NISTIR 7535

Reasoning in Manufacturing Part-Part Examples with OWL 2

Nenad Krdzavac
Conrad Bock

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 7535

Reasoning in Manufacturing Part-Part Examples with OWL 2

Nenad Krdzavac
Conrad Bock
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

October 2008

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Deputy Director

Reasoning in Manufacturing Part-Part Examples with OWL 2

Nenad Krdžavac, Conrad Bock

Manufacturing Systems and Integration Division,
National Institute of Standards and Technology
100 Bureau Drive, Stop 8263, Gaithersburg, MD 20899-8263

This report examines whether the Web Ontology Language 2 (OWL 2) is expressive enough for some common manufacturing examples involving relations between parts of composite or assembled objects (part-part relations). We specify the semantics of these examples in first-order logic, express them in complex role inclusions, and compare these to OWL 2 role inclusions. We give tests based on the first order semantics and determine whether OWL 2 supports the semantics. We also give restrictions typical of part-part applications in manufacturing that might be useful in future research.

1 Introduction

The basic notions in description logics (DLs) are *concepts* (unary predicates), *roles* (binary predicates) [1], and *individuals* to which concepts and roles might apply. For example, the concept Car might apply to an individual vehicle owned by a particular person, and the role engineInCar might link that individual vehicle to a particular engine with a certain serial number in the car. Individuals *satisfy* concepts that apply to them. Reasoning services in DLs attempt to prove whether it is possible for particular concepts to apply to any individuals at all (*satisfiability*). The SROIQ DL[1] is the basis for Web Ontology Language 2 (OWL 2) [2][3].

This paper examines whether OWL 2 can express some common manufacturing examples involving relations between parts of the same composite or assembly (*part-part relations*) [4]. The examples are subject to restrictions common in manufacturing problems (*m-restrictions*), for example the relations between parts differ from the relations between part and whole. The most applicable portion of OWL 2 for part-part relations are complex role inclusion axioms (RIAs). Simple role inclusions are generalizations between two roles, for example the sisterOf role is included in the siblingOf role, meaning that one's sister is also one's sibling. The notation used in this paper is sisterOf $\subseteq$ siblingOf, where sisterOf is the specialized role and siblingOf is the generalized role. Complex role inclusions use role composition to construct the specialized role. For example, sisterOf composed with parentOf (notated as sisterOf o parentOf) produces a new role linking individual people with their sister's parents. The new composed role can be included in others. For example, sisterOf composed with parentOf is included in parentOf (sisterOf o parentOf $\subseteq$ siblingOf), which means your sister's parent is your parent also.

The primary issue in applying OWL 2 RIAs to part-part relations is that OWL 2 is undecidable for most cyclic role inclusion axioms, while common manufacturing problems involving part-part relations translate to RIAs (*m-RIAs*) that are cyclic. These have the same role on both sides of an inclusion, or a chain of inclusions. To achieve decidability, OWL 2 requires acyclic dependencies between roles in these RIAs in many cases. To regain decidability of OWL 2 extended with m-RIAs, it is important to find restrictions on the roles (*m-restrictions*).

[1] DLs are named with letters corresponding to the particular fragment of first order logic [4] captured in the DL.

This report is organized as follows. The next section explains the first order logic (FOL) [4] formalization of the examples that is used to facilitate comparison with OWL 2, which is a fragment of FOL. In section 3, we represent some common manufacturing part-part examples as RIAs. In section 4 we compare m-RIAs with RIAs defined in OWL 2, in particular, we discuss whether some m-RIAs cannot be expressed as OWL 2 RIAs. We also address support for acyclic restrictions in some part-part relations. Section 5 examines the restrictions on roles in example 1 and OWL 2. It also gives a definition for general role inclusion, supporting part-part relations. In section 6, we analyze consistency of one of the examples in detail. We explain the behavior of standard DLs reasoners (PELLET [5] and FACT++ [6]) in checking consistency. Finally we give some ideas for future research.

2 First order reasoning in part-part examples

For each role representing a part-part relation, two sets limit the things linked by the role [4]:

1. The set of things that must play the part-part role for a particular individual (*minimum value set*). For example, for the role of powering something, the minimum value set for an engine in a car is the wheels of the same car.

2. The set of things that must not play the part-part role for a particular individual (complement of *maximum value set*). For example, for the role of providing power to something, the maximum value set for an engine in a car might include the wheels in the same car, or other things in the car, but not things in other cars.

The two subsections below apply minimum and maximum value sets to the first part-part example of the paper. The example is the one given above, requiring engines in cars to power wheels in the same car (minimum value set), and not wheels in other cars (maximum value set), see illustration in Figure 1 in the Unified Modeling Language class diagram notation [8]. Concepts are notated as labeled rectangles, while roles are notated as labeled, directed links. In this example, **powers** is a part-part relation, while **engineInCar** and **wheelInCar** are part-whole relations (the **engineInCar^{-1}** role is used in Section 3).

