A Semantic Web Representation of a Product Range Specification based on Constraint Satisfaction Problem in the Automotive Industry

Fadi Badra[1], François-Paul Servant[2], and Alexandre Passant[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway, Ireland
firstname.lastname@deri.org
[2] Renault SA
13 avenue Paul Langevin, 92359 Plessis Robinson, France
francois-paul.servant@renault.com

Abstract. Product Range Specification (PRS) in the automotive world is one of the most complex PRS that exists in industrial contexts. PRS plays therefore a key role in the information system of an automaker: related data pervades many systems, and numerous applications are using it. This is the case at Renault, where PRS is modelled as a Constraint Satisfaction Problem. In this paper, we study how to represent the objects, concepts and services related to such a PRS using Semantic Web standards. Plugging them into a Linked Data based architecture enables with new ways to access corresponding data and tools in the whole car manufacturing and selling process.

Keywords: Constraint Satisfaction Problem, Product Range Specification, Enterprise Linked Data, RDF(S), Configuration

1 Introduction and Motivation

In order to cut the cost of accessing data and exchanging it between systems — both internally and externally — Renault, one of the world's largest automakers[3], is considering the use of Linked Data principles and Semantic Web technologies in its information system. Several prototypes were done in the past years [12], and the first operational application based on Linked Data at Renault was released in early 2010, enabling the vision of "Linking Enterprise Data" [16] in the company. To go one step further, Renault aims at building a Semantic Web compliant representation of the objects, concepts and services related to its Product Range Specification (PRS). PRS is used to specify the set of all possible car configurations that an automaker can sell. Several reasons motivate the representation of PRS into Renault's current Linked Data infrastructure. First, PRS impacts many business tasks, and is a core component of Renault's information system. Product diversity being huge and complex, dedicated tools and

[3] http://www.renault.com

services are required to handle it. Moreover, PRS related data pervades many systems, and must be used in many applications which, for instance, must refer to subsets of the range (such as the "Twingo petrol with air conditioning"), and must state things about them (starting price, features, available options, etc.). Hence, enabling easier access to PRS data using the Linked Data principles, and making the PRS functionalities available as REpresentational State Transfer (REST) services, would be very rewarding. In addition, representing PRS in Semantic Web format would provide efficient ways to share data with industrial partners.

On the other hand, PRS data is not plain vanilla relational data. That could be a pain point within a Linked Data framework. In the evaluation of Linked Data for Renault, we need therefore to carefully check how this difficulty can be overcome. If the deployment of Semantic Web technologies is to be expanded at Renault, we need to represent and manipulate the objects of the PRS in the Web's Resource Description Framework (RDF). If this turns out to be impossible, RDF cannot be the solution for data modelling at Renault.

From a general perspective, Renault's PRS can be modelled as a Constraint Satisfaction Problem (CSP). Renault has developed several tools, based on a compiled representation of this CSP, for the many PRS-related questions that need to be answered in day to day operation of business. Thus, instead of representing PRS using Semantic Web technologies, we decided to represent Constraint Satisfaction Problems as such. This also bridges the gap between CSP and Semantic Web, and facilitates exchanges with academic partners, for instance for benchmark purposes. Yet, so far, we limited ourselves to the kind of CSP used at Renault (CSP with finite domain variables), and do not pretend to cover the whole question of CSP in RDF, which may come at a later stage.

From Renault's perspective, requirements on the representation of PRS objects include that the representation language is a published W3C standard and that open-source Java tools support this standard (both for producing and consuming it). Furthermore, its syntax must be serialisable in RDF, or at least allow a serialisation of the main PRS objects, so that the latter can be exchanged with Renault's partners. Succinctness and readability of the syntax is also a requirement in order to limit data transmission bandwidth and to enable Semantic-Web agnostic developers to understand the representation format. Hence, the proposed solution must find the right trade-off between strict standard compliance (no extensions to existing standards are required to be implemented by the Semantic Web client for reading the data), usability and conciseness of the description. Finally, the proposed solution should build upon the existing PRS infrastructure. Note that the purpose of this work is strictly about defining a representation of PRS data, not about providing tools for reasoning over it: all reasoning tasks on the PRS knowledge base are to be left to the existing CSP reasoner at Renault, which is already highly efficient and optimised.

