entities and relationships. They are relationships since they join two or more entities together, but are still entities since they may have their own attributes. Because associative entities semantically express a relationship, they should be naturally translated as object properties in OWL. It is possible to apply constraints and restrictions to properties thereby adding knowledge (semantic) to the ontology.

There is, however, an advantage to having a relation represented as a class rather than a property. A property is a binary relation; it is used to link two individuals or an individual and a value. In some cases, however, the natural and convenient way to represent certain concepts is to use relations to link an individual to more than just one individual or value. These relations are called n-ary relations. A common representation pattern for modelling n-ary relations is to represent the relation as a class rather than a property [14]. Additional properties added to the class provide binary links to each argument of the relation. For example, the class Is_brother_of does not distinguish between brother and half-brother, in the first case both parents are common and in the second case only mother or father is common to both. Thus, the relation *Is brother of* is a ternary relation linking a person to a person and a member of the enumeration class CommonParent = {father, mother, both}. In Fig. 2, the individual is_brother_of_instance is a member of the class Is_brother_of. Object properties is-the-object-of and is-the-subject-of are inherited from the parent OBJECT-ITEM-ASSOCIATION and have a domain of OBJECT-ITEM and a range of OBJECT-ITEM-ASSOCIATION. The new property has-common-parent has a domain of OBJECT-ITEM-ASSOCIATION and a range of CommonParent. By making a property into a class, it is possible to qualify that class by adding n properties to it, thus representing an n-ary relation. This pattern increases the expressiveness but lowers the inference power since specific characteristics of a property cannot be applied. For example, it is not possible with this design to assert that the relation *is brother of* is symmetric. The inference engine cannot deduce that *Bob* is brother of *Alex*.

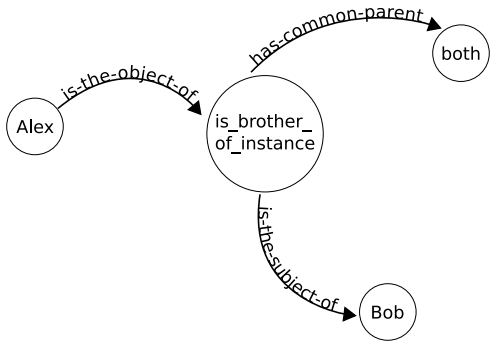

Figure 2: Example of an instance of the class ls_brother_of.

An important consequence of translating category-codes into classes—versus having category-codes defined as values of a property as is done for every other attribute of JC3IEDM—is the ability to reason on that class and thus change the category-code of an individual by changing its class membership. OWL can infer the type of an individual but cannot change the values that a particular individual's property may have. For example, it is not possible to assign the value of Hostile to the property object-item-status-hostility-code based on the condition that an organisation has performed an ambush. Thus the rule

if an organisation has performed an ambush

then the organisation is hostile

cannot be implemented in P-JC3IEDM. To express the previous rule, the different values that the property object-item-status-hostility-code may take would need to be turned into classes. It would then be possible to reason on the type of individual and classify it according to the previous rule (for an example of this see section 6).

4.2.2 OBJECT-TYPE and OBJECT-ITEM Taxonomy

JC3IEDM represents an object with a combination of OBJECT-ITEM and OBJECT-TYPE or their respective subtypes. An OBJECT-TYPE refers to a class object and an OBJECT-ITEM to an individually identified instance. JC3IEDM dictates, via a cardinality constraint, that a specific OBJECT-ITEM be associated with at least one instance of an OBJECT-TYPE. This ability to classify OBJECT-ITEM as OBJECT-TYPE makes any information that is stored as type data applicable to the item. Thus, any characteristic of an item that can be described as a type property does not need to be carried as an attribute on the item side.

OWL represents an object mainly by its membership to one or more classes. Relations in the form of ObjectProperty are used to link two individuals together.

One obvious consequence of the item-type dual taxonomy is that the type of an individual is not only given by its class membership, but by a combination of ObjectProperty and other individuals. The type of an OBJECT-ITEM is defined by two object properties with an OBJECT-ITEM-TYPE in the middle. OI-is-classified-as-OIT links the OBJECT-ITEM to the

OBJECT-ITEM-TYPE and OT-is-used-as-a-classification-for-OIT links the OBJECT-TYPE to the OBJECT-ITEM-TYPE.

It is not clear at this point what are the consequences (except for the increased complexity) of having two taxonomies in P-JC3IEDM for identifying a specific object, the OBJECT-ITEM tree and the OBJECT-TYPE tree. Task 7 should give more insight into this.

4.2.3 Multiple Inheritance in P-JC3IEDM

OWL allows an individual to be a member of two classes that are not disjoint, and by default OWL does not assume that classes are disjoint. Disjointedness between two classes must be explicitly stated as such. Since none of P-JC3IEDM's classes are asserted as disjoint it is possible to have individual *Alex* classified as PERSON and MATERIEL. For reasons that will be given in section 7.3 and further studied in Task 7, it is not obvious if P-JC3IEDM should enforce disjointedness between all classes that should obviously be disjoint.

4.2.4 Mandatory Attributes in P-JC3IEDM

Mandatory attributes are not enforced in P-JC3IEDM. The Null_Option is not translated into a cardinality constraint by the XSLT scripts. If Null_Option is true it expresses the fact that an attribute must be defined for a particular entity, i.e. the attribute is mandatory for the entity. For example, P-JC3IEDM does not enforce the connection between an OBJECT-TYPE and an OBJECT-ITEM-TYPE. Thus, it is possible to instantiate an OBJECT-ITEM and associate it with an OBJECT-ITEM-TYPE without ever associating the OBJECT-ITEM-TYPE with an OBJECT-TYPE. It is not clear at this point if mandatory attributes of JC3IEDM should be enforced in P-JC3IEDM. There is certainly an increase in representation flexibility, at the expanse however of a possible malformed knowledge base.

4.2.5 Miscellaneous Items

The only member of the enumeration class corresponding to aircraft-type-model-code is authsr, meaning Authoritative source, i.e. the list of allowable values is maintained externally from the JC3IEDM model. This seems to make the ontology incomplete; an external source must be referenced. OWL demands that an enumeration class has all its values explicitly asserted, this is not the case here.

5 Reasoning with JC3IEDM: a Case Study

This section shows how it is possible to encode a simple reasoning argument in JC3IEDM using SQL statements. The argument was chosen to illustrate important concepts, limitations and strengths of possible inferences using SQL in this case and OWL in section 6. The operational value of the argument, i.e. how *real-world* it is, is admittedly not very high, but it merely serves as a first step in reasoning toward the real-world arguments given in section 7.

Let us say that:

if a person is assigned to a military-organisation

then he is a military.

This argument is not directly translatable into JC3IEDM. There are numerous intervening tables that encode the necessary knowledge. The reasoning argument can be encoded with the following pseudo-Datalog[5] rule:

PERSON-TYPE(*person-type*, Military,–,–) ←
OBJECT-ITEM-TYPE(*person*, –, *person-type*, –),
PERSON(*person*, –, –, –, –),
OBJECT-ITEM-ASOCIATION(–, *person*, *object*, –, Has on assignment, –),
ORGANISATION(*object*, –),
OBJECT-ITEM-TYPE(*object*, –, *mil-org*, –),
MILITARY-ORGANISATION-TYPE(*mil-org*, –, –).

This is a simplified representation of the actual logical database tables, but it is sufficient for the present example. Variables are denoted by italic text. Constants are either put in single quotes or are written in roman font. Variables that occur only once are called anonymous variables, and stand for a uniquely named variable that does not appear anywhere else in the rule. Anonymous variables are denoted by the symbol –. Thus, the preceding pseudo-Datalog rule will set to Military the category code of a PERSON-TYPE which is used as a classification for a PERSON that is associated to an ORGANISATION which is classified as a MILITARY-ORGANISATION-TYPE.