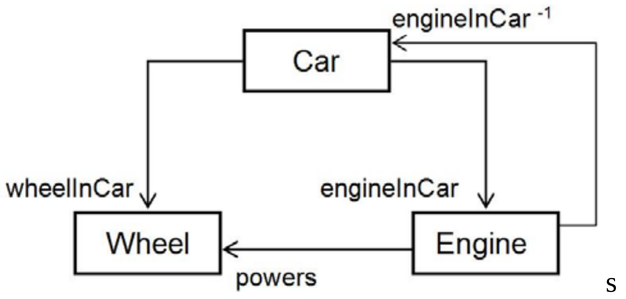


Figure 1: Part–part example 1

The example is expressed first in FOL [4] to facilitate comparison with OWL 2, which is a fragment of FOL. One important aspect of FOL is the semantics of implication, as shown in Expression 1, using Common Logic Interchange Format (CLIF) [10], which uses "if" for implication. It can be illustrated with the truth table in Table 1. Column A gives the truth value for proposition A in Expression 1 (the premise), column B the truth value for proposition B (the conclusion), and the rightmost column the truth value for implication as a whole. The expression is false only if A is true and B false, otherwise the expression is true. In particular, if A is false, the expression is true regardless of the value of B, and the same if B is true.

(if A B)

Expression 1: Logical Implication in CLIF notation

Case	A	B	(if A B)
1	T	F	F
2	F	T	T
3	T	T	T
4	F	F	T

Table 1. Truth table for implication

2.1 Minimum value set for example 1

This subsection explains minimum value sets for the first part-part example of the paper. The FOL for this is shown in Expression 2 [4], and illustrated in Figure 1. Expression 2 has the part-part relation **powers** in the conclusion of the implication, which is characteristic of minimum value sets, giving sufficient conditions for individuals linked by **powers**. The expression says if a car has an engine (via role **engineInCar**) and a wheel (via role **wheelInCar**) (column A of the implication, see Table 1) then the engine **powers** the wheel.

```
(forall (?car ?engine ?wheel)
   (if (and (engineInCar ?car ?engine)
            (wheelInCar ?car ?wheel))
       (powers ?engine ?wheel)))
```

Expression 2: FOL for minimum value set example 1

Each row of Table 1 gives a consistency case for a knowledge base (KB) according to the FOL expression:

1. An individual car has an engine and wheels, via **engineInCar** and **wheelInCar** respectively (column A is true), and the engine does not power the wheels, via **power** (column B is false), then the implication is false, and the KB is inconsistent.
2. An individual engine is not in a car (via **engineInCar)**, or the car has no wheels (via **wheelInCar**), or both (column A is false), and the engine happens to power the wheels anyway (column B is true), via **powers**, then the implication is true and the KB is consistent.
3. An individual car has an engine and wheels, via **engineInCar** and **wheelInCar** respectively (column A is true), and the engine powers the wheels, via **powers** (column B is true), then the implication is true, and the KB is consistent.
4. An individual engine is not in a car (via **engineInCar$^-$)**, or individual wheels are not in a car (via **wheelInCar**), or both (column A is false), and the engine does not power the wheels, via **powers** (column B is false), then the implication is true and the KB is consistent.

The KB of case 1 is the only inconsistent one, and can be "repaired" by modifying it to fall under one of the other cases. The engine or wheel, or both, can be removed from the car, giving case 4. The engine can power the wheels, giving case 3. Finally, the engine can power wheels with the engine or wheels removed from the car, or both, giving case 2. Case 3 is a typically chosen repair, but all the repairs have the same logical effect of making the KB consistent.

2.2 Maximum value set for example 1

This subsection explains maximum value sets for the first part-part example of the paper. The FOL for this is shown in Expression 3, and illustrated in Figure 1. Expression 3 has the part-part relation **powers** in the premise of the implication, which is characteristic of maximum value sets, giving necessary conditions for individuals linked by **powers**. It says if a car has an engine (via role **engineInCar**) and the engine powers something (column A of the implication, see Table 1) then the car has the powered thing as a wheel (via role **wheelInCar**). It does not require engines in cars to power wheels in the cars, as Expression 2 does for minimum value sets, which is a sufficient condition for individuals to be linked by **powers**.

```
(forall (?car ?engine ?poweredThing)
   (if (and (engineInCar ?car ?engine)
            (powers ?engine ?poweredThing))
       (wheelInCar ?car ?poweredThing)))
```

Expression 3: FOL for maximum value set of example 1

Each row of Table 1 gives a consistency case for a knowledge base (KB) based on the FOL expression:

1. An individual car has an engine that powers something, via **engineInCar** and **powers** respectively (column A is true), and the powered things are not wheels in the car, via **wheelInCar** (column B is false), then the implication is false, and the KB is inconsistent.
2. An individual engine is not in a car (via **engineInCar)**, or it does not power anything (via **powers**), or both (column A is false), and the car happens to have wheels (column B is true), via **wheelInCar**, then the implication is true and the KB is consistent.
3. An individual car has an engine that powers something, via **engineInCar** and **powers** respectively (column A is true), and the powered things are wheels in the car, via **wheelInCar** (column B is true), then the implication is true, and the KB is consistent.
4. An individual engine is not in a car (via **engineInCar)**, or it does not power anything (via **powers**), or both (column A is false), and the car has no wheels, via **wheelInCar** (column B is false), then the implication is true and the KB is consistent.