Overall, this paper discusses how we represented PRS objects using Semantic Web technologies, and how we applied this in the industrial context of car manufacturing. The next section introduces Renault's Product Range Specification

and shows how it is modelled as a constraint network. Section 3 presents some use cases of applications that need to represent, store, manipulate, or exchange different PRS objects. Section 4 gives some related work and discusses our modelling choices regarding a Semantic Web representation of the objects of the PRS. Section 5 presents an RDF Schema representation of the main PRS objects. Section 6 compares this representation with its corresponding OWL representation and in section 6 with its corresponding SPARQL representation. Section 7 highlights a RESTful API developed to provide the functionalities developed over the PRS. Finally, section 8 concludes the paper and gives some future work.

2 Product Range Specification

2.1 Modelling vehicle diversity

PRS is used by Renault to specify all possible car configurations and comes as a lexicon, i.e., a set of discrete variables representing the descriptive features or attributes of a vehicle (body type, engine type, gearbox type, colour, etc.), together with a set of constraints that restrict the possible combinations of variable assignments. A particular vehicle is uniquely defined by a tuple of values, one and only one per variable. Constraints invalidate some of the possible car configurations to reflect industrial, engineering or marketing imperatives. The purpose of the PRS is to specify Renault's vehicle diversity, that is, the set of the distinct cars that Renault can build. The size of this set being very big (exceeding 10^{20}), it cannot be enumerated and it must therefore be defined in intention. Yet, product range is not only huge, it is also complex, because of numerous technical, commercial and legal constraints: should every combination of distinctive features and options be possible, the number of distinct vehicles would be in the order of 10^{25} rather than 10^{20}. To address these issues, fast and reliable reasoning services are required. To this aim, the PRS vocabulary and constraints are expressed as a constraint network and a CSP solver is used for reasoning over this constraint network [3].

2.2 Modelling PRS as a constraint network

A constraint network consists of a finite set of variables $\mathcal{X}$ such that each variable $x \in \mathcal{X}$ is associated with a finite domain $D(x)$ denoting the set of values allowed for x, and a finite set of constraints $\mathcal{C}$ that restricts the values the variables can simultaneously take. A constraint $c \in \mathcal{C}$ is a subset of the Cartesian product on domains: $c \subseteq D(x_1) \times D(x_2) \times \ldots \times D(x_n)$. A solution to a constraint network is the assignment of a value to each variable such that all the constraints are satisfied. A constraint network is said to be satisfiable if it admits at least a solution. To a constraint network is associated the constraint satisfaction problem, which task is to determine if such a constraint network is satisfiable. Renault's PRS is modelled as a constraint network, in which a solution (i.e., an assignment of all variables of $\mathcal{X}$) completely defines a particular vehicle. A vehicle range is

defined by a partial assignment of variables of $\mathcal{X}$. Therefore, some constraints will be represented as Boolean expressions on fluents. A *fluent* is a pair (x,A), where $x \in \mathcal{X}$ and $A \subseteq D(x)$. A fluent is *elementary* when A is a singleton (it thus represents an assignment of x).

3 Use cases

3.1 The Bill of Materials

The Bill of Materials, which is the process of defining the parts used in each vehicle of the range, is organised in "generic parts": a generic part is a function fulfilled by a part. For instance, the steering wheel, as a function, is a generic part which may be fulfilled by different steering wheels, as parts, depending on the vehicle. The relationships between the PRS, the generic parts and the corresponding real parts are defined by Boolean expressions called "use cases": the use case of a part is a Boolean expression (over the variables of the PRS) specifying on which vehicles the part is used. This definition of the references of parts corresponding to a given "generic part" is equivalent to defining a new variable whose part references constitute the list of possible values. These values have to be defined with respect to the variables of the PRS[4]. Basically: any vehicle has one and only one steering wheel, as part.