The preceding may be expressed by a *view* in SQL (see section 2.2.1). A view is a construct that gathers information from different tables, and presents it in a single *virtual* table. The preceding inference may be encoded as a view by the SQL code shown in Fig. 3. The distinction is made between an inference and an active rule. The view does not encode a rule because no information is put back into the database; it just presents the result of an inference.

To update the database with the fact that someone is Military, as deduced from the above rule, it is necessary to use a trigger along with the view defined by Fig. 3. Upon insertion of new information into table object_item_association, the military_person_view is checked to see if there is a new row. If there is, the appropriate row of table person_type is updated with a category-code of Military.

The code presented in Fig. 3 and 4 is a lot more complicated than the above Datalog rule. SQL was designed for a specific purpose, querying data contained in a relational database. It is a set-based declarative query language, and not an imperative language such as C. There are, however, extensions to it that add control-flow construct functionality for example.

[5] Normally, Datalog does not allow the head of the rule to have base predicates. Base predicates correspond to database relations and are defined by the database schema, while derived predicates are defined by rules.

The main advantage of using SQL in the case presented above is that it could be put directly to work on the original JC3IEDM database. Reasoning arguments would be part of the physical implementation, without the need of any external reasoner.

```
1   CREATE VIEW military_person_view AS SELECT
2       person.person_id,
3       object_item_association.object_item_association_object_object_item_id,
4       object_item_association.object_item_association_subject_object_item_id,
5       organisation.organisation_id,
6       object_item_type.object_item_id,
7       object_item_type.object_type_id,
8       military_organisation_type.military_organisation_type_id
9     FROM organisation, object_item_type, military_organisation_type, person, object_item_association
10    WHERE
11        organisation.organisation_id = object_item_type.object_item_id
12      AND
13        military_organisation_type.military_organisation_type_id = object_item_type.object_type_id
14      AND
15        person.person_id = object_item_association.object_item_association_object_object_item_id
16      AND
17       organisation.organisation_id = object_item_association.object_item_association_subject_object_item_id;
```

Figure 3: SQL code to create a view of a military person.

```
1   delimiter $
2   CREATE TRIGGER military_trigger AFTER insert ON object_item_association
3     FOR EACH row
4     BEGIN
5       DECLARE object_type_index int;
6       DECLARE row_count int;
7       SELECT object_item_type.object_type_id INTO object_type_index
8         FROM object_item_type, military_person_view
9         WHERE object_item_type.object_item_id = military_person_view.person_id
10        LIMIT 1;
11      SELECT count(*) INTO row_count FROM military_person_view;
12      IF row_count > 0
13        THEN
14          UPDATE person_type SET person_type_category_code='Military'
15            WHERE person_type.person_type_id = object_type_index;
16      END IF;
17    END $
18  delimiter ;
```

Figure 4: SQL code to create a trigger upon insertion of any new row into table object_item_association. The trigger will update the person_type table by setting the appropriate category-code to Military.

6 Reasoning with P-JC3IEDM: a Case Study

This section presents the same reasoning example from the previous section but using P-JC3IEDM. It is intended as a demonstration of what can be done and how it can be done. It also serves to highlight some of the previous discussions and to show the strengths and weaknesses of P-JC3IEDM in drawing inferences.

The goal is to encode the necessary knowledge in the ontology so the inference engine may automatically classify an individual in any one of the three subclasses of PERSON-TYPE: Military, Civilian or paramilitary. Currently in P-JC3IEDM only subsumption relations exist between classes and their parent. The subsumption relation Military is a PERSON-TYPE represents a necessary condition for an individual to be classified as Military, i.e. it is necessary to be a PERSON-TYPE in order to be a Military. The inference engine, however, must have access to sufficient conditions in order to automatically classify an unknown individual in a particular class. Necessary and sufficient conditions are required for automatic reasoning. These conditions are encoded as `owl:equivalentClass`.

Encoding knowledge in an OWL-DL ontology consists in adding an `owl:equivalentClass` description about already defined classes. The following short example, which does not uses P-JC3IEDM's exact properties but simpler ones, illustrates the case for the Military class only. The reasoning rule is that any person assigned to a military organisation is classified as military.[6] This is depicted in Fig. 5 and by the following rule:

$$\mathsf{Person}(?x) \wedge \mathsf{isAssignedTo}(?x, ?y) \wedge \mathsf{Military\text{-}Organisastion}(?y) \rightarrow \mathsf{Military}(?x).$$

Using the rolling-up technique, it is possible to translate this rule into a class expression that describes an individual as being a member of a class if it has a specific property with some values (i.e. the range has a restriction) that are part of a particular class:

$$\mathsf{Person} \sqcap \exists \mathsf{isAssignedTo}.\mathsf{Military\text{-}Organisastion} \equiv \mathsf{Military}.$$

In this very simple ontology, an individual would be a Military person if it participates in the property isAssignedTo with the restriction that the values this property takes are members of the class Military-Organisation. Thus, it can be stated that an individual must satisfy the following necessary and sufficient conditions:

- a Person *and*
- any object where at least one of the values of the isAssignedTo property is:
  - a Military-Organisation.

Encoding this knowledge in P-JC3IEDM is more complicated due to the presence of associative tables like OBJECT-ITEM-TYPE and the existence of item-type classification trees.

[6] This is obviously just an example and a more complete description should be used.

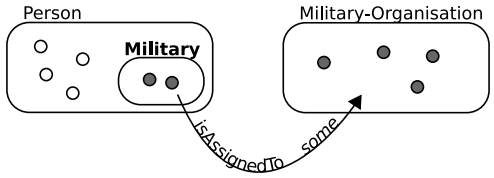

Figure 5: The Military equivalent class defined by the isAssignedTo property.

The equivalent class assertion in P-JC3IEDM is a long chain of property's range restriction on individual's membership. The range of each property is defined not by a single defined class as in the above example, but rather by an anonymous class described by yet another restriction on the property's range.

Since the starting point of the classification is on an OBJECT-TYPE using OBJECT-ITEM, and since the description of an equivalent class is directional (Fig. 7), it becomes necessary to define new inverse relationships. A new property OIT-associates-OI is defined that links an OBJECT-ITEM-TYPE to an OBJECT-ITEM. There exists in P-JC3IEDM the property OI-is-classified-as-OIT that links an OBJECT-ITEM to an OBJECT-ITEM-TYPE, but no inverse property is defined. Similarly, the property OIA-associates-as-subject-OI was added which links an OBJECT-ITEM-ASSOCIATION to an OBJECT-ITEM.

The following reasoning rule describes the equivalent class of Military as being the definition of a PERSON mentioned in a Has_on_assignment association to an ORGANISATION defined as a MILITARY-ORGANISATION-TYPE. In other words an individual is defined as a Military if it satisfies the necessary and sufficient conditions given in Fig. 6 and 7. The latter figure graphically shows the relations between different entities. Rounded rectangles represent P-JC3IEDM classes. They are either named, with the name given just above, or anonymous. An anonymous class is generally defined in terms of a restriction on a property. In the case of the anonymous subclass of PERSON for example, it is defined by the restriction that all its members have the property OI-is-the-object-of-OIA with some values that are members of an anonymous subclass of Has_on_assignment defined itself by a restriction.

