Integration of Ontologies and Constraint Satisfaction Problems for Product Configuration

M. Mohammad Amini, Thierry Coudert, Élise Vareilles, Michel Aldanondo

Integration of Ontologies and Constraint Satisfaction Problems for Product Configuration

M. Mohammad Amini[1], T. Coudert[1], E. Vareilles[2], M. Aldanondo[3]
[1]INP-ENIT, University of Toulouse, Tarbes, France
[2]ISAE-SUPAERO, University of Toulouse, France
[3]IMT Mines Albi, University of Toulouse, Albi, France
(maryam.mohammadamini@enit.fr, thierry.coudert@enit.fr, elise.vareilles@isae-supaero.fr, michel.aldanondo@mines-albi.fr)

Abstract - **In this work, the domain is a product or system configuration. We focus on ontologies, Constraint Satisfaction Problems (CSPs), and their integration. The aim is to capitalize knowledge and define mechanisms that will permit reasoning to find configuration problem solutions regarding customers' requirements. On one hand, an ontology is used to formalize and capitalize knowledge about products or systems structure including concepts, systems, subsystems, components, relationships, attributes, and their possible values. Protégé 5.5.0 is used to create the ontology. On the other hand, we used CSPs to formalize the relationships between attributes values or between concepts that are allowed or forbidden. CSPs (restricted to compatibility tables) are translated into rules to be integrated into the ontology using the SWRL. Therefore, we defined a filtering algorithm based on arc consistency to restrict the domains by removing inconsistent values. First, related works on ontologies, CSPs are presented. The formalization of the ontology, CSPs, and their translation into SWRL rules and their use are presented. Finally, an illustrative application based on the configuration of a simplified bike is presented.**

Keywords – **Product configuration, Knowledge, Ontology, Constraints Satisfaction Problem**

I. INTRODUCTION

The context of industry 4.0 implies that companies have to answer quickly to many customers' demands and to offer solutions that match perfectly numerous and often complex requirements. In this context, configuration tools which permit selection, among a great number of systems, subsystems, components, etc., those that will permit to satisfy the customer requirements are useful. It is necessary to model knowledge about all these items and their integration and to define efficient tools which can use this knowledge to provide solutions. This work will take place in the context of System Engineering (SE) and in particular in systems or products configuration. The problem is to choose solution elements (systems, subsystems, components, attributes values) to integrate, produce and deliver to the customer.

In this work, we focus on the integration of two formalisms: ontology and Constraint Satisfaction Problem (CSP). The ontology provides a shared understanding of the different items that compose products. It is useful to structure, formalize and capitalize knowledge and reuse it. An ontology represents a set of classes of the domain and the relationships between them and also attributes values. We used Protégé 5.5.0 to model and edit ontology. CSPs allow formalizing the relationships between variables' values which are allowed or forbidden. Then, a CSP represents the restrictions on the combination of variables domains. CSPs are restricted in this work to compatibility tables and they are translated into rules to be integrated into the ontology in Protégé 5.5.0 using the SWRL language.

To reuse this capitalized knowledge during a product configuration process, it is necessary to be able to reason based on the ontology and the rules representing the different constraints. The aim is to restrict the domains of the different characteristics of products following their structure, the constraints, and the customer requirements. That means that it is necessary to remove inconsistent values from the domains of the variables. Standard reasoners associated with ontologies do not allow non-monotonic reasoning. Therefore, they do not allow removing values from variables domains. To solve this problem, we propose to use an arc-consistency-based algorithm that allows removing inconsistent values.

The contribution of this paper is to propose first models and methods that permit integration of ontology with CSPs and reason to configure products.

In section 2, related works about product configuration, ontologies, CSPs, and their integration are presented. Then, in section 3, we describe the representation of knowledge using the ontology, the formalization of constraints using rules, and the generic algorithm which allows configuring products. In section 4, an illustrative application based on a simplified bike composed of a frame is presented. Two constraints are defined to link the frame sizes to bike users and the bike colors to bike users. Finally, in section 5, the conclusion and future works are presented.