3.2 Accessing after-sales documentation

One user wants to find, for instance, how to remove the gearbox of the car under repair. Assuming methods are tagged with their subject, this looks like a simple SPARQL query — and the first part of the question, indeed, is a simple SPARQL query, such as "select documents where the subject is gearbox removal". Yet, filtering on the vehicle is more tricky: each document has a "condition" property, which points to the set of vehicles for which the document in question is relevant. This condition must be evaluated against the vehicle to decide whether the document must be returned by the query or not. In other words, the service accessing the technical documentation has two input parameters, one being the vehicle (possibly only partially defined with regards to PRS variables), the other one being a standard SPARQL query over documents described with metadata. We could even consider that any vehicle has its own SPARQL endpoint that provides access to its documentation.

3.3 Exchanging PRS related information with partners

It is not uncommon that automotive constructors need to exchange information related to their respective PRS. This occurs for instance when one of them assembles in its own plants cars conceived by the other one, or sell under its own brand cars conceived and / or assembled by the other one. In such cases, each

[4] See [3] for a description of the controls needed for the Bill of materials.

one has its own definition of the PRS. The core of it originates of course from the constructor who conceived the model. Yet, even before adding its own marketing constraints, the other one will first rephrase this PRS using the terms it is used to. This corresponds to the definition of new variables and attached values, based on the original ones, though the use of Boolean expressions . Several variables can be involved in the definition of a given value. Here's a real world example, regarding the seats and their features (such as "adjustable height with lumbar support") : one of the constructor uses the two variables "driver's seat" and "passenger's seat", while the other one uses "left seat" and "right seat". Cross definition of the values involves the variable "traffic flow direction".

This boils down to the creation of translation tables between PRS vocabularies. Tools to assist in the creation of such translations needs (for the GUI) information about the variables and values (typically what we put into a RDF description, such as labels, etc.). CSP based computation is necessary to control the validity of the translation table.

4 Representing constraints on the Semantic Web

Constraint satisfaction is an important reasoning paradigm in artificial intelligence. Constraints are essentially declarative, which makes them very well suited for knowledge sharing and reuse [10]. In [11], an XML format is proposed that can be used to represent CSP instances, as well as quantified or weighted CSP instances. While no standard formalism currently exist to represent constraints on the Semantic Web, several languages have been proposed to extend existing Semantic Web standards in order to express various types of constraints.

4.1 Representing constraints in OWL

Different constraint languages based on OWL/SWRL have been proposed. For example, CIF/SWRL [6] extends SWRL with quantifiers and nested implications in order to express complex range-restricted constraints. [9] further extends CIF/SWRL to add disjunction, negation in rule antecedents and the ability to use any OWL description in the scope of quantifiers. A complementary approach can be found in [5], where OWL is extended to support arithmetic constraints. But these languages are not currently Semantic Web standards and no implementation is to be found. Besides, the semantics of OWL does not seem well suited for constraint checking because it makes the open world assumption and it does not adopt the unique name assumption. Indeed, though a constraint can be represented in many different ways, including mathematical inequalities, logical formulas or matrices, it can be seen conceptually as the set of all legal compound labels for a set of variables [14]. In this regard, constraint satisfaction is very close to database theory, and a constraint satisfaction problem can even be expressed as a database-theoretic problem [15]. This has important implications regarding the choice of the formalism to use to represent CSPs. In

particular, constraint satisfaction problems typically apply a closed world assumption[5] and a unique name assumption[6] as in database modelling, which is usually not the case on the Semantic Web [8]. For this reason, providing OWL with a constraint-checking mechanism would require in some cases to grant subsequent axioms with alternative semantics [13]. For example, value constraints can be expressed using range restrictions but can not be checked using direct OWL inference. The following axiom could be used to state that a car must have its `version` variable set, and its value is either `generic` or `other`:

$$\texttt{Car} \sqsubseteq \exists\, \texttt{version}.\{\texttt{generic}, \texttt{other}\}$$

Yet, due to the open world assumption, there is no way to ensure that the constraint is not violated on a particular dataset using strict OWL semantics, since a missing value for the variable `version` in an OWL knowledge base would not cause logical inconsistency.