The Military equivalent class may be written using description logics notation:

PERSON-TYPE $\sqcap$ $\exists$OT-is-used-as-a-specification-for-OIT.
(OBJECT-ITEM-TYPE $\sqcap$ $\exists$OIT-associates-OI.
(PERSON $\sqcap$ $\exists$OI-is-the-object-of-OIA.
(Has_on_assignment $\sqcap$ $\exists$OIA-asociates-as-subject-OI.
(ORGANISATION $\sqcap$ $\exists$OI-is-classified-as-OIT.
(OBJECT-ITEM-TYPE $\sqcap$ $\exists$OIT-associates-OT.MILITARY-ORGANISATION-TYPE))))))
$\equiv$ Military

Some sample facts needed in the knowledge base for the reasoning rule to fire are given below. First, two OBJECT-ITEM, *Alex* and *Land_Force*, must be added and their type defined

by asserting they have a property with values an OBJECT-ITEM-TYPE, *OIT-0001* and *OIT-0002* respectively:

Individual(*Alex* type(PERSON) value(OI-is-classified-as-OIT *OIT-0001*))

Individual(*Land_Force* type(ORGANISATION) value(OI-is-classified-as-OIT *OIT-0002*))

The OBJECT-ITEM-TYPE individuals are asserted and linked to object-type individuals:

Individual(*OIT-0001* type(OBJECT-ITEM-TYPE) value(OIT-associates-OT *PERSON-TYPE-0001*))

Individual(*OIT-0002* type(OBJECT-ITEM-TYPE) value(OIT-associates-OT *Canadian_Military*))

Individual(*PERSON-TYPE-0001* type(PERSON-TYPE))

Individual(*Canadian_Military* type(MILITARY-ORGANISATION-TYPE))

Finally, the association between individual Alex and individual Land_Force is asserted like this:

Individual(*OIA-0001* type(Has_on_assignment) value(OIA-associates-as-object-OI *Alex*) value(OIA-associates-as-subject-OI *Land_Force*))

When these facts are present in the knowledge base, the inference engine concludes that PERSON-TYPE-0001 is of type Military.

An individual is classified as a Military if it satisfies the following necessary and sufficient conditions; it is:

- a PERSON-TYPE *and*
- any object where at least one of the values of the OT-is-used-as-a-classification-for-OIT property is:
  - an OBJECT-ITEM-TYPE *and*
  - any object where at least one of the values of the OIT-associates-OI property is:
    - a PERSON *and*
    - any object where at least one of the values of the OI-is-the-object-of-OIA property is:
      - a Has_on_assignment (subclass of OBJECT-ITEM-ASSOCIATION) *and*
      - any object where at least one of the values of the OIA-associates-as-subject-OI property is:
        - an ORGANISATION *and*
        - any object where at least one of the values of the OI-is-classified-as-OIT property is:
          - an OBJECT-ITEM-TYPE *and*
          - any object where at least one of the values of the OIT-associates-OT property is:
            - a MILITARY-ORGANISATION-TYPE.

Figure 6: The Military equivalent class.

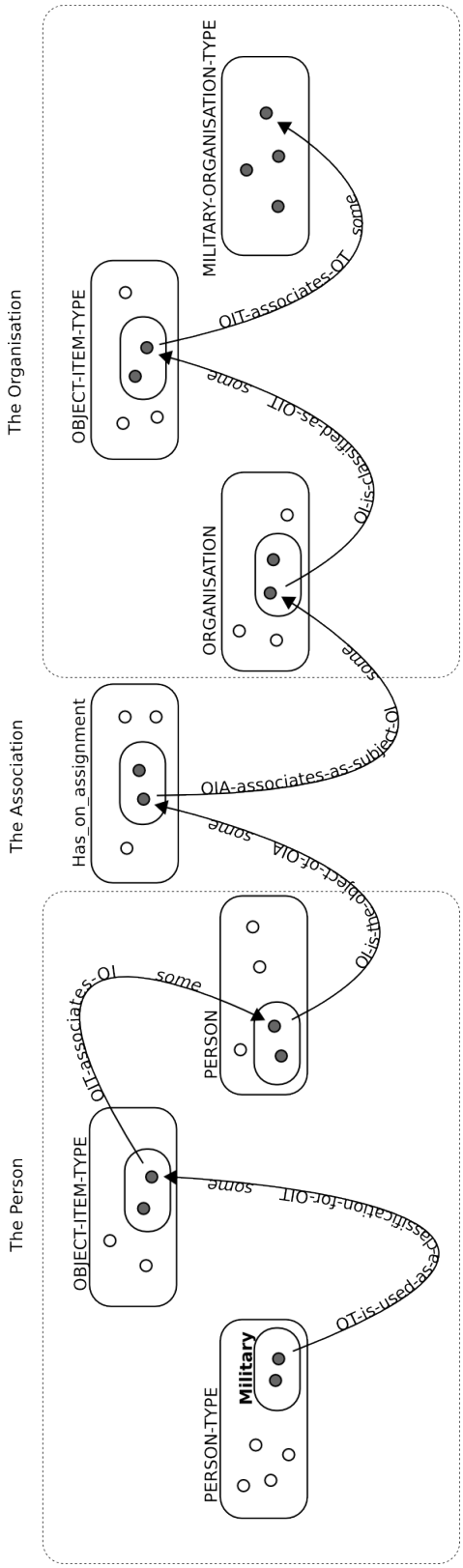

Figure 7: The Military equivalent class.

6.1 Discussion and Proposed Modifications to P-JC3IEDM

This section is intended as a discussion of the knowledge representation capabilities of P-JC3IEDM through the analysis of the above example. It details the observations and consequences of the above inference. The following is divided into three sections. The modifications proposed in each are given in order of increasing complexity, i.e. how difficult it would be to modify P-JC3IEDM. Whenever recommendations are made they are enclosed in a box. The first section suggests adding property characteristics, the second section suggests modifying how P-JC3IEDM is built and the third section suggests a way forward for translating the conceptual model of JC3IEDM into OWL.

6.1.1 Minimal Modifications: Adding Missing Properties

The first set of modifications are intended to be minimal and allow the inference engine to draw simple inferences like the one presented in section 6. The addition of two property characteristics are suggested (see section 2.3.1.1 for some examples).

In order to reason on a type, new relations (i.e. OWL properties) need to be added to the ontology. JC3IEDM does not define the inverse of the relationship *is-classified-as* between an OBJECT-ITEM and an OBJECT-ITEM-TYPE. The need to define the inverse of the previous relationship as the OWL property OIT-associates-OI arises from the need to define an anonymous subclass of OBJECT-ITEM-TYPE in terms of a restriction on a property having a domain of OBJECT-ITEM-TYPE and a range of OBJECT-ITEM (see Fig. 6 and 7). Similarly, other properties need to be added for reasoning on other entities like OBJECT-ITEM-ASSOCIATION. In general, since JC3IEDM does not always define the inverse of a relationship, P-JC3IEDM should add it.

> Undefined JC3IEDM's inverse relationships should be added to P-JC3IEDM as new properties. This operation could be automated by the translation script.

In JC3IEDM, the instances of OBJECT-TYPE are used as classification by many OBJECT-ITEM instances. If two persons are military, then they will both have an entry in the OBJECT-ITEM-TYPE pointing to the same OBJECT-TYPE instance.[7] This cannot be in the previous example since OWL's classification is done on individuals and P-JC3IEDM uses individuals (OBJECT-TYPE) to define others (OBJECT-ITEM). If more than one individual, *Alex* and *Bob*, use the same individual *PERSON-TYPE-0001*, and that individual's class membership changes, then both *Alex* and *Bob* will have their definition changed.

> The OT-is-used-as-a-classification-for-OIT object property that links an OBJECT-TYPE to an OBJECT-ITEM-TYPE should be functional in order to have one OBJECT-TYPE individual for each OBJECT-ITEM-TYPE individual. In other words, the relation between an individual OBJECT-TYPE and an individual OBJECT-ITEM-TYPE should be one-to-one. This way, the inference engine can change the classification of one individual without affecting any other.