II. RELATED WORKS

As mentioned, our application domain is SE and more specifically product configuration. It is considered as the problem of designing a product. This product is designed after assembling some components which are predefined and their connections are only possible in specific ways [1]. In other words, a configuration can be defined as using a set of pre-defined components while considering a set of restrictions on how the components can be

combined. [2] also presented various types of relationships that could exist between components such as aggregation and generalization.

A. Ontology

So far different definitions were mentioned for ontology regarding the various contexts. [3] defined an ontology as a "*formal, explicit specification of a shared conceptualization* ". In this definition, several words play an important role. Formal refers to a machine-readable representation, explicit means that the type of concepts used and the constraints on their uses are explicitly defined, shared refers to the fact that the knowledge represented in an ontology are agreed upon by a group, and conceptualization refers to an abstract model of the world that describes knowledge by using various concepts and their relationships.

Applying ontologies to improve System Engineering is attracting attention [4]. [5] presented an inclusive review of ontology-based systems engineering. This paper attempted to clarify what, where and how ontologies are used in the field of SE and using what languages, tools, and methods. [6] and [7] are two review articles on product configuration that presented various definitions in this area as well as future road-maps. [8] used conceptualizations of four approaches, namely connection-based, resource-based, product structure-based, and function-based approaches in the definition of the ontology.

B. Constraint Satisfaction Problems

[9] defined a CSP as a triplet {X, D, C}. It includes a set of variables (X), a set of domains of variables (D), and a set of constraints (C). Constraints represent restrictions on the combination of variable values. A CSP model attempts to assign values to the variables to satisfy all constraints. [10] presented a CSP-based approach for modeling distributed configuration problems. [11] presented an object-oriented approach to develop a web-based configuration design system. It represented the structures and rules of a configurable bicycle and then applied the invasion algorithm. The product configuration problem was also modeled as CSP. Considering the definition of CSP, [12] defined the configuration task as a CSP (X, D, C) where C is the union of both the configuration knowledge base and the user requirements.
An empty variable is a variable without any value in its domain. A non-valuated variable is a variable which in its domain the number of values is more than one. Valuated variable is a variable that has been assigned to one and only one consistent value.

C. Ontology and CSP integration

A few articles dealt with both ontology and constraints. [13] represented an ontology-based approach for modeling product configuration knowledge. It used Web Ontology Language (OWL) to define classes and their relationship to formalize product configuration then used SWRL to define the constraints. The constraints are mainly concerning the structure of products. The proposed approach was applied to configuring the ranger drilling machine. [14] presented an Ontology-based method for product configuration knowledge. They used OWL, SWRL, and a rule engine called JESS to improve the product configuration system. The approach was applied to a case for the personal computer. [15] presented an ontology-based approach for product extension services configuration. It also used OWL, SWRL, and JESS. The approach was applied to an example of configurable product extension services. As far as we know, these approaches are based on the modulization by rules of some constraints on allowed or forbidden associations of components. They are not explicitly integrating ontologies and CSPs.

For product configuration, ontologies are a powerful model of knowledge to represent the different items at different abstraction levels and what are their relationships. They can formalize knowledge about products and their structure by creating classes, properties, individuals, etc. However, they cannot represent constraints directly and impose to the user to use specific components or to define a specific value for an attribute. In contrast to ontologies, CSPs have difficulties to represent abstract concepts and abstract relationships. However, they can formalize relationships between attributes values, or between classes or concepts that are allowed or forbidden. That is the reason in this article we propose an approach which permits to integrate them and reason directly to configure products.

III. PROPOSITION

A. Ontology to represent knowledge about product structure and characteristics

To build and edit the ontology for product configuration, Protégé 5.5.0 is used. It can also be used for reasoning using some integrated reasoners. The Web Ontology Language (OWL) is a standard Semantic Web language that is designed for ontology representation. Within the ontology, the elements we have to model are classes, sub-classes, properties, and individuals (or instances).

A class provides an abstraction mechanism for gathering individuals with common characteristics. Classes allow modeling the different generic items: products, sub-products, components... They are stored in the ontology to represent hierarchical relationships between classes and sub-classes (taxonomy). An individual is an instance of a class. Classes relationships and individuals' relationships are represented using object properties. Another kind of property is data property. A

data property permits to define attributes of a class with their datatype (integer, string, etc.).