4.2 Representing constraints in SPARQL

In the Semantic Web community, propositions have been made to assimilate constraints to the SPARQL queries that can be triggered on a given RDF dataset to check for their validity [1, 4, 13]. For example, the above constraint could be expressed as the following SPARQL ASK query:

```
ASK { NOT EXISTS {
  ?x :version ?y . FILTER (?y != :generic && ?y != :other)
} }
```

In this approach, a constraint is modelled as the evaluation of a graph pattern on a dataset, and checking its validity amounts to running the query. One of the main benefits of this approach lie in the expressivity of the subsequent constraint language (the SPARQL query language has the expressive power of the relational algebra, as shown in [2]). Besides, SPARQL/SPIN [4] provides an RDF Schema for SPARQL queries so that they can be serialised in RDF. SPIN is not a Semantic Web standard but provides an implementation[7] on top of the Jena API, which is already used by Renault.

4.3 Representing constraints in RIF

As Renault's main requirement is to use a Semantic Web standard to represent Boolean expressions, we also considered the W3C's Rule Interchange Format (RIF)[8]. In particular, RIF's dialect RIF-BLD could fairly suit Renault's needs

[5] The closed world assumption states that if a fact is not explicitly stated, then it is assumed to be false.

[6] The unique name assumption states that two different identifiers can not refer to the same individual.

[7] http://www.topquadrant.com/topbraid/spin/api

[8] http://www.w3.org/TR/rif-core/

as an interchange format for constraints but it still suffers from a lack of implementations, and at the time of writing no consistent RDF serialisation is to be found. It can be hoped that future developments on RIF would include defining a dedicated dialect for constraints, as well as an RDF serialisation format.

4.4 Discussion

Regarding the user's requirements, RDF(S), SPARQL/SPIN and OWL appear to be the best candidates to represent PRS objects. Our hypothesis here is that an RDF(S) vocabulary might be sufficient to express the main PRS objects, all the more so that Renault has no need for any constraint-checking mechanism. For these reasons, we started with a direct translation of the main PRS objects into a lightweight RDF Schema, and then studied how this representation translates to OWL and SPARQL/SPIN.

5 An RDF(S) representation of the main PRS objects

5.1 Variables and their domains

The basic building blocks of the PRS is a set of variables and their associated domains of values. A variable can be seen as a function associating a value to

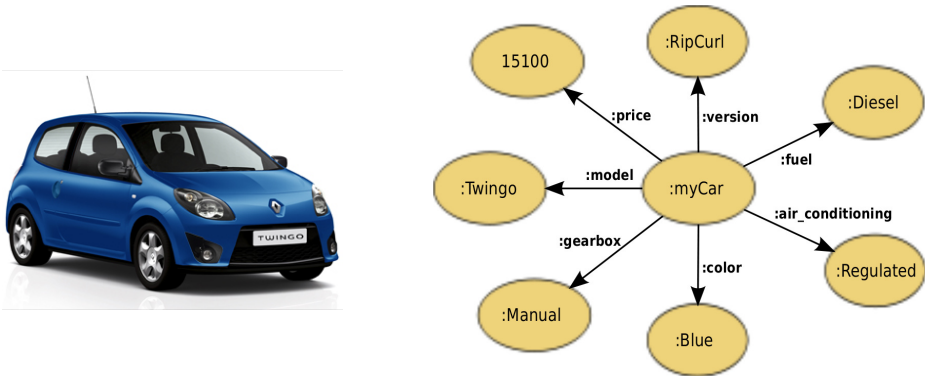

Fig. 1. The RDF representation of a particular car.

an object (here a vehicle). Therefore, it is modelled as an RDF property:

```
csp:variable a rdf:Property ;
  rdfs:domain csp:Solution .
```

The domain of values of a given variable is specified by the `rdfs:range` property:

```
:fuel rdfs:subPropertyOf csp:variable ;
  rdfs:label "The car fuel type." ;
  rdfs:domain :Vehicle ;
  rdfs:range [ owl:oneOf (:Diesel :Petrol :Electric) ] .
```

In this example, the variable `fuel` can take only 3 different values: `Diesel`, `Petrol` or `Electric`. A particular vehicle is assigned a URI and described using the values it takes for different variables (Fig. 1).