[7] This is not strictly true since many types of military may be defined.

The previous prescription is a direct consequence of having OBJECT-TYPE for classifying OBJECT-ITEM. This design has the disadvantage of:

- complicating the classification of items by requiring associative classes like OBJECT-ITEM-TYPE and numerous properties for linking the associative table to the individuals
- hampering the reasoning power of OWL, notably by making the classification based on property domain and range impossible since most properties relate an object with a common associative table (see section 4.2.2)
- requiring complex equivalent class definitions.

There is, however, one important advantage to such a design. It allows an OBJECT-ITEM to have more than one property with the value of an OBJECT-ITEM-TYPE enabling more than one type to be defined for an item, obviously subject to constraints of the ontology such as disjoint classes. If the type of an item were to change over time, it would be possible to keep the pedigree by adding a new OBJECT-ITEM-TYPE individual linked to a new OBJECT-TYPE. This design partly makes up for the monotonic nature of OWL, by which adding a fact to the knowledge base never changes what has already been inferred.

6.1.2 Improving and Increasing the Semantic Richness of P-JC3IEDM

Asserting the existence of an individual of type PERSON-TYPE amounts to declaring that the individual is a *not known* PERSON-TYPE. If that individual was of a known type, it would be classified as one of PERSON-TYPE's subclasses. The subclass Not_Known does not further specialize the class PERSON-TYPE. Moreover, if *PERSON-TYPE-0001* were declared of type Not_Known, the inference engine would infer as per the above equivalent class definition that it is both a Military and a Not_Known, since these classes are not disjoint. By default, OWL considers that classes can overlap, so that an individual may be a member of two classes. In this case, the dual membership is confusing since a PERSON cannot be at the same time Not_Known and Military.

> Classes translated from category-code *Not known* should not be used in P-JC3IEDM. Unknown individuals should be members of the base class, translated from the corresponding entity. The automatic translation of P-JC3IEDM should omit all category-codes *Not known*.

The category-code *not otherwise specified* has the meaning of *not member of all other subclasses*. For example, the class Not_otherwise_specified subclass of PERSON-TYPE has the equivalent definition of:

not (Civilian or Military or Paramilitary)

In OWL, this is asserted using the constructor complementOf. Thus, an individual that is a member of Not_otherwise_specified is also a member of the class defined by the complement of the union of Civilian, Military and Paramilitary.

Classes translated from the category-code *Not otherwise specified* should be the complement of the union of all their sibling classes, excluding the Not_known class. This could probably be done automatically by the translation script.

In P-JC3IEDM, a Military is not disjoint from a Civilian. The fact that a MILITARY cannot be a Civilian should be reflected in the ontology. Not only would it be more realistic, but it would allow the inference engine to detect an inconsistent knowledge base. Any PERSON classified via an OBJECT-ITEM-TYPE as a Military would become inconsistent if new information were entered in the knowledge base asserting that the same person was instead classified as a Civilian. By asserting mutually exclusive classes, the inference engine can detect conflicts and allow their resolution by the operator.

Whenever possible (see section 7.3), mutually exclusive classes of JC3IEDM should be asserted as disjoint in P-JC3IEDM. This cannot be automatically implemented and requires human intervention after the automatic translation.

The relation between OBJECT-ITEM and OBJECT-TYPE could be simplified without losing the advantage described in section 4.2.1. OWL allows many-to-many relations, and associative tables introduced into JC3IEDM because the E-R model cannot deal with many-to-many relations are not necessary as long as the n-ary representation pattern is not needed. Recall that this pattern is useful when more than two individuals need to be linked; it consists in making the relation into a class with attributes having as their value the other participants of the relation (the third, fourth, etc.). Since the OBJECT-ITEM-TYPE does not profit from the n-ary representation pattern, it could be removed. The OBJECT-ITEM would then be linked directly to an OBJECT-TYPE. This does not pose any problems because there must be one OBJECT-TYPE individual for each possible report. The REPORTING-DATA individual would also be linked directly to the OBJECT-TYPE. The properties linked with the OBJECT-ITEM-TYPE would be replaced by properties directly linking an OBJECT-ITEM to an OBJECT-TYPE. The immediate benefit would be a simpler knowledge representation.

6.1.3 Toward an Exact Translation of JC3IEDM into OWL-DL

As mentioned already, the best approach for building an ontology that will serve primarily as the basis for inferring is to start by basing the conceptual model on the reasoning that needs to be supported. Obviously, the strengths and limitations of the knowledge representation language used needs to be taken into account right from the beginning. For example, a class Hostile would be needed if any such classification were to be done in OWL. Otherwise, it could just be an attribute in some other language, as long as that language can conclude using attributes (OWL cannot).

There are other reasons besides reasoning limitations for rebuilding P-JC3IEDM. Business rules could be captured using a more expressive KR language than the E-R model. Knowledge contained in text descriptions could also be captured. Note that this would enrich

the ontology without necessarily making it apt to reason any more than P-JC3IEDM. Such endeavours have been undertaken before. LC2IEDM, for example, was translated into a frame-based ontology by Loaiza et al, described in [15]. The authors even went so far as to describe what a length is, that a length times a length gives an area and so on. The semantic of what a length is was explicitly captured. Their resulting ontology includes all entities, attributes and relations, and also selected high-level concepts of the command and control ontology such as the definition of a threat. This kind of work, however, cannot be done by an automatic translation of an E-R diagram; not enough knowledge is expressible using E-R diagrams.

7 Representing Elements of a Scenario with JC3IEDM and P-JC3IEDM

This section shows how the reasoning arguments of sections 4.1 and 4.2 of Task 3 can be encoded in JC3IEDM and P-JC3IEDM ontologies. The premise numbering below refers to the numbering used in Task 3 section 4.1.1 and 4.2.1.

7.1 Encoding Premises of the Mechanical Incident Scenario

The mechanical incident scenario was described in the Task 3 report [1], section 4.1. In this scenario, a mounted unit must travel along an isolated route to investigate a civilian vehicle incident. While enroute, they are stopped by a flat tire between two villages in a potentially dangerous area with possible hostiles living nearby. They must assess the situation and decide on a course of action.

Most conclusions of the premises of this scenario involve some action that is deemed unsafe or some person or group that is deemed suspect. An unsafe action can be encoded using the attribute action-task-entailed-safety-degree-code of entity ACTION-TASK. This attribute is defined as the specific value that represents the degree of safety or risk an ordered operation entails. The definition of an ACTION-TASK is "an ACTION that is being or has been planned and for which the planning details are known" [16]. The hostility of a person or group is encoded by the attribute object-item-status-hostility-code of OBJECT-ITEM-STATUS. It takes values like *Assumed friend*, *Friend*, *Hostile*, *Neutral*, *Faker*, *Unknown*, etc.

In the following the premise is stated first, followed by the possible encoding of the rule in JC3IEDM. For this, the premise is encoded using a pseudo-Datalog notation where predicate names are taken from the entity names and the arguments of the predicate are the attributes of the entity. Arguments in italic are variables that have to match over the whole rule. Anonymous variables are denoted by –. These are variables that appear only once in the rule and may take any value. The ordering of the attributes of a predicate is the same as in the HTML documentation of JC3IEDM Edition 3.0.