The KB of case 1 is the only inconsistent one, and can be “repaired” by modifying it to fall under one of the other cases. The engine can be removed from the car, or not power anything, giving case 4. If the engine is powering wheels, the wheels can be put in the car, giving case 3. Finally, The engine can be removed from the car, or not power anything, or both, giving case 2. Case 4 is a typically chosen repair, but all the repairs have the same logical effect of making the KB consistent.

3 Part-part examples in complex role inclusions

This section represents some common manufacturing part-part examples as RIAs (m-RIAs).

3.1 Example 1

See Section 2 for the description and FOL of the first example. RIA-1.1 below is a complex role inclusion for the minimum value set (Expression 2).[2] Its uses the inverse relation of **engineInCar** (**engineInCar^{-1}**), see Figure 1. The composition ("o" operator) proceeds from an individual engine on the left, to the car that has the engine (via **engineInCar^{-1}**), then through the composition from that car to an individual wheel in the car (via **wheelInCar**). The complex role formed by composition links individual engines to wheels in the same car as the engine. The inclusion ($\subseteq$ operator) means every two individuals linked by the complex role on the left are also linked by the **powers** role on the right. Altogether, RIA-1.1 requires every engine in a car with wheels to power the wheels. It is equivalent to the FOL in Expression 2.

RIA-1.1: engineInCar^{-1} o wheelInCar $\subseteq$ powers

RIA-1.2 below is a complex role inclusion for the maximum value set (Expression 3). The composition proceeds from an individual car on the left, to an engine in that car (via **engineInCar**), then through the composition to whatever that engine powers (via **powers**). The complex role formed by composition links individual cars to things powered by their engines. The inclusion means every two individuals linked by the complex role on the left are also linked by the **wheelInCar** role on the right. Altogether RIA-1.2 restricts whatever is powered by an engine in a car to be a wheel in the same car. It does not require an engines in a car to power wheels in the car, as RIA-1.1 does. It is equivalent to the FOL in Expression 3.

RIA1.2: engineInCar o powers $\subseteq$ wheelInCar

3.2 Example 2

The second example modifies the first with individuals that play parts in more than one whole. This applies to models of organizations, as in supply chains. The example requires teachers to teach students in the same school as the teacher (minimum value set), and not students in other schools (maximum value set), but does not prevent teachers from teaching in more than one school, see illustration in Figure 2. In this example, **teaches** is a part-part relation, while **teacherAtSchool** and **studentAtSchool** are part-whole relations.

[2] This was suggested by Evren Sirin, see http://lists.w3.org/Archives/Public/public-owl-dev/2007JanMar/0245.html.

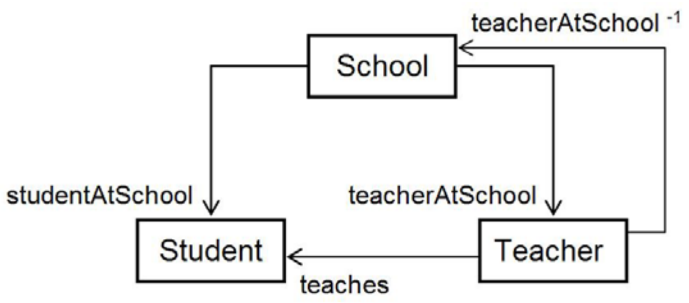


Figure 2: Part –part example 2

FOL for the minimum value set of this example is shown in Expression 4. It has the same form as Expression 2, requiring all teachers at a school to teach all students at that school (the example is for a very small school). The equivalent complex role inclusion RIA-2.1 below, has the same form as RIA-1.1.

```
(forall (?school ?teacher ?student)
   (if (and (teacherAtSchool ?school ?teacher)
            (studentAtSchool ?school ?student))
       (teaches ?teacher ?student)))
```

Expression 4: FOL for minimum value set of example 2

RIA-2.1: teacherAtSchool^{-1} o studentAtSchool $\subseteq$ teaches

The FOL for the maximum value set of the second example is shown in Expression 4. It restricts teaching to only happen between teachers and students who are at the same school. The equivalent role inclusion RIA-2.2 below has a complex role on the right side, using the inverse role **teacherAtSchool^{-1}**. It proceeds from an individual teacher to a school they teach at, then through composition to students at that school. The inclusion means every two individuals linked by the **teaches** role on the left are also linked by the complex role on the right. Altogether RIA-2.2 restricts teaching to teachers and students of the same school. It does not require teachers to teach students at the same school, as RIA-2.1 does. It is equivalent to the FOL in Expression 4.

```
(forall (?school ?teacher ?personTaught)
   (if (teaches ?teacher ?personTaught)
       (exists (?someSchool)
          (and (teacherAtSchool ?someSchool ?teacher)
               (studentAtSchool ?someSchool ?personTaught)))))
```

Expression 5: FOL for maximum value set of example 2

RIA-2.2: teaches $\subseteq$ teacherAtSchool^{-1} o studentAtSchool

RIA-2.2 is a more general complex role inclusion than RIA-1.2, because it has more than one role on the right.