5.2 Fluents

A fluent (x,A), where $x \in \mathcal{X}$ and $A \subseteq D(x)$ is the association of a variable and a subset of its domain of values. Two properties are introduced to model a fluent in RDF: the property `csp:var` links a fluent to its variable and the property `csp:val` links a fluent to its different values. For example, the fluent (`fuel`,{`Petrol`,`Diesel`}) is represented by the RDF graph given in Fig. 2. In this figure, the empty node represents a blank node, but this blank node can be replaced by a URI in order to provide the fluent with an identifier.

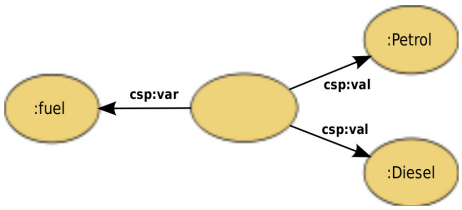

Fig. 2. The RDF representation of the fluent (`fuel`,{`Petrol`,`Diesel`}).

5.3 Boolean expressions

To represent Boolean expressions on fluents, a class `csp:BoolExpr` is introduced, along with 3 subclasses `csp:And`, `csp:Or`, and `csp:Not`, that model logical sub-expressions. The Boolean operators $\wedge$, $\vee$, and $\neg$ can be viewed as functions taking Boolean expressions as arguments. Therefore, three properties `csp:and`, `csp:or`, and `csp:not` are introduced, to link an operator to its arguments (Fig. 3). The subject of each property can either be a blank node or a URI, thereby enabling each Boolean expression to be given an identifier.

5.4 Subsets of the product range

This vocabulary can be used to represent subsets of the product range. For example, the set of cars with manual gearbox and either diesel or petrol type of

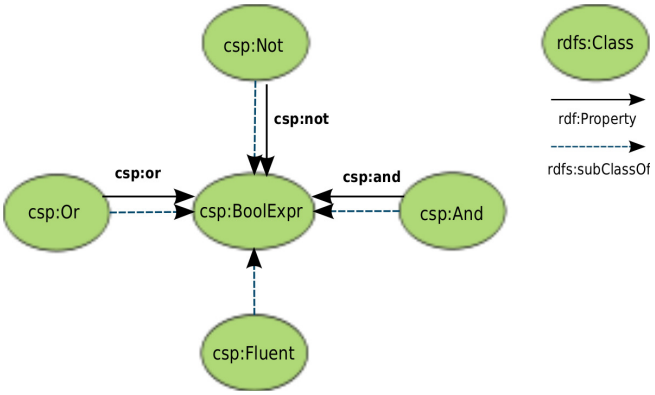

Fig. 3. An RDF Schema to represent Boolean expressions.

fuel corresponds to the conjunction of fluents:

$$(\texttt{gearbox}, \{\texttt{Manual}\}) \wedge (\texttt{fuel}, \{\texttt{Petrol}, \texttt{Diesel}\})$$

which is written in RDF (here with Turtle syntax) as:

```
:myCarSet a csp:And ;
  csp:and
    [csp:var :gearbox ; csp:val :Manual],
    [csp:var :fuel ; csp:val :Petrol,:Diesel] .
```

5.5 Constraints given in intension

Constraints are given either in intension or in extension. Constraints given in intension are represented using Boolean expressions on fluents. An example of such constraint expressed in propositional logic is: $\texttt{Electric} \implies \texttt{NoGearbox}$. This constraint states that electric cars have no gearbox. We could have introduced a representation of the Boolean operator $\implies$ in our vocabulary, but it would be only syntactic sugar since all Boolean operators can be rewritten using solely the $\wedge$, $\vee$, and $\neg$ operators. Indeed, the previous constraint can be rewritten as the following Boolean expression on fluents:

$$\neg(\texttt{fuel}, \{\texttt{Electric}\}) \vee (\texttt{gearbox}, \{\texttt{NoGearbox}\})$$