Since OWL cannot conclude on a specific value of an object property, and since all premises of Task 3 do conclude on a specific value of a property, it follows that all the premises of

Task 3 are not encodable in P-JC3IEDM. A typical OWL-DL condition like the following:

$$\text{Person} \sqcap \exists \text{isAssignedTo.Military-Organisation} \equiv \text{Military}.$$

expresses the fact that there exists a class Military that is equivalently described by every object having the property isAssignedTo with value Military-Organisation and which is also a member of class Person. What should be noted here, is that the premise concludes on the membership of a class, i.e. Military, and not on a value of a property that an individual should have. P-JC3IEDM needs to be modified in order to encode any of the premises of Task 3. The proposed modification is to turn any attribute for which the premise needs to conclude into a class. For example, to conclude that someone is suspect, it would be necessary to have a class Object-Item-Status-Suspect and assign that person to that class. In the following, the premises will be encoded as necessary and sufficient conditions using description logic notation and all needed attributes will be turned into classes.

7.1.1 Premise Number 1

if person P is suspect *and* action A involves person P

then action A is unsafe.

7.1.1.1 JC3IEDM

The body of the following pseudo-Datalog[8] rule says that there must be an entry in table ACTION-RESOURCE-ITEM with two keys named *action* and *person* that must be matched in the head and body of the rule, respectively. For the key *person*, the same entry must be present in both the PERSON table, and in the OBJECT-ITEM-STATUS table with the attribute Suspect on the same line. When all of the body keys are matched, the head will take effect, stating here that a line with key *action* and attribute Unsafe must be present in table ACTION-TASK.

$$\begin{aligned}
&\text{ACTION-TASK}(\ \textit{action}, \ldots, -, \text{Unsafe}, -, \ldots, -\ ) \ \leftarrow \\
&\qquad \text{ACTION-RESOURCE-ITEM}(\ \textit{action}, -, \textit{person}\ ) \\
&\qquad \text{PERSON}(\ \textit{person}, -, -, -, -\ ), \\
&\qquad \text{OBJECT-ITEM-STATUS}(\ \textit{person}, -, -, -, -, \text{Suspect}, -\ ),
\end{aligned}$$

The corresponding JC3IEDM tables for the antecedent are presented in Figures 8a, 8b and 8c. The consequent table is given in 8d. Only the relevant attributes are presented, those that need to be matched across the tables. The attributes that can take any values are written as –.

[8] Pure Datalog is limited in its manipulation of the stored data, called an extensional database. It cannot add back to the extensional database what has been inferred. Instead, the conclusion is kept in what is known as the intensional database. These considerations become important at the implementation stage.

action-id	***-index	object-item-id
4 [*action*]	1	23 [*person*]

(a) ACTION-RESOURCE-ITEM

***-id	***-birth-datetime	***-blood-type-code	***-gender-code	***-professing-indicator-code
23 [*person*]	–	–	–	–

(b) PERSON

***-id	***-index	***-hostility-code	reporting-data-id
23 [*person*]	–	Suspect	–

(c) OBJECT-ITEM-STATUS

action-task-id	***-activity-code	***-category-code	***-entailed-safety-degree-code
4 [*action*]	–	–	Suspect

(d) ACTION-TASK

Figure 8: SQL tables for the ACTION-TASK rule.

7.1.1.2 P-JC3IEDM

It is necessary to introduce new classes to express this premise. P-JC3IEDM has an OBJECT-ITEM-STATUS attribute called object-item-status-hostility-code giving the hostility of an OBJECT-ITEM. Here, it is necessary to turn each value of that attribute into its own class inheriting from owl:Thing.[9]

Encoding knowledge in OWL for an inference engine to infer new class membership is done by adding equivalent class descriptions to the ontology. Equivalent classes are defined by necessary and sufficient conditions of membership.

$$\begin{aligned}&\text{ACTION-TASK} \sqcap \exists \text{A-requires-AR}.\\ &\quad(\text{ACTION-RESOURCE-ITEM} \sqcap \exists \text{ARI-is-specification-of-OI}.\\ &\quad(\text{PERSON} \sqcap \text{Object-Item-Status-Suspect}))\\ &\equiv \text{Action-Task-Unsafe}\end{aligned}$$

7.1.2 Premise Number 2

if group G is suspect *and* action A involves group G

then action A is unsafe.

[9]Other schema could be envisaged; however for this simple proof-of-concept, inheriting from owl:Thing will be sufficient.

7.1.2.1 JC3IEDM

ACTION-TASK(*action*, . . . , –, Unsafe, –, . . . , –) ←
ACTION-RESOURCE-ITEM(*action*, –, –, –, *group*)
ORGANISATION(*group*, –),
OBJECT-ITEM-STATUS(*group*, –, –, –, –, Suspect, –),

7.1.2.2 P-JC3IEDM

ACTION-TASK ⊓ ∃A-requires-AR.
(ACTION-RESOURCE-ITEM ⊓ ∃ARI-is-specification-of-OI.
(ORGANISATION ⊓ Object-Item-Status-Suspect))
≡ Action-Task-Unsafe

7.1.3 Premise Number 7

if group G is suspect *and* person P is a member of group G
then person P is suspect.

7.1.3.1 JC3IEDM

OBJECT-ITEM-STATUS(*person*, –, –, –, –, Suspect, –) ←
PERSON(*person*, –, –, –, –),
OBJECT-ITEM-ASSOCIATION(–, *person*, *group*, –, Has as a member, –),
ORGANISATION(*group*, –),
OBJECT-ITEM-STATUS(*group*, –, –, –, –, Suspect, –)

7.1.3.2 P-JC3IEDM

P-JC3IEDM does not define an object property having a domain of OBJECT-ITEM-ASSOCIATION and a range of OBJECT-ITEM because JC3IEDM has no inverse relation defined for OBJECT-ITEM is-the-subject-of OBJECT-ITEM-ASSOCIATION. It is necessary to have such an inverse relation to encode the above premise. This relation will be called OIA-associates-as-subject-OI having a domain of OBJECT-ITEM-ASSOCIATION and a range of OBJECT-ITEM.

PERSON ⊓ ∃OI-is-the-object-of-OIA.
(Has_as_a_member ⊓ ∃OIA-associates-as-subject-OI.
(ORGANISATION ⊓ Object-Item-Status-Suspect))
≡ Object-Item-Status-Suspect

7.1.4 Premise Number 8

if person P is suspect *and* person P is a member of group G

then group G is suspect.[10]

7.1.4.1 JC3IEDM

OBJECT-ITEM-STATUS(*group*, –, –, –, –, Suspect, –) ←
ORGANISATION(*group*, –),
OBJECT-ITEM-ASSOCIATION(–, *person*, *group*, –, Has as a member, –),
PERSON(*person*, –, –, –, –),
OBJECT-ITEM-STATUS(*person*, –, –, –, –, Suspect, –)

7.1.4.2 P-JC3IEDM

The inverse relation OIA-associates-as-object-OI is added to encode premise 8 (see comment to premise 7).

ORGANISATION ⊓ ∃OI-is-the-subject-of-OIA.
(Has_as_a_member ⊓ ∃OIA-associates-as-object-OI.
(PERSON ⊓ Object-Item-Status-Suspect))
≡ Object-Item-Status-Suspect

7.2 Encoding Premises of the Mission Planning Scenario

This scenario was described in the Task 3 report [1], section 4.2. It describes an expert system that helps a tactical commander prepare a mission plan. Suppose that the tactical commander must investigate an incident at some village. He must plan his route, prepare his equipment, and learn about the place where he is going. He inputs to the expert system via the interface his intended itinerary and other information and the system replies by showing relevant information as determined by its rules.

7.2.1 Premise Number 1

if a known terrorist K has a relative R

then relative R is considered suspect.

There is more than one way of defining a terrorist. A terrorist might be defined simply as someone who is suspect (i.e. having a hostility-code of suspect), or it might be someone affiliated with a terrorist group. The latter will be used. A relative needs to be defined also.

[10] This premise is probably too harsh, but is used here to illustrate some important consequences of having two premises opposite one another (premises 7 and 8).