3.3 Example 3

The third example modifies the first with individuals involved multiple times in the same part-part relation. The example requires engines in cars to power multiple things in the same car (minimum value set), and not those same kinds of things in other cars (maximum value set), see illustration in Figure 3. In this example, **powers** is a part-part relation, while **engineInCar**, **wheelIncar**, **generatorInCar**, and **oilPumpInCar** are part-whole relations.

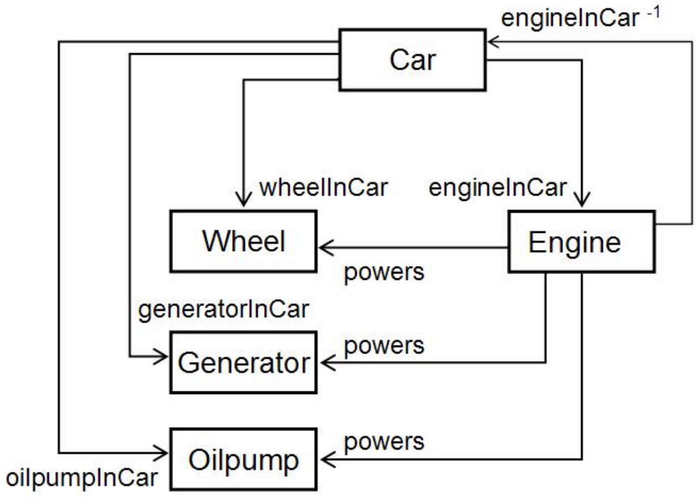


Figure 3: Part–part example 3

FOL for the minimum value set of this example is shown in Expression 6. It has multiple subexpressions, repeating the form of Expression 2 and Expression 4, requiring the engine in every car to power the wheels, generator, and oil pump in the same car. The equivalent complex role inclusions RIA-3.1, RIA-3.2, and RIA-3.3 below repeat the form of RIA-1.1 for all the things powered in the car. It is characteristic of minimum value sets that they can be combined without affecting each other (they are "open world", as is typical of sufficient conditions).

```
(forall (?car ?engine ?wheel)
   (if (and (engineInCar ?car ?engine)
            (wheelInCar ?car ?wheel))
       (powers ?engine ?wheel)))

(forall (?car ?engine ?generator)
   (if (and (engineInCar ?car ?engine)
            (generatorInCar ?car ?generator))
       (powers ?engine ?generator)))

(forall (?car ?engine ?oilpump)
   (if (and (engineInCar ?car ?engine)
            (oilpumpInCar ?car ?oilpump))
       (powers ?engine ?oilpump)))
```

Expression 6: FOL for minimum value set of example 3

RIA-3.1: engineInCar^{-1} o wheelInCar $\subseteq$ powers
RIA-3.2: engineInCar^{-1} o generatorInCar $\subseteq$ powers
RIA-3.3: engineInCar^{-1} o oilpumpInCar $\subseteq$ powers

The FOL for the maximum value set of the third example is shown in Expression 7. It restricts whatever is powered by engines in cars to be either the wheels, generator, or oil pump of the same car, or all of these. The equivalent role inclusion RIA-3.2 below has a union of roles on the right side. The union forms a role linking cars to their wheels, generators, and oil pumps. It is characteristic of maximum value sets that they affect each other when combined, and cannot be written simply by conjoining them, as minimum value sets can (they “close” the open world minimum value sets, as is typical of necessary conditions). This union role includes the complex role on the left. Altogether RIA-3.2 restricts engines in cars to power the wheels, generator, oil pump, or all of these, in the same car. It does not require engines to power those things, as RIA-3.1 does. It is equivalent to the FOL in Expression 7.

```
(forall (?car ?engine ?poweredThing)
   (if (and (engineInCar ?car ?engine)
            (powers ?engine ?poweredThing))
       (or (wheelInCar ?car ?poweredThing)
           (generatorInCar ?car ?poweredThing)
           (oilpumpInCar ?car ?poweredThing))))
```