This Boolean expression is represented in RDF as:

```
:myConstraint a csp:BoolExpr ;
  csp:or
    [csp:not [csp:var :fuel ; csp:val :Electric]],
    [csp:var :gearbox ; csp:val :NoGearbox] .
```

5.6 Constraints given in extension

Other constraints are given in extension, and simply consist in the list of all valid combinations of values of a set of variables. Consider for example the compatibility array given in Tab. 1 for the variables `fuel` and `gearbox`. In this

gearbox \ fuel	Diesel	Petrol	Electric
Manual	x	x	
Automatic	x	x	
NoGearbox			x

Table 1. A compatibility array for the variables `fuel` and `gearbox`.

array, the crosses specify the valid combinations of values of these two variables. So the interpretation of such a compatibility array is that uncrossed combinations are illegal, i.e., no solution of the constraint network should include one of these combinations. For some compatibility arrays, it is also required that each crossed combination must be satisfiable, i.e., there must exist at least one solution of the constraint network that includes this combination. A list of valid combinations of values for a set of variables is represented in RDF(S) by a list of tuples, i.e., an RDF list of conjunctions of elementary fluents. The presence of the additional requirement regarding the satisfiability of each crossed combination is modelled by adding an attribute `csp:isSatisfiable` that points to a Boolean literal. So the compatibility array given in Tab. 1 is represented in RDF by:

```
:myRelation a csp:Relation ;
  csp:isSatisfiable
    "true"^^<http://www.w3.org/2001/XMLSchema#Boolean> ;
  csp:supports (
    [csp:and
      [csp:var :fuel ; csp:val :Diesel],
      [csp:var :gearbox ; csp:val :Manual]]
    [csp:and
      [csp:var :fuel ; csp:val :Petrol],
      [csp:var :gearbox ; csp:val :Manual]]
    [csp:and
      [csp:var :fuel ; csp:val :Diesel],
      [csp:var :gearbox ; csp:val :Automatic]]
    [csp:and
      [csp:var :fuel ; csp:val :Petrol],
      [csp:var :gearbox ; csp:val :Automatic]]
    [csp:and
      [csp:var :fuel ; csp:val :Electric],
      [csp:var :gearbox ; csp:val :NoGearbox]]) .
```

6 Comparison with OWL and SPARQL/SPIN

The syntax of language presented so far seems to be very close to the OWL RDF syntax and the transformation from our language to OWL is quite straightforward. For example, the properties `csp:or`, `csp:and` and `csp:not` can be translated into the OWL properties `owl:intersectionOf`, `owl:unionOf` and `owl:complementOf`. A fluent can also be represented in OWL as an existential restriction. For example, conjunction of fluents

$$(\texttt{gearbox}, \{\texttt{Manual}\}) \wedge (\texttt{fuel}, \{\texttt{Petrol}, \texttt{Diesel}\})$$

which was introduced in section 5.4 to represent the set of cars with manual gearbox and either diesel or petrol type of fuel can be written in OWL as:

```
:myCarSet rdf:type owl:Class ;
            owl:equivalentClass
            [ rdf:type owl:Class ;
              owl:intersectionOf (
              [ rdf:type owl:Restriction ;
                owl:onProperty :fuel ;
                owl:someValuesFrom
                [ rdf:type owl:Class ;
                  owl:oneOf (:Petrol :Diesel)]]
              [ rdf:type owl:Restriction ;
                owl:onProperty :gearbox ;
                owl:someValuesFrom
                [ rdf:type owl:Class ;
                  owl:oneOf (:Manual)]])] .
```

However, Renault has currently no need of using OWL semantics for constraint satisfaction as the reasoning is done by its external CSP solver, and the OWL RDF syntax is somewhat cumbersome. Thus, adopting OWL RDF syntax only as a serialisation means might be of limited interest. On the other hand, providing a complete representation of a CSP instance in OWL that would enable constraint satisfaction by the means using OWL inference would require supplementary axioms (Tab. 2). These axioms correspond to a set of constraints that are implicitly added by the CSP solver at execution time but are not explicitly represented in the definition of the CSP instance. They would be needed in OWL e.g., to make the unique name assumption and to ensure that in any solution of the CSP a variable is assigned one and only one value.