JC3IEDM has many category-codes of OBJECT-ITEM-ASSOCIATION that correspond to the definition of relative, e.g. *Is aunt of*, *Is brother of*, *Is cousin of*, *Is daughter of*, etc. To fully implement this rule, all of the category-code values would need to be taken into account. The first two examples are given, namely *Is aunt of* and *Is brother of*.

7.2.1.1 JC3IEDM

OBJECT-ITEM-STATUS(*subject*, –, –, –, –, Suspect, –) ←
PERSON(*subject*, –, –, –, –),
OBJECT-ITEM-ASSOCIATION(–, *object*, *subject*, –, Is aunt of, –),
PERSON(*object*, –, –, –, –),
OBJECT-ITEM-AFFILIATION(*affiliation*, –, *object*, –),
AFFILIATION-FUNCTIONAL-GROUP(*affiliation*, Terrorist, –),

OBJECT-ITEM-STATUS(*subject*, –, –, –, –, Suspect, –) ←
PERSON(*subject*, –, –, –, –),
OBJECT-ITEM-ASSOCIATION(–, *object*, *subject*, –, Is brother of, –),
PERSON(*object*, –, –, –, –),
OBJECT-ITEM-AFFILIATION(*affiliation*, –, *object*, –),
AFFILIATION-FUNCTIONAL-GROUP(*affiliation*, Terrorist, –),

Adding all possible values of the category-code corresponding to a relative completes the encoding of the premise.

7.2.1.2 P-JC3IEDM

The affiliation-functional-group-code is needed as an OWL class; thus Affiliation-Functional-Group-Terrorist represents the class of every terrorist affiliation group.

PERSON ⊓ ∃OI-is-the-subject-of-OIA.
(Is_aunt_of ⊓ ∃OIA-associates-as-object-OI.
(PERSON ⊓ ∃OI-has-OIA.
(OBJECT-ITEM-AFFILIATION ⊓ ∃OIA-is-referenced-to-A.
(AFFILIATION-FUNCTIONAL-GROUP ⊓ Affiliation-Functional-Group-Terrorist))))
≡ Object-Item-Status-Suspect

PERSON ⊓ ∃OI-is-the-subject-of-OIA.
(Is_brother_of ⊓ ∃OIA-associates-as-object-OI.
(PERSON ⊓ ∃OI-has-OIAf.
(OBJECT-ITEM-AFFILIATION ⊓ ∃OIAf-is-referenced-to-A.
(AFFILIATION-FUNCTIONAL-GROUP ⊓ Affiliation-Functional-Group-Terrorist))))
≡ Object-Item-Status-Suspect

Care must be taken when asserting the above conditions that they are evaluated independently. They should not be linked by logical conjunctions since no conclusion would ever be reached.

7.2.2 Premises Number 3, 6, 9 and 11

3. **if** house H is suspect and house H is within area A of plan P
   then house H is relevant to plan P.

6. **if** route R is unsafe and route R is within area A of plan P
   then route R is relevant to plan P.

9. **if** village V is unsafe and village V is within area A of plan P
   then village V is relevant to plan P.

11. **if** location T is unsafe and location T is within area A of plan P
    then location T is relevant to plan P.

These premises are all in the form:

if OBJECT-ITEM/LOCATION is unsafe and OBJECT-ITEM/LOCATION is within area A of plan P
then OBJECT-ITEM/LOCATION is relevant to plan P.

This last premise will be used to try and encode in JC3IEDM and P-JC3IEDM.

7.2.2.1 JC3IEDM

Several tables are needed to encode the relevancy of an OBJECT-ITEM. An interesting table to use would have been the CONTEXT-OBJECT-ITEM-ASSOCIATION for which there exists a category-code with a value of *Is relevant to*, meaning that the "specific CONTEXT has significance with respect to a specific OBJECT-ITEM"; a CONTEXT is relevant to an OBJECT-ITEM. The semantic of the relation, however, is backward. The subject of the premises above is the OBJECT-ITEM, not the CONTEXT (a context could have been linked to a plan). Twisting the semantic of the model to use a CONTEXT-OBJECT-ITEM-ASSOCIATION could cause problems. Instead, a CONTEXT will be used to agglomerate all things relevant and they will be linked to a plan, represented by an ACTION-TASK, using an ACTION-CONTEXT entity.

For example, let the first three premises above be true for a specific plan P, represented by an ACTION-TASK. Tables 9b – 9f give the necessary information to encode the relevancy of some of the OBJECT-ITEM to a plan without any mention of location and area (these will be discussed below). An ACTION-TASK is defined in Table 9a that represents a plan P and three OBJECT-ITEM are defined in Table 9b: a house, a route and a village. Their status (Suspect, Table 9c) and type (Facility, Table 9d) are defined and they are all reported as fact (Table 9e). A CONTEXT is used to link all three OBJECT-ITEM through their REPORTING-DATA. The meaning of the context is given by the context-name-text attribute; here it is *Relevant Items*. The context is then linked to the ACTION-TASK using ACTION-CONTEXT with a category-code of *Initial state, actual* meaning that some information was observed as factual and was deemed relevant at the initial state of some of the ACTION-TASK.

Table 9a encodes the fact that some objects are relevant to some of the action tasks or plans, but not the fact that these objects must be within the radius of where the action is taking place. It seems there is no way to encode something like in JC3IEDM, that this OBJECT-ITEM is within the area of action of some of the ACTION-TASK. The information to find out whether or not this is the case is present in the schema, but is not encoded directly. For example, an ACTION-LOCATION enables the geographic position of an ACTION to be specified, and an object-item-location gives the location of an OBJECT-ITEM. By comparing both locations, it is possible to determine whether or not an OBJECT-ITEM is within the area of action of a plan. Using Datalog, the body of the rule could be:

OBJECT-ITEM(*object*, –, –),
OBJECT-ITEM-LOCATION(*point*, *object*, –, . . .),
LOCATION(*point*, –)
IsInArea(*point*,*area*),
LOCATION(*area*, –),
ACTION-LOCATION(*plan*, –, *area*, –, –, –),
ACTION(*plan*, –, –),

where IsInArea is a procedure called by the inference engine to check whether *point* is within *area*.[11] A Datalog rule will fire if there exists a set of rows that match the body, i.e. the variables of each body's predicate matches lines of the corresponding tables. In the above rule, *plan* and *area* are linked with ACTION-LOCATION, and *object* and *point* are linked with OBJECT-ITEM-LOCATION. The link between the first three predicates and the last three is provided by the procedure; there is no table in JC3IEDM that links a point and an area with the attached meaning that the point is part of the area. Thus the procedure is necessary to make that link.

Datalog does not allow multi-predicate heads; the conclusion must be about only one table. Premises 3, 6, 9 and 11 conclude on the relevancy using three tables: CONTEXT,

[11] There exist extensions to Datalog that allow procedures in rule bodies. The system called XSB, among others, does it.

action-task-id	***-activity-code	***-category-code	***-entailed-safety-degree-code
at4	–	–	–

(a) ACTION-TASK

object-item-id	***-category-code	***-name-text
oi6	Facility	Suspect House
oi7	Facility	Suspect Route
oi8	Facility	Suspect Village

(b) OBJECT-ITEM

object-item-id	***-index	***-hostility-code	reporting-data-id
oi6	1	Suspect	rd2
oi7	2	Suspect	rd3
oi8	3	Suspect	rd4

(c) OBJECT-ITEM-STATUS

object-item-id	object-item-type-index	object-type-id	reporting-data-id
oi6	[House]	5	rd2
oi7	[Road]	6	rd3
oi8	[Village]	7	rd4

(d) OBJECT-ITEM-TYPE

reporting-data-id	***-category-code	***-credibility-code	***-reporting-datetime	***-reporting-organisation-id
rd2	Reported	Reported as a fact	20070816091000.000	[ASC]
rd3	Reported	Reported as a fact	20070818093000.000	[ASC]
rd4	Reported	Reported as a fact	20070817094000.000	[ASC]

(e) REPORTING-DATA

context-id	context-category-code	context-name-text	context-security-classification-code
c75	Not otherwise specified	Relevant Items	NATO SECRET

(f) CONTEXT

context-id	context-element-index	reporting-data-id
c75	1	rd2
c75	2	rd3
c75	3	rd4

(g) CONTEXT-ELEMENT

action-id	context-id	***-index	***-category-code
at4	c75	5	Initial state, actual

(h) ACTION-CONTEXT

Figure 9: Tables to encode part of premises 3, 6 and 9 of the mission planning scenario

CONTEXT-ELEMENT and ACTION-CONTEXT. When a premise is true, an entry is added to table CONTEXT-ELEMENT, but entries to tables CONTEXT and ACTION-CONTEXT must already exist, else they must be added. In the following Datalog rule, it is assumed that entries in these last two tables already exist. Other rules or some other mechanism, would be needed to create them.