Therefore, using the ontology, it is possible to capitalize all the classes (or concepts) that are used by the company to design the products as well as their hierarchical relationships (generalization and specialization of classes within the taxonomy). It is also possible to represent the composition relationships between some classes (e.g. a *Car* is composed of exactly four *Wheels*). Data properties and object properties are named "variables" in the next sections. The ontology is a powerful piece of knowledge for product configuration which permits to represent structures of the different products with their characteristics but it does not allow modeling constraints to restrict the different choices for the designer.

B. Rules to represent constraints

In this article, we consider only constraints formalized using tables of compatibility. A table of compatibility is composed of rows corresponding to variables and lines corresponding to the allowed values for the variables (extension representation). This kind of constraint can be easily represented and filtered by standard filtering tools (Choco-solver, CoFiADe…). However, these tools are not integrated with ontologies. Therefore, we propose to represent constraints using rules which are integrated with ontologies using standard first-order logic languages as SWRL.

The Semantic Web Rule Language[1] (SWRL) is an expressive OWL-based rule language of the Semantic Web. All rules are expressed based on the OWL concepts: classes, properties, and individuals. SWRL provides powerful deductive reasoning capabilities and that is why it is chosen in our proposition.

An SWRL rule consists of two parts: an antecedent part (or body) and a consequent part (or head) separated by an arrow "→" such that (Atom ^ Atom ^ …) → (Atom ^ Atom ^...). Both parts include conjunctions of atoms. The consequent of the rule fires if and only if every atom in the antecedent is satisfied.

An atom is an expression of the form: *P(arg1, arg2, …).* *P* is a predicate symbol that could be a class or property and *arg1*, *arg2*, ... can be variables, individuals, or data properties. An atom can be on the form C(? x), P(?x, ?y), sameAs(x, y) or differentFrom(x, y). C is an OWL class, P is an OWL property, and x, y are either variables, OWL individuals, or OWL data values. Only variables that occur in the antecedent of a rule may occur in the consequent.

For instance, the atom C_1(?i) (resp. C_2(?j)) links the variable *i* to an instance of the class C_1 (resp. the variable *j* to an instance of C_2). The atom is*ComposedOf(?i, ?j)* links the variable *i* (i.e. an instance of the class C_1) to the variable *j* (an instance of C_2). It is then a composition relationship and the atom allows to check if *i* is composed of *j*. SWRL supports various built-ins including comparison built-ins (swrlb:lessThan, swrlb:greaterThanOrEqual, etc.), mathematical built-ins (swrlb:add, swrlb:multiply), etc. An atom can also add data properties or object properties. Considering a data property *Prop*, the execution of the atom *Prop(?x, 15)* in a consequent will add the integer value "15" as a new attribute value for the instance *x*.

However, SWRL does not support non-monotonic reasoning which means we cannot change values or remove inconsistent ones, we can only add new properties. To consider this problem and define a product configuration method, we have proposed an algorithm which is described in the next section.

C. Algorithm

The proposed algorithm (named *Configure*) is based on standard arc-consistency algorithms and allows to remove non-consistent values from the different domains when changes have been done by the application of rules (i.e. the constraints) or by the user. An impacted variable is a variable linked by a constraint to another variable for which the domain has been changed.

Algorithm *Configure*
Begin
Set up domains to their initial range
While (we don't have any empty variables AND there is still non-valuated variables)
The user chooses a variable and restricts its domain
Add the impacted variables to S_{imp}
While (S_{imp} is not empty)
Modify the domains of impacted variables (applying corresponding rules)
Make the intersection of new domains and previous domains
Add impacted variables in S_{imp} only if their domain has been changed
End while
End While
If (each variable is valuated) then there is one solution
If (there is an empty variable) then there is no solution
End

To illustrate the propositions, an illustrative application based on the configuration of a simplified bicycle is presented in the next section.

IV. ILLUSTRATIVE APPLICATION

In this section, the application that consists in configuring a bicycle is presented. First, the ontology corresponding to bicycles is created. Second, two constraints (tables of compatibility) are given and translated into SWRL rules. Third, the algorithm

[1] https://www.w3.org/Submission/SWRL/

Configure is applied following the interactions with the user.