A translation to SPARQL/SPIN syntax would be also quite straightforward. A Boolean expression corresponds to a SPARQL abstract query. The `csp:and`, `csp:or` and `csp:not` operators translate respectively to a set of basic graph patterns, a UNION of basic graph patterns, and a NOT EXIST filter expression. For example, the above conjunction of fluents could be expressed as the following SPARQL query, in which `?this` is the special SPIN variable that binds to the current instance of the class the constraints apply to:

```
ASK {
  ?this :gearbox :Manual .
  { ?this :fuel :Diesel } UNION { ?this :fuel :Petrol }
}
```

The SPIN serialisation of this query gives:

```
:myCarSet    a     sp:Ask ;
     sp:where ([ sp:object :Manual ;
                 sp:predicate :gearbox ;
                 sp:subject spin:_this
               ] [ a     sp:Union ;
                 sp:elements (([ sp:object :Diesel ;
                                 sp:predicate :fuel ;
                                 sp:subject spin:_this
                               ]) ([ sp:object :Petrol ;
                                 sp:predicate :fuel ;
                                 sp:subject spin:_this
                               ]))
               ]) .
```

However, expressing PRS constraints in SPARQL/SPIN seems a bit artificial since no constraint-checking mechanism is needed by Renault, which relies on its own CSP solver to check whether a constraint (or a set of constraints) is satisfied or not.

Constraint	**Additional Axiom**	**Description**
Unique Name Assumption	`DifferentIndividuals`	Two different variable (resp., value) identifiers can not refer to the same individual.
var at most 1	`FunctionalProperty`	In any solution of the CSP a variable is assigned at most one value.
var at least 1	`SomeValuesFrom`	In any solution of the CSP a variable is assigned at least one value.

Table 2. Some additional axioms that would need to be added to an OWL knowledge base to enable constraint satisfaction using OWL inference.

7 Adhering to the Linked Data principles

One last step is required to seamlessly use these Boolean expressions in a Linked Data architecture: recognising them as first class citizens, a status they truly deserve, as they do represent very concrete "real world things", i.e. precisely characterised sets of vehicles, in the case of Renault PRS. To do so, we also need to assign them URIs, and — as per compliance with the Linked Data principles — making those URIs dereferenceable, and returning information about corresponding subset of the range (first of all their definition in RDF) when they

are accessed. However, as the possible number of distinct cars (and therefore of subsets of the range) is so huge, we cannot use URIs that are truly opaque: the server in charge of returning information about them must be able to reconstruct the Boolean expression from the URI (otherwise, we would need a database with an infinite number of lines to store them), without storing all the possibilities in an internal database or RDF store. All services to PRS can then be integrated on the enterprise Linked Data bus as REST services. As an illustration, and to highlight the interest of this compliance to Linked Data principles, let us consider the configuration question.

Configuration is a very classical and challenging problem regarding PRS. It is defined as the process of choosing interactively a vehicle by defining its features, the CSP solver ensuring that only possible choices are proposed to the user [7]. The configuration process can be seen as a traversal of Linked Data provided by a REST service. At any step, the configuration state (i.e, the list of choices already done) corresponds to a Boolean Expression over the variables of the PRS (typically, a conjunction of elementary fluents). Following our model, a URI provides, in particular, the links to the following step in the configuration process (i.e. the choices that can be picked up), as well as any appropriate marketing information (including price, description of features, etc.)

The HTML page presented to the user can be built from that data. Some RDFa in the page maintains the link to the "real world thing" it is about — a car that is partially defined, in other word, a subset of the range. Which is a very interesting thing to take care of: it captures indeed the exact expression of the customer's choice at the moment. This opens some interesting possibilities, in particular from a marketing point of view. One obvious example is related to recommendations (considering for instance the association of configuration with Facebook's OpenGraph[9] protocol and its "like" buttons). It makes no doubt also that the inclusion of RDFa data linking to open description of products in e-commerce context, such as GoodRelations[10], will soon be useful for the structured information it can provide to search engines.