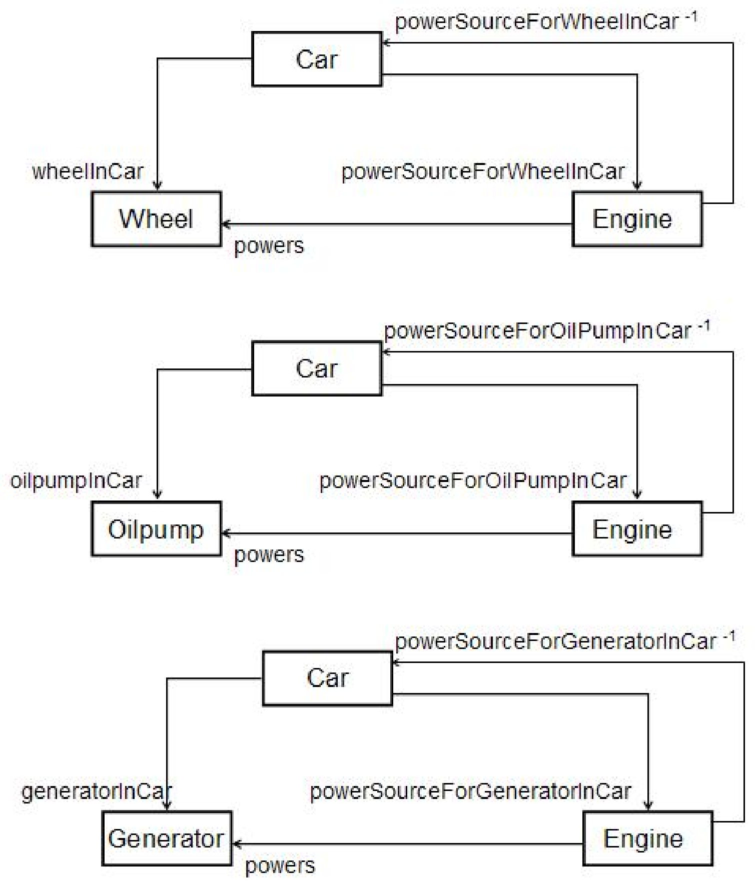
Expression 7: FOL for maximum value set of example 3

RIA-3.4: engineInCar o powers $\subseteq$ wheelInCar $\cup$ generatorInCar $\cup$ oilpumpInCar

RIA-3.4 is a more general complex role inclusion than RIA-2.2, because it has more than one role on the right and uses other operators besides composition.

3.4 Example 4

The fourth example modifies the third with the same individual identified by the different part-whole relations. The example has multiple relations identifying the power sources for things in the car (minimum value set), and not the power sources for same kinds of things in other cars (maximum value set), where the power source identified is always the engine, see illustrations in Figure 4. In this example, **powers** is a part-part relation, while **powerSourceForWheelInCar**, **powerSourceForGeneratorInCar**, and **powerSourceForOilPumpInCar** are part-whole relations.



powerSourceForWheelInCar -1
Car
wheelInCar
powerSourceForWheelInCar
Wheel
Engine
powers
powerSourceForOilPumpInCar -1
Car
oilpumpInCar
powerSourceForOilPumpInCar
Oilpump
Engine
powers
powerSourceForGeneratorInCar -1
Car
generatorInCar
powerSourceForGeneratorInCar
Generator
Engine
powers

Figure 4: Part –part example 4

FOL for the minimum value set of this example is shown in Expression 8. It has multiple subexpressions, repeating the form of Expression 2 and Expression 4, requiring the power sources to power their corresponding parts of the car. The equivalent complex role inclusions RIA-4.1, RIA-4.2, and RIA-4.3 below repeat the form of RIA-1.1 for all the things powered in the car.

```
(forall (?car ?engine ?wheel)
   (if (and (Car ?car)
            (powerSourceForWheelInCar ?engine)
            (wheelInCar ?car ?wheel))
       (powers ?engine ?wheel)))

(forall (?car ?engine ?generator)
   (if (and (Car ?car)
            (powerSourceForGeneratorInCar ?car ?engine)
            (generatorInCar ?car ?generator))
       (powers ?engine ?generator)))

(forall (?car ?engine ?oilpump)
   (if (and (Car ?car)
            (powerSourceForOilPumpInCar ?car ?engine)
            (oilpumpInCar ?car ?oilpump))
       (powers ?engine ?oilpump)))
```

Expression 8: FOL for minimum value set of example 4

RIA-4.1: powerSourceForWheelInCar^{-1} o wheelInCar $\subseteq$ powers
RIA-4.2: powerSourceForOilPumpInCar^{-1} o oilpumpInCar $\subseteq$ powers
RIA-4.3: powerSourceForGeneratorInCar^{-1} o generatorInCar $\subseteq$ powers

The FOL for the maximum value set of the fourth example is shown in Expression 9. It restricts whatever is powered by the engine in a car identified by multiple power source relations to be either the wheels, generator, or oil pump of the same car, or all of these. The equivalent role inclusion RIA-4.2 below has an intersection of roles on the right side. The intersection forms a role linking cars to their engine playing multiple roles. This intersection role is included in the union on the left. Altogether RIA-4.2 restricts the engine playing to power source roles in cars to power wheels, generator, oil pump, or all of these, in the same car. It does not require engines to power those things, as RIA-4.1 does. It is equivalent to the FOL in Expression 9.

```
(forall (?car ?engine ?poweredThing)
     (if (and (and (powerSourceForWheelInCar ?car ?engine)
                   (powerSourceForGeneratorInCar ?car ?engine)
                   (powerSourceForOilPumpInCar ?car ?engine))
              (powers ?engine ?poweredThing))
         (or (poweredWheelInCar ?car ?poweredThing)
             (generatorInCar ?car ?poweredThing)
             (oilpumpInCar ?car ?poweredThing))))
```

Expression 9: FOL for maximum value set of example 4

RIA-4.4: (powerSourceForWheelInCar $\cap$ powerSourceForGenratorInCar
$\cap$ powerSourceForOilPumpInCar) o powers $\subseteq$ wheelInCar $\cup$ generatorInCar
$\cup$ oilpumpInCar

RIA-4.4 is a more general complex role inclusion than RIA-3.4, because it has more than one role on the right, and uses other operators besides composition on both sides.