The complete rule needed to add an element to a context associated with an action is:

CONTEXT-ELEMENT(*relevant-context*, –, *report-data*) ←
REPORTING-DATA(*report-data*, –, –, –, –),
CONTEXT(*relevant-context*, –, –, –),
ACTION-CONTEXT(*plan*, *relevant-context*, –, Initial state, actual),
ACTION(*plan*, –, –)
ACTION-LOCATION(*plan*, –, *area*, –, –, –),
LOCATION(*area*, –),
IsInArea(*point*, *area*),
LOCATION(*point*, –),
OBJECT-ITEM-LOCATION(*point*, *object*, –, . . .),
OBJECT-ITEM(*object*, –, –),
OBJECT-ITEM-STATUS(*object*, –, –, –, –, Suspect, *report-data*),

The above rule is complex since it involves a procedure (IsInArea) and will not fire as long as there are no entries in tables CONTEXT and ACTION-CONTEXT.

7.2.2.2 P-JC3IEDM

The major obstacle to encoding these premises in P-JC3IEDM is the nature of the conclusion about the relation CONTEXT-ELEMENT. The inference is not about classifying an object into a class, but rather about establishing a relation between a CONTEXT (*relevant-context* in the Datalog rule above) and a REPORTING-DATA (*report-data*) via the CONTEXT-ELEMENT. In P-JC3IEDM, the relation would be established using two object properties. The first, C-has-as-constituent-part-CE, is between a CONTEXT and a CONTEXT-ELEMENT, and the second, CE-is-referenced-to-RD, is between a CONTEXT-ELEMENT and a REPORTING-DATA. The inference is about being able to establish these relations, i.e. to conclude about object properties, but OWL cannot do that. Even if the relation were direct, something like CONTEXT being referenced to REPORTING-DATA, it would still be impossible to conclude using OWL.

In all the preceding examples of P-JC3IEDM encoding, it was possible to create a class, e.g. Object-Item-Status-Suspect, and use that to infer the type of instance. It is not possible here to create a class like Relevant, for example, and put all relevant REPORTING-DATA into it. One such class would be needed for each plan that was ever conceived. The relation between a CONTEXT and a REPORTING-DATA is binary, i.e. two individuals are involved, and not a unary property like class membership.

Thus, it is not possible to encode these premises in P-JC3IEDM. More generally, any premises that conclude about an association of two things, like the relevancy, will be represented in OWL using an object property, and that cannot lend itself to automatic reasoning. Reasoning in OWL is about type inferencing, i.e. putting individuals into classes.

7.3 Discussion

It is not clear how numerous OWL equivalent class descriptions will interact when applied on a single class. This would be the case for premises 7 and 8 of the mechanical incident scenario. They both assert necessary and sufficient conditions on the Object-Item-Status-Suspect class. Premise 7 states that some individual member of PERSON are also members of Object-Item-Status-Suspect, while premise 8 states that some ORGANISATION are also members of Object-Item-Status-Suspect. These two conditions are asserted as necessary and sufficient conditions and when combined, have the effect of preventing the inference engine from concluding on the hostility of any object. This can be seen by focusing attention on the necessary parts of the conditions which, when both conditions are combined using a conjunction, state that it is necessary to be a PERSON *and* an ORGANISATION in order to be an Object-Item-Status-Suspect. Thus, if an object is asserted as being a PERSON, it will never be classified as Object-Item-Status-Suspect since it is not also an ORGANISATION. This problem can be solved by removing the restriction on the domain and range of the relations involved in the premises. Premise 7 would become:

$$\begin{aligned}&\exists \text{OI-is-the-object-of-OIA}.\\ &\quad(\text{Has\_as\_a\_member} \sqcap \exists \text{OIA-associates-as-subject-OI}.\\ &\quad(\text{ORGANISATION} \sqcap \text{Object-Item-Status-Suspect}))\\ &\equiv \text{Object-Item-Status-Suspect}\end{aligned}$$

and premise 8 would become:

$$\begin{aligned}&\exists \text{OI-is-the-subject-of-OIA}.\\ &\quad(\text{Has\_as\_a\_member} \sqcap \exists \text{OIA-associates-as-object-OI}.\\ &\quad(\text{PERSON} \sqcap \text{Object-Item-Status-Suspect}))\\ &\equiv \text{Object-Item-Status-Suspect}\end{aligned}$$

These conditions have no restrictions on the kind of object that participates in the relations OI-is-the-object-of-OIA and OI-is-the-subject-of-OIA, respectively, i.e. it is not necessary to be a PERSON to have the relation OI-is-the-object-of-OIA with an OBJECT-ITEM-ASSOCIATION. The inference engine then concludes that a PERSON is suspect when premise 7 is fulfilled. Removing all domain and range assertions does not have many consequences. These assertions are not integrity constraints that must be enforced to assert a specific relation between two individuals; rather they are used by the inference engine to deduce the type of the individuals participating in a relation.

Disjointedness is also problematic when combined with assertions on the domain and range of the relations. When PERSON is asserted as disjoint from ORGANISATION and the domain

and range of premises 7 and 8 are asserted, both make the ontology inconsistent since an object cannot be a PERSON and an ORGANISATION at the same time. In other words, the necessary condition of being a PERSON and an ORGANISATION to be suspect clashes with the disjointedness. This problem is solved by removing the domain and range assertions of the relations and adding the disjoint axioms or vice versa, i.e. keeping the domain and range and not asserting disjoint axioms. Again, it is not clear at this point which approach would be better, although as mentioned above, it appears that asserting disjointedness and removing domain and range assertions would yield better inference results. Further research is needed.

The proof-of-concept will surely uncover other problems related to the combinations of the above premises. It seems necessary to modify P-JC3IEDM to reason with it, not only to add new classes but also to modify some of the initial assertions.

8 Conclusions

Considering the reasoning restrictions of OWL, it appears that an ontology built for any other purposes than inference is hard to adapt to the intricacies of OWL. And even if that ontology is built with some reasoning in mind, it might not be the kind of reasoning or appropriate conclusions that OWL supports. Any ontologies must be built with the specific inferences to be performed in mind and take into account the limitations of the specific KR language used. P-JC3IEDM is not built for automatic reasoning but for data exchange, and this is reflected in the ontology. As a consequence it is difficult to reason with it.

Modifying the whole of P-JC3IEDM to accommodate reasoning would be beyond the scope of this project. Specific inferences would need to be identified and the conceptual model modified accordingly. Obviously, this solution has many shortcomings. This would lead to the existence of two models: one for reasoning and one for exchanging information. It is not clear how the reasoning model would evolve with the addition of new reasoning arguments. Would the conceptual model need to be modified with each new set of arguments? The proof-of-concept that will be built in Task 8 will certainly shed some light on these problems and others.