A. Ontology for Bikes

Using Protégé 5.5.0, several classes are created and stored into the ontology of Fig. 3. These classes are *User*, *System*, *Bike*, *Frame*, and *Color*. The classes *Bike* and *Frame* are sub-classes of the class *System*.

Several individuals are also created. Bike1, Bike2, …, Bike 5 are instances of the class *Bike*. They correspond to five different categories of bikes. Five instances of *Frame* are created and three colors Pink, Black, and Blue are created as instances of *Color*. Three object properties are created: *isComposedOf*, *hasUser* and *hasColor*. One data property *frameSize* is also created to define the possible frame sizes (integers between 10 and 25).

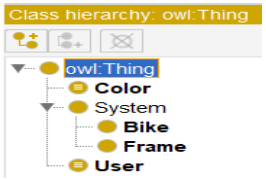

Fig. 1. Classes

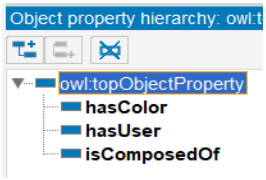

Fig. 2. Object properties

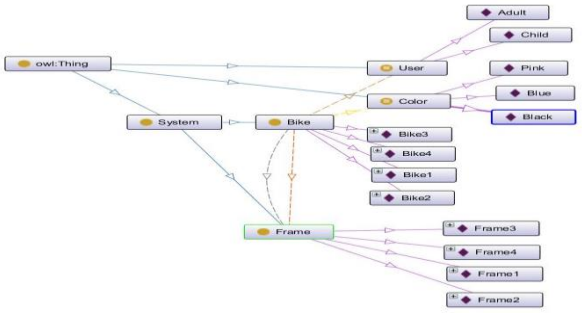

Fig. 3. Ontology of bikes

B. Model of constraints and SWRL rules

Two constraints are created with the tables of compatibility (TABLE I, TABLE II). The variables are the frame size (FS), the bike user (USR), and the color (CL).

TABLE I

CONSTRAINT CT_1

FS	USR
[10, 17]	Child
[16, 25]	Adult

TABLE II

CONSTRAINT CT_2

USR	CL
Child	Pink
Child	Blue
Adult	Blue
Adult	Black

In TABLE I, the constraint CT_1 is formalized using the R_1 to R_4. In TABLE II, the constraint CT_2 is formalized using the R_5 to R_9. All these rules are formalized and integrated within the ontology using SWRL. It is important to notice that the rules consider the object properties, the data properties, the individuals, and the constraints. The R_1 can be interpreted as follows. If the bike *i* is composed of the frame *j* AND the frame *j* has a frame size *x* which is greater than or equal to 10 AND *x* is less than or equal to 17 THEN the bike user of the bike *i* is "Child". When the rule is applied, if the body is true, then the object property "Child" is added to the new domain of the bike user variable.

TABLE III

RULES CORRESPONDING TO THE CONSTRAINT

R_1	Bike(?i) ^ Frame(?j) ^ isComposedOf(?i, ?j) ^ frameSize(?j, ?x) ^ swrlb:greaterThanOrEqual(?x, 10) ^ swrlb:lessThanOrEqual(?x, 17) → hasUser(?i, Child)
R_2	Bike(?i) ^ Frame(?j) ^ isComposedOf(?i, ?j) ^ frameSize(?j, ?x) ^ swrlb:greaterThanOrEqual(?x, 16) ^ swrlb:lessThanOrEqual(?x, 25) → hasUser(?i, Adult)
R_3	Bike(?i) ^ hasUser(?i, Child) ^ Frame(?j) ^ isComposedOf(?i, ?j) → frameSize(?j, 10) ^ frameSize(?j, 11) ^ frameSize(?j, 12) ^ frameSize(?j, 13) ^ frameSize(?j, 14) ^ frameSize(?j, 15) ^ frameSize(?j, 16) ^ frameSize(?j, 17)
R_4	Bike(?i) ^ hasUser(?i, Adult) ^ Frame(?j) ^ isComposedOf(?i, ?j) → frameSize(?j, 16) ^ frameSize(?j, 17) ^ frameSize(?j, 18) ^ frameSize(?j, 19) ^ frameSize(?j, 20) ^ frameSize(?j, 21) ^ frameSize(?j, 22) ^ frameSize(?j, 23) ^ frameSize(?j, 24) ^frameSize(?j, 25)
R_5	Bike(?i) ^ hasUser(?i, Child) →hasColor(?i, Pink) ^ hasColor(?i, Blue)
R_6	Bike(?i) ^ hasUser(?i, Adult) → hasColor(?i, Black) ^ hasColor(?i, Blue)
R_7	Bike(?i) ^ hasColor(?i, Pink) → hasUser(?i, Child)
R_8	Bike(?i) ^ hasColor(?i, Blue) → hasUser(?i, Child) ^ hasUser(?i, Adult)
R_9	Bike(?i) ^ hasColor(?i, Black) → hasUser(?i, Adult)