3.5 Example 5

The fifth example modifies the first with relations between parts of parts. The example requires crankshafts in engines in cars to power hubs in wheels in the same car (minimum value set), and not hubs in wheels in other cars (maximum value set), see illustration in Figure 5. In this example, **powers** is a part-part relation, while **engineInCar**, **crankshaftInEngine**, **wheelInCar**, and **hubInWheel** are part-whole relations. **Car** is the whole for **engineInCar** and **wheelInCar**, while **Engine** is the whole for **crankshaftInEngine**, and **Wheel** the whole for **hubInWheel**.

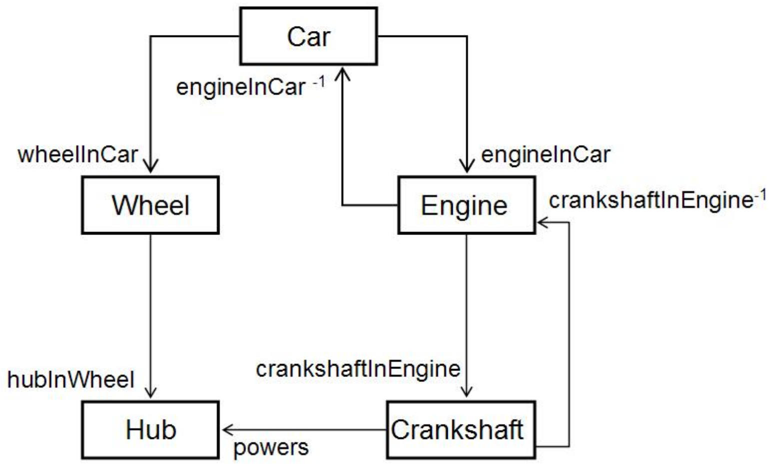


Figure 5: Part–part example 5

FOL for the minimum value set of this example is shown in Expression 10. It has a similar form to Expression 2, requiring the crankshaft in the engine of a car to power the hubs in the wheels of the same car (the **crankshaftPowers** relation is a special case of **powers**). The equivalent complex role inclusion RIA-5.1 below, has a similar form to RIA-1.1. The complex role on the left proceeds from an individual crankshaft to the engine that has it, through a composition to the car that has the engine, then through another two compositions to the wheel in the same car, and finally to the hub in the wheel. The complex role is included in the role on the right for crankshaft power. Altogether RIA-5.1 requires every crankshaft in an engine in a car to power the hub in the wheels of the same car.

```
(forall (?car ?engine ?wheel ?crankshaft ?hub)
   (if (and (engineInCar ?car ?engine)
            (crankshaftInEngine ?engine ?crankshaft)
            (wheelInCar ?car ?wheel)
            (hubInWheel ?wheel ?hub))
       (crankshaftPowers ?crankshaft ?hub)))
```

Expression 10: FOL for minimum value set of example 5

RIA-5.1: engineInCar^{-1} o crankshaftInEngine^{-1} o wheelInCar o hubInWheel $\subseteq$ crankshaftPower

The FOL for the maximum value set of the fifth example is shown in Expression 11. It restricts whatever is powered by a crankshaft in an engine in a car to be a hub in a wheel in the same car. The equivalent role inclusion RIA-5.2 below, has a similar form to RIA-1.2. The complex role on the left proceeds from an individual car to its engine, then through two compositions to the crankshaft in the engine and the hub powered by the crankshaft. The complex role on the left is included in the complex role on the right, which proceeds from an individual car to this wheel, then through composition to the hubs in the wheels. Altogether RIA-5.2 restricts the hubs powered by the crankshaft in the engine of a car to be in the wheels of the same car. It does not require the crankshaft to power the hubs as Expression 10 does.

```
(forall (?car ?engine ?crankshaft ?poweredThing)
   (if (and (engineInCar ?car ?engine)
            (crankshaftInEngine ?engine ?crankshaft)
            (crankshaftPowers ?crankshaft ?poweredThing))
       (exists (?wheel)
          (and (poweredWheelInCar ?car ?wheel)
             (hubInWheel ?wheel ?poweredThing)))))
```

Expression 11: FOL for maximum value set of example 5

RIA-5.2: engineInCar o crankshaftInengine o crankshaftPower $\sqsubseteq$ wheelInCar o hubInWheel

RIA-5.2 is a more general complex role inclusion than RIA-1.2, because it has more than one role on the right side, and more general than RIA-2.2, because it has more than one role on the left.