Not all Task 3 premises are encodable using OWL constructs. Rule-based languages that extend OWL like SWRL might be necessary to encode them. Although more expressive, SWRL is still built upon OWL and has all the same limitations explained above. But perhaps more importantly it is undecidable. There exist other frameworks that combine OWL and rules; AL-Log, CARIN and DL-safe rules [7] are some of them. None of them, however, is robustly decidable. It must be noted that decidability is opposed by expressivity. The more expressive a language is, the lesser the chances are that it be decidable. Expressivity must be limited for a language to be decidable.

Ideally, reasoning should be done directly on the instance database. Thus, it might be easier to reason with JC3IEDM using other mechanisms than OWL. The field of deductive databases contains such other mechanisms and seems a promising avenue, one unfortunately

that has not been explored much in this study. Deductive databases represent databases that may interact with the user using languages that are more expressive and overall more powerful than SQL. They are logic-based languages that subsume relational calculus, the most popular being Datalog. One interesting system implementing deductive technology is XSB. It is a logic programming and deductive database system for Unix and MS Windows. It implements an interface that allows one to call data stored in a database and to translate Datalog rules into SQL automatically. An inference engine should reason on JC3IEDM's instance database without the need to modify it.

References

[1] Lockheed Martin Canada (2007), SATAC Knowledge Representation and Automated Reasoning with JC3IEDM, Task 3—Development of Situation Analysis Examples with a Tactical Commander, Contractor Report Defence R&D Canada—Valcartier.

[2] Lockheed Martin Canada (2007), SATAC Knowledge Representation and Automated Reasoning with JC3IEDM, Task 1—State-of-the-Art in Knowledge Representation, Task 2—Characterization of JC3IEDM and P-JC3IEDM, Contractor Report Defence R&D Canada—Valcartier.

[3] Wagner, G. (2003), Seven Golden Rules for the Web Rule Language, In *IEEE Expert: Intelligent Systems and Their Applications*, Vol. 18-5.

[4] Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., and Tsarkov, D. (2005), OWL Rules: A Proposal and Prototype Implementation, *Journal of Web Semantics*, 3(1).

[5] Bry, F. and Marchiori, M., The Theses on Logic Languages for the Semantic Web (online), W3C, `http://www.w3.org/2004/12/rules-ws/paper/15` (Access Date: July 5, 2007).

[6] Zaniolo, C., Ceri, S., Faloutsos, C., et al. (1997), Advanced Database Systems, Morgan Kaufmann Publishers Inc.

[7] Parsia, B., Sirin, E., and aud others, B. C. G., Cautiously Approaching SWRL (online), `http://www.mindswap.org/papers/CautiousSWRL.pdf` (Access Date: July 18 2007).

[8] Yarger, R. J., Reese, G., and King, T. (1999), MySQL & and mSQL, O'Reilly.

[9] McGuinness, D. L. and van Harmelen, F., OWL Web Ontology Language Overview (online), W3C, `http://www.w3.org/TR/2004/REC-owl-features-20040210/` (Access Date: May 5, 2007).

[10] Smith, M. K., Welty, C., and McGuinness, D. L., OWL Web Ontology Language Guide (online), W3C, `http://www.w3.org/TR/2004/REC-owl-guide-20040210/` (Access Date: May 5, 2007).

[11] Bechhofer, S., van Harmelen, F., Hendler, J., et al., OWL Web Ontology Language Reference (online), W3C, `http://www.w3.org/TR/2004/REC-owl-ref-20040210/` (Access Date: May 5, 2007).

[12] Sowa, J. F. (2007), Fads and Fallacies about Logic, *IEEE Intelligent Systems*, 22(2), 84–87.

[13] Matheus, C. and Ulicny, B. (2006), On the Automatic Generation of an OWL Ontology based on the Joint C3 Information Exchange Data Model. Article provided by the author.

[14] Noy, N. and Rector, A., Defining N-ary Relations on the Semantic Web (online), W3C, http://www.w3.org/TR/swbp-n-aryRelations/ (Access Date: May 20, 2007).

[15] Loaiza, F. L. (2005), A GH-Based Ontology to Support Applications for Automating Decision Support, Technical Report Institute for Defense Analyses.

[16] The Joint C3 Information Exchange Data Model (JC3IEDM Main) 3.0, Multilateral Interoperability Programme, Greding, GE.

Distribution list

DRDC Valcartier CR 2008-255

Internal distribution

1 Director General

2 Document Library

1 Head / C2 Decision Support Systems

1 Head / Intelligence and Information

1 Head / System of Systems

1 Dr. A. Auger

1 M. Bélanger

1 A. Benaskeur

1 J. Berger

1 A. Bergeron-Guyard

1 Dr. A. Boury-Brisset

1 C. Daigle

1 É. Dorion

1 Y. Ferland

1 LCol M. Gareau

1 A. Guitouni

1 H. Irandoust

1 R. Lecocq

1 P. Maupin

1 Dr. L. Pigeon

1 J. Roy

1 A. Sahi

1 G. Thibault

1 Dr. P. Valin

Total internal copies: 25

External distribution

Department of National Defence

1 Director Research and Development Knowledge and Information Management (PDF file)

1 Director Science & Technology Land (DSTL)
Constitution Building, 305 Rideau St., Ottawa, ON, K1N 9E5

1 Director Science & Technology Air (DSTA)
Constitution Building, 305 Rideau St., Ottawa, ON, K1N 9E5

1 Director Science & Technology Maritime (DSTM)
Constitution Building, 305 Rideau St., Ottawa, ON, K1N 9E5

1 Director Science & Technology C4ISR (DSTC4ISR)
Constitution Building, 305 Rideau St., Ottawa, ON, K1N 9E5

1 Director Land Requirements (DLR) 4
Louis St-Laurent Bldg, 555 Boul. de la Carrière, Gatineau, QC, J8Y 6R5

1 Director Land Command Systems Program Management (DLCSPM) 3
Louis St-Laurent Bldg, 555 Boul. de la Carrière, Gatineau, QC, J8Y 6R5

1 Director Land Command Systems Program Management (DLCSPM) 4
Louis St-Laurent Bldg, 555 Boul. de la Carrière, Gatineau, QC, J8Y 6R5

Total external copies: 8

Total copies: 33

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

   Lockheed Martin Canada
   6111 ave. Royalmount, Montréal, Québec, H4P 1K6

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

   UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

   SATAC Knowledge Representation and Automated Reasoning with JC3IEDM

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

   Demers, H.; Duquet, J.-R.

5. DATE OF PUBLICATION (Month and year of publication of document.)

   September 2008

6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)

   62

6b. NO. OF REFS (Total cited in document.)

   16

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

   Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

   Defence R&D Canada – Valcartier
   2459 Pie-XI Blvd. North Val-Bélair QC G3J 1X5, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

   12of

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

   W7701-061941/001/QCL

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

   DRDC Valcartier CR 2008-255

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

   (X) Unlimited distribution
   ( ) Defence departments and defence contractors; further distribution only as approved
   ( ) Defence departments and Canadian defence contractors; further distribution only as approved
   ( ) Government departments and agencies; further distribution only as approved
   ( ) Defence departments; further distribution only as approved
   ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

   Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report describes the knowledge representation languages that are used by the *Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM)* and the *Protégé-JC3IEDM (P-JC3IEDM)*. It then describes the semantic expressiveness of JC3IEDM and P-JC3IEDM and how these can be used to support automated reasoning with respect to reasoning examples that were developed in Task 3 of this contract. [1]

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE R&D DÉFENSE

www.drdc-rddc.gc.ca