C. Illustration of configuration

We illustrate the configuration using the algorithm *Configure* with a simple scenario. For the three variables FS, USR, and CL, the initial domains of validity are $10 \leq FS \leq 25$, USR∈ {Child, Adult}, CL∈ {Pink, Blue, Black}. The variable FS is represented as a data property that takes values within the integers range.

The user decides to restrict the domain of the variable FS in order to know what are the users and colors allowed by the constraints: $d_{FS} = 10$. The variable USR is the only impacted variable considering the R_1 to R_4 (corresponding to CT_1). Applying these rules, the new domain of USR is $d_{USR}^{CT_1} = \{Child\}$. The intersection of the new domain and previous domain gives:

$$d_{USR} = d_{USR} \cap d_{USR}^{CT_1} = \{Child, Adult\} \cap \{Child\} = \{Child\}$$

Therefore, as the domain of USR has been modified, the impacted variables are FS (following R_3 and R_4) and

CL (following R_5 and R_6). The R_3, R_4, R_5, R_6 are applied to define the new domains of CL and FS:
$d_{CL}^{CT_2} = \{Pink, Blue\}$, $d_{FS}^{CT_1} = [10, 17]$. The intersection of the new domains with the previous domains gives:
$d_{FS} = d_{FS} \cap d_{FS}^{CT_1} = 10 \cap [10, 17] = 10$
$d_{CL} = d_{CL} \cap d_{CL}^{CT_2} = \{Pink, Blue, Black\} \cap \{Pink, Blue\} = \{Pink, Blue\}$

As the domain of CL is modified, the variable USR is the only impacted variable considering R_7, R_8, and R_9 (CT_2). Then, these rules are applied to obtain: $d_{USR}^{CT_2} = \{Child\}$. The intersection of the new domain and the previous domain gives:
$d_{USR} = d_{USR} \cap d_{USR}^{CT_2} = \{Child\} \cap \{Child\} = \{Child\}$

The domains have not been modified but the user chooses the color to be pink: $d_{CL} = \{Pink\}$. USR is the only impacted variable. The R_7, R_8, and R_9 are applied: $d_{USR}^{CT_2} = \{Child\}$. The intersection with the previous domain gives:
$d_{USR} = d_{USR} \cap d_{USR}^{CT_2} = \{Child\} \cap \{Child\} = \{Child\}$

There are no new impacted variables because the domains have not been changed. The obtained solution is then:
$d_{USR} = \{Child\}$, $d_{CL} = \{Pink\}$, $d_{FS} = 10$. Therefore, if the designer chooses a frame size of 10 and a pink color for a bike, this bike is only suitable for children.

V. CONCLUSION

In this work, we proposed an approach to integrate ontology and Constraint Satisfaction Problem (CSP). The aim was to define, using the ontology, a model of knowledge which can represent the different products, sub-products, components structures, and characteristics as well as their relationships (composition, properties...). The ontology helped us to represent classes at different abstraction levels. The ontology is suitable to represent possible associations of items for several products. However, it does not allow to represent constraints such that the choice of a specific component impacts the choice of another one. Moreover, it is not possible to guarantee that the designer has selected the right number of items even if it is specified by object properties. For instance, it is possible to represent the knowledge about a car that must be composed of four wheels, but it is not possible to define a constraint that imposes it. Therefore, to solve the problem, CSPs have been chosen to model constraints between variables values. As CSPs are not directly integrated within ontologies, we have proposed to translate constraints into SWRL rules. Because SWRL rules do not support non-monotonic reasoning, we defined an arc-consistency-based algorithm to restrict the domain and remove inconsistent values for product configuration.