4 Part-part examples in OWL 2 complex role inclusions

This section compares the m-RIAs of Section 3 with RIAs in OWL 2. The m-RIAs for each example are examined individually and collectively. Individually, OWL 2 RIAs, must have exactly one role on the right side, and only composition of roles on the left. Table 2 summarizes whether the individual m-RIAs satisfy this condition. RIA-2.2 does not satisfy the condition, because it has two roles composed on the right side. RIA-3.4 and RIA-4.4 do not, because they have a union of roles on the right side, and RIA-4.4 has an intersection of roles on the left. RIA-5.2 does not satisfy the condition, because it has a composition of roles on the right side.

mRIA / Subexpression	.1	.2	.3	.4
RIA-1	Yes	Yes		
RIA-2	Yes	No		
RIA-3	Yes	Yes	Yes	No
RIA-4	Yes	Yes	Yes	No
RIA-5	Yes	No		

Table 2: Representing example part-part relations with role inclusions in OWL 2

Collectively, OWL 2 RIAs must be acyclic in all but a few cases. This means the same role cannot appear on both the right and left sides of a set of RIAs. The only exceptions are if the role on the right is at the beginning or end of the composition on the left, but not both. None of the examples in Section 3 can be represented as acyclic RIAs. For example, in RIA-1.1, the **powers** role appears on the right side, while in RIA-1.2, it appears on the left, reflecting the sufficient and necessary (open and closed) nature of these RIAs, respectively. It is essential for reasoning to close open world statements, in this case, to prevent engines from powering wheels in other cars at the same time they are required to power the engines in their own.

5 Restrictions on roles in part-part example in OWL 2

This section identifies restrictions on m-RIAs (*m-restrictions*) in examples 1 and 5 of Section 3, which have subexpressions that can be individually represented in OWL 2. The restrictions in this section are typical of physical assembly. They are important for further research in part-part reasoning, because RIAs without restrictions on roles are known to be undecidable [2].

The form of m-RIAs in examples 1 and 5 of Section 3 are:

1. R_1^{-1} o $R_2 \subseteq S$
2. R_1 o $S \subseteq R_2$

where R_i (i=1,2) are potentially compositions of other roles. R_i are the part-whole relations and S is the part-part relation.

Restrictions on m-RIAs (m-restrictions) are:[3]

1. $\forall i\ R_i^{-1}$ is a partial function (is not required to link all elements in its domain). An object cannot be part of more than one whole at a time, but might not be in any whole at all.

2. $\forall i\ (p, w) \in R_i^{-1\,I} \Rightarrow \forall j\ _{j \neq i}\ (p,w) \notin R_j^{-1}$; whole cannot link to a part by more than one role.

3. $\forall i,j\ _{i \neq j}\quad C_i^{\ I} \cap C_j^{\ I} = \emptyset$; The concepts used in part-part examples are disjoint, see restrictions below.

4. $\forall i,j\ _{i \neq j}\ R_i^{\ I} \cap R_j^{\ I} = \emptyset$, $R_i^{\ I} \cap S^{\ I} = \emptyset$, $S^{\ I} \cap R_j^{\ I} = \emptyset$; The part-whole relations are disjoint with each other and with the part-part relation. Each part is identified by only one role, which is not the one used to connect parts to each other.

5. $\forall i\ \ \text{domain}(R_i)^{\ I} \cap \text{range}(R_i)^{\ I} = \emptyset$; The domains and ranges of the part-whole relations are disjoint.

6. $\forall i\ R_i^{\ I} \cap R_i^{-1\,I} = \emptyset$; Parts do not contain their wholes. This is a consequence of restriction 5.

These restrictions are supported in OWL 2, except for second one, see Table 3.

m-restriction	OWL 2
1	YES
2	NO
3	YES
4	YES
5	YES

Table 3. Supporting the restrictions in OWL 2

6 Checking consistency using RIAs in example 1

In this section, we test consistency of knowledge bases (KBs) containing RIAs in example 1 on the well-known description logic reasoners PELLET [5] and FACT++ [6], with and without the m-restrictions of Section 5. The tests consist of assertions about individuals (*ABox* in DL terminology), and the m-RIAs (*RBox* in DL terminology). The first four tests

[3] The *I* superscript in these restrictions means the set of individuals of the concept or links of the role to which it is applied (the interpretation of the concept or role). For example Car^{I} means the set of all individual cars, while engineInCar^{I} is the set of all links between individual cars and engines.

check minimum value sets in example 1, see Figure 7, while the last four check maximum value sets, see Figure 8.[4] The first four tests are:

- The first test is not consistent with the minimum value set of **powers**, because it asserts an engine is part of a car that also has a wheel, but the engine does not power the wheel. This test corresponds to case 1 for implication, see Table 1 in Section 2.
- The second test is consistent with the minimum value set of **powers**, because it asserts an engine powers a wheel when neither are in a car. The test corresponds to case 2 in Table 1.
- The third test is similar to the first, but is consistent with the minimum value set of **powers**, even though it is not asserted that the engine powers the wheels. It might be that the engine powers the wheels, but it happens to not be recorded in the KB (OWL 2 is open world). The KB is consistent, but incomplete, because it does not record whether the engine powers the wheels or not. This test corresponds to case 3 in Table 1, because open world allows the possibility that column B is true.
- The fourth reasoning task is consistent with the minimum value set of **powers**, because an engine is not powering a wheel and neither are in a car. The test corresponds to case 4 in Table 1.