8 Conclusion and Future Work

In this paper, an RDF(S) representation of the main objects related to Renault's Product Range Specification have been provided. We discussed the various requirements to build this, as well as our modelling decisions and the way our model relates with other Semantic Web standards such as OWL or SPARQL. In addition, we discussed design patters for representing PRS on the Web, and their applicability for end-user applications.

While our model focus on the particular PRS representation at Renault, future work may include extensions to the model to cope with other PRS needs, as well as using the related model as an interoperability format between various CSP solvers. Yet, while being specific to our particular use-case, we demonstrated

[9] http://ogp.me
[10] http://www.heppnetz.de/projects/goodrelations

in this paper how Linked Data principles and Semantic Web technologies can be used to model one of the core components of the manufacturing and commercial process in the automotive industry.

Acknowledgements. The work presented in this paper is funded in part by Science Foundation Ireland under grant number SFI/08/CE/I1380 (Lion 2) and by a research grant from Renault SA.

References

1. Faisal Alkhateeb, Jean-François Baget, and Jérôme Euzenat. Constrained Regular Expressions in SPARQL. Technical report, EXMO - INRIA Rhône-Alpes, Grenoble, France, 2007.
2. Renzo Angles and Claudio Gutierrez. The Expressive Power of SPARQL. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 114–129, Berlin, Heidelberg, 2008. Springer-Verlag.
3. J.M. Astesana, L. Cosserat, and H. Fargier. Constraint-based modeling and exploitation of a vehicle range at Renault's: Requirement analysis and complexity study. In *ECAI2010, Configuration Workshop Proceedings*, 2010.
4. Christian Fürber and Martin Hepp. Using SPARQL and SPIN for Data Quality Management on the Semantic Web. In *Business Information Systems, 13th International Conference*, volume 47 of *LNBIP*, pages 35–46. Springer, 2010.
5. Hak-Jin Kim, Wooju Kim, and Myungjin Lee. Semantic Web Constraint Language and its application to an intelligent shopping agent. *Decision Support Systems*, 46(4):882 – 894, 2009. IT Decisions in Organizations.
6. Craig McKenzie, Peter M. D. Gray, and Alun D. Preece. Extending SWRL to Express Fully-Quantified Constraints. In *RuleML 2004*, volume 3323 of *LNCS*, pages 139–154. Springer, 2004.
7. Bernard Pargamin. Vehicle Sales Configuration: the Cluster Tree Approach. In *ECAI2002, Configuration Workshop Notes*, 2002.
8. Peter F. Patel-Schneider and Ian Horrocks. A comparison of two modelling paradigms in the Semantic Web. *Journal of Web Semantics*, 5(4):240 – 250, 2007.
9. Alun Preece, Stuart Chalmers, Craig McKenzie, Jeff Z. Pan, and Peter M.D. Gray. A semantic web approach to handling soft constraints in virtual organisations. *Electronic Commerce Research and Applications*, 7(3):264 – 273, 2008.
10. Alun D. Preece, Kit ying Hui, W. A. Gray, Philippe Marti, Trevor J. M. Bench-Capon, Zhan Cui, and Dean M. Jones. Kraft: An agent architecture for knowledge fusion. *Int. J. Cooperative Inf. Syst.*, 10(1-2):171–195, 2001.
11. Olivier Roussel and Christophe Lecoutre. XML Representation of Constraint Networks: Format XCSP 2.1. *CoRR*, abs/0902.2362, 2009.
12. François-Paul Servant. Semantic Web Technologies in Technical Automotive Documentation. In *Proceedings of OWLED 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
13. Jiao Tao, Evren Sirin, Jie Bao, and Deborah L. McGuinness. Integrity Constraints in OWL. In *AAAI 2010*. AAAI Press, 2010.
14. Edward Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.
15. Moshe Y. Vardi. Constraint satisfaction and database theory: a tutorial. In *Proceedings of ACM PODS '00*, pages 76–85, New York, NY, USA, 2000. ACM.
16. David Wood, editor. *Linking Enterprise Data*. Springer, 2010.