The limits of the approach concern the translation of constraints into SWRL rules when constraints imply several variables and when structures of products are complex. Moreover, to execute specific rules and make the intersections between domains, it is necessary to develop an external program as a plug-in to connect to Protégé. In order to better integrate CSP within ontologies, we will investigate the use of SHACL and SPARQL languages. The SHACL language can model constraints directly in the ontology and reason with them. However, the reasoning only shows what are the unsatisfied constraints. SPARQL will also be studied concerning non-monotonic reasoning requirements.

REFERENCES

[1] F. Mittal, S., & Frayman, "Towards a Generic Model of Configuration Tasks," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 89, pp. 1395–1401, Nov. 1989.

[2] D. Sabin and R. Weigel, "Product configuration frameworks - A survey," *IEEE Intell. Syst. Their Appl.*, vol. 13, no. 4, pp. 42–49, 1998, doi: 10.1109/5254.708432.

[3] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and methods," *Data Knowl. Eng.*, vol. 25, no. 1–2, pp. 161–197, 1998, doi: 10.1016/S0169-023X(97)00056-6.

[4] N. Hallberg, E. Jungert, and S. Pilemalm, "Ontology for systems development," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 24, no. 3, pp. 329–345, 2014, doi: 10.1142/S0218194014500132.

[5] L. Yang, K. Cormican, and M. Yu, "Ontology-based systems engineering: A state-of-the-art review," *Comput. Ind.*, vol. 111, pp. 148–171, 2019, doi: 10.1016/j.compind.2019.05.003.

[6] L. L. Zhang, "Product configuration: A review of the state-of-the-art and future research," *Int. J. Prod. Res.*, vol. 52, no. 21, pp. 6381–6398, 2014, doi: 10.1080/00207543.2014.942012.

[7] G. Oddsson and K. R. Ladeby, "From a literature review of product configuration definitions to a reference framework," *Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM*, vol. 28, no. 4, pp. 413–428, 2014, doi: 10.1017/S0890060413000620.

[8] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, "Towards a general ontology of configuration," *Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM*, vol. 12, no. 4, pp. 357–372, 1998, doi: 10.1017/s0890060498124083.

[9] U. Montanari, "Networks of constraints: Fundamental properties and application to picture processing," *Inf. Sci. (Ny).*, vol. 7, pp. 95–132, 1974.

[10] D. Jannach and M. Zanker, "Modeling and solving distributed configuration problems: A CSP-based approach," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 3, pp. 603–618, 2013, doi: 10.1109/TKDE.2011.236.

[11] S. K. Ong, Q. Lin, and A. Y. C. Nee, "Web-based configuration design system for product customization," *Int. J. Prod. Res.*, vol. 44, no. 2, pp. 351–382, 2006, doi: 10.1080/00207540500244153.

[12] J. Felfernig, A., Hotz, L., Bagley, C., & Tiihonen, *Knowledge-Based Configuration From Research to Business Cases*. Newnes, 2014.

[13] D. Yang, M. Dong, and R. Miao, "Development of a product configuration system with an ontology-based approach," *CAD Comput. Aided Des.*, vol. 40, no. 8, pp. 863–878, 2008, doi: 10.1016/j.cad.2008.05.004.

[14] D. Yang, R. Miao, H. Wu, and Y. Zhou, "Product configuration knowledge modeling using ontology web language," *Expert Syst. Appl.*, vol. 36, no. 3 PART 1, pp. 4399–4411, 2009, doi: 10.1016/j.eswa.2008.05.026.

[15] J. Shen, L. Wang, and Y. Sun, "Configuration of product extension services in servitisation using an ontology-based approach," *Int. J. Prod. Res.*, vol. 50, no. 22, pp. 6469–6488, 2012, doi: 10.1080/00207543.2011.652744.