Test 1 Abox: (engine1, car1) : engineInCar $^{-1}$ wheelInCar(car1, wheel1) (engine1, wheel1) : ¬ powers car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar	Test 2 Abox: (engine1, car1): ¬engineInCar $^{-1}$ (car1, wheel1): ¬wheelInCar powers (engine1, wheel1) car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar
Test 3 Abox: (engine1, car1) : engineInCar $^{-1}$ wheelInCar(car1, wheel1) car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar	Test 4 Abox: (engine1, car1) : ¬ engineInCar $^{-1}$ (car1, wheel1): ¬ wheelInCar (engine1, wheel1): ¬ powers car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar

Figure 7. Four example KBs for minimum value sets in example 1

The second four tests check maximum value sets for example 1, see Figure 8:

- The fifth test is inconsistent with the maximum value set of **powers**, because an engine in a car powers something that is not a wheel in the car. This test corresponds to case 1 for implication Table 1 in Section 2.

[4] We performed the above tests with these examples in Protégé 4.0 - beta [11], with PELLET 1.5 and FACT ++ plug-ins. The reasoners returned exceptions in all tests, due to cycles in the RIAs.

- The sixth test is consistent with the maximum value set of **powers,** because a car has a wheel, but not the engine, and the engine does not power the wheel. This test corresponds to case 2 in Table 1.
- The seventh test is consistent with the maximum value set of **powers**, because an engine powers a wheel in the same car. This test corresponds to case 3 in Table 1.
- The eighth test is consistent with the maximum value set of **powers**, because neither the engine nor wheel are in the car, and the engine does not power the wheel.

Test 5 Abox: engineInCar(car1, engine1) powers(engine1, wheel1) (car1, wheel1) : ¬wheelInCar car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar	Test 6 Abox: (car1, engine1): ¬engineInCar (engine1, wheel1) : ¬powers wheelInCar (car1, wheel1) car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar
Test 7 Abox: engineInCar (car1, engine1) powers(engine1, wheel1) wheelInCar(car1, wheel1) car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar	Test 8 Abox: (car1, engine1) : ¬ engineInCar (engine1, wheel1): ¬ powers (car1, wheel1): ¬ wheelInCar car1: CAR wheel1: WHEEL engine1: ENGINE Rbox: engineInCar $^{-1}$o wheelInCar => powers engineInCar o powers => wheelInCar

Figure 8. Four example KBs for maximum value sets in example 1

7 Conclusion and Future work

In this report we show how OWL 2 does or does not address some common manufacturing part-part examples. We formalize the examples in FOL, and translate to complex role inclusion axioms. We found some of the most common examples can be defined in OWL 2 RIAs. Finally, we give tests for these RIAs based on the first order semantics. The RIAs contain cyclic dependencies, which are not supported in OWL 2 reasoning.

Our future work will be focused on extending OWL 2 to support part-part relations, in particular, cycles in complex role inclusions common in manufacturing part-part examples, and using typical restrictions from the manufacturing domain given in this paper. In future research we will try to prove decidability of such logic and we will try to propose a tableau-based algorithm for the logic. We plan also to implement a DL reasoner based on this algorithm and tests this software in several practical applications.

8 Disclaimer

Commercial equipment and materials might be identified to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the U.S. National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

9 References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., The Description Logic Handbook - Theory, Implementation and Application, Cambridge University Press, 2003.

[2] Horrocks, I., Kutz, O., and Sattler, U., "The Even More Irresistible SROIQ," in Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57-67, http://www.comlab.ox.ac.uk/people/ian.horrocks/Publications/download/2006/HoKS06a.pdf, American Association of Artificial Intelligence Press, 2006.

[3] W3C, OWL Working Group, http://www.w3.org/2007/OWL/wiki/OWL_Working_Group, 2008.

[4] Enderton, H., A Mathematical Introduction to Logic, Academic Press, 1972.

[5] Sirin, E., Parsia, B., "An OWL DL Reasoner," in Proceedings of the International Workshop on Description Logics (DL2004), British Columbia, Canada, June 2004.

[6] Tsarkov, D., and Horrocks, I., "FaCT++ Description Logic Reasoner: System Description," in Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006), Lecture Notes in Artificial Intelligence, 4130, pp. 292-297, Springer, 2006.

[7] Bock, C., "Part-part Relations in an RDF/S and OWL Extension," U.S. National Institute of Standards and Technology, Interagency Report 7507, 2008.

[8] Object Management Group, "OMG Unified Modeling Language, Superstructure," http://doc.omg.org/formal/07-11-02, November 2007.

[9] Horrocks, I., Sattler, U., "Decidability of SHIQ with complex role inclusion axioms," Journal of Artificial Intelligence, 160:1-2, pp. 79-104, December 2004.

[10] International Standards Organization, "Common Logic (CL) - A framework for a family of logic-based languages," ISO 24707, http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip, October 2007.

[11] Stanford Center for Biomedical Informatics Research, "The Protégé Ontology Editor and Knowledge Acquisition System", http://protege.stanford.edu, 2008.