Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 9, No. 4, 2021, 555-570.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

An Executive Model to Improve Reasoning and Realization in Ontology using Fuzzy-Colored Petri Nets

Mojtaba Shokohinia, Abbas Dideban * and Farzin Yaghmaee

Department of Electrical and Computer Engineering, Semnan University, Semnan, Iran,

Article Info

Article History:
Received 12 April 2021
Revised 11 June 2021
Accepted 01 November 2021

DOI:10.22044/JADM.2021.10706.2206

Keywords:
Reasoning, Ontologies, Unified Modelling Language (UML), Fuzzy-Colored Petri Nets (FCPNs).

**Corresponding author :adideban@semnan.ac.ir (A. Dideban).*

Abstract

Despite the success of ontology in knowledge representation, its reasoning is still challenging. The main challenge in the reasoning of the ontology-based methods is to improve the reasoning process realization. The time complexity of the realization problem-solving process is equal to that of NEXP Time. This can be achieved by solving the subsumption and satisfiability problems. In addition, uncertainty and ambiguity are inevitable in these characteristics. Considering these requirements, using fuzzy theory is necessary. A method is proposed in this work in order to overcome this problem, which provides a new solution with a suitable time position. This work aims to model and improve the reasoning and realization in an ontology using Fuzzy-Colored Petri Nets (FCPNs). To this end, an algorithm for improving the realization problem is presented. Then, the Unified Modeling Language (UML) class diagram is used for standard description and representation of the efficiency characteristics. The Resource Description Framework Schema (RDFS) representation is converted to the UML diagram. Then, the fuzzy concepts are introduced in FCPNs. Then, an algorithm for converting the ontology description based on the UML class diagram into an executive model based on FCPNs is presented. Using this approach, a simple method is developed in order to obtain the desired results from an executive model and reasoning based on FCPNs through various queries. Finally, the efficiency of the proposed method is evaluated. The results obtained show that the performance of the proposed method is improved from various aspects.

1. Introduction

Recently, many real-world and experimental applications have been developed that focus on the data, that originates from the challenges posed by the data nature. One of these challenges is the efficient and accurate processing of the real-world data, or data analysis, in order to infer the data and extract its inherent logic and laws. When the data is semantic and inference should also be semantic and correspond to the data, a challenge is required. Ontology is one of the concepts that can significantly influence this area and solve most of the related problems. Therefore, it is used as an efficient tool in this context [1]. Petri net is a graphical and mathematical tool used in various systems such as discrete event systems, intelligent systems, and communication protocols. Ontology and modeling can be combined in order to obtain a favorable result for reasoning, which is of great importance. Fuzzy-colored Petri nets (FCPNs) have attracted attention since their properties and mathematical basis make them suitable for modeling ontology [2]. Yim et al. [3] have proposed a method that uses Petri nets to evaluate the ontology of Location-Based Services (LBS). In this method, a model of the ontology based on Petri nets was first developed and then analyzed by running the Petri net. Zhu et al. [4] have proposed an approach in which a platform has been developed to evaluate the efficiency of the software systems using UML and CPNs. For this purpose, the case diagrams and UML were used to obtain the efficiency data of a software system. Then, these models were transformed into hierarchical CPNs that allow evaluating the efficiency of the software systems. In [5], a workflow scheduling method based on fuzzy-colored Petri nets has been proposed. Compared to the other methods, which are simpler and more informal, the ontology-based methods have more capabilities and advantages, in terms of simplicity and integration. However, the development of ontology models based on Ontology Web Language Description Logics (OWL-DL) shows that this technique is insufficient when it comes to define and understanding complicated descriptions and relationships. This problem originates from the fact that the existing builders of OWL-DL are selected such that the reasoning procedures support decision-making, and any other procedure in OWL-DL is not reasonable and determinable [6]. Recently, some studies have been conducted in the Semantic Web community in order to extend the capabilities of OWL-DL. They have led to logical languages such as the Semantic Web Rule Language (SWRL), which has been used in [7]. A new method for extending queries has been presented in [8]. This method, which is a combination of the relevant feed-back and latent semantic analysis, finds the terms relative to the topics of the user's original query based on the relevant documents selected by the user in the relevant feed-back step. However, the main problem with OWL-DL is reasoning, which requires a large amount of computation. Combining and integrating laws make reasoning indeterminable. Over and above all, ontology-based reasoning in OWL-DL suffers from other problems. The natural solution for obtaining complex data using ontological reasoning is to solve the realization problem. The realization problem is about finding the most associated class of an object. Unfortunately, the realization problem is a problem with NEXP Time complexity. Online execution of ontology-based reasoning has fundamental problems in terms of scalability and execution time, especially when the ontology includes a large number of individuals [9]. With this in mind, in this paper, we present a modeling method and propose an algorithm in order to improve the realization problem with FCPNs. In general, the advantages of this innovation can be summarized as follows:

1. Decomposing the realization problem into some sub-categories and solving these sub-categories in a reasonable time and with a proper proficiency.
2. The logical division of the problem into some sub-categories provides computational simplicity and logical support for the realization requirements.
3. This work aims to find a solution to improve reasoning in ontology based on FCPNs. Using fuzzy logic has improved reasoning.
4. The fuzzy concepts in colored Petri nets are introduced. A clearer and simpler explanation than fuzzy logic is given for semantic reasoning with Petri nets. The rest of the paper is organized as what fallows.

First, a description of the ontology is given, and all aspects of the ontology are described using the UML. Also an easy-to-develop execution model is presented, which primarily aims to develop an execution model for ontology reasoning based on CPNs. Then, an algorithm is presented in order to convert the ontology description based on the UML class diagram into an executive model based on CPNs. UML is an extremely important tool that can be used to describe ontology. In Section 2, the basic concepts are introduced. In Section 3, the proposed method is presented, which is done in different steps. First, the ontology is represented using RFDS, and then described using UML class diagram. Moreover, the ontology description based on the UML class diagram is introduced into the CPN-based model and a standard description using UML is presented. Finally, the fuzzy concepts in FCPNs are introduced. Fuzzy Petri nets are a type of Petri nets in which the principles of fuzzy logic are observed. Fuzzy Petri nets can model and represent the fuzzy systems and the systems based on fuzzy data. With fuzzy Petri nets, we can achieve the best of both worlds, in the sense that we can benefit from the advantages of Petri nets and fuzzy logic at the same time. Fuzzification, inference, and defuzzification are applied to the fuzzy Petri net according to the defined functions. In Section 4, the results obtained are implemented and evaluated by developing an ontology about automobiles and applying queries. By running the ontology model, the ontology structure can be evaluated from different aspects. The topic that has attracted attention in this work is reasoning in ontology queries, where the executive reasoning model based on CPNs can dynamically receive various queries and present the corresponding results favorably. In each section, we consider the reasoning structure. We can also interrupt the reasoning at each step. In Section 5, the proposed method is compared with the HermiT and FaCT++ reasoners in terms of speed and accuracy, and the results are provided. Finally, Section 6 concludes the work.

2. Basic Concepts

2.1. Ontology

Definition 1 [10]: Ontology is defined as a 5-tuple model: $O = \{ C; H^c; R; rel; A^o \}$, where *O* is the name of the ontology, *C* is the set of concepts, and *Hc* is a taxonomy of concepts with multiple inheritances. For example, $H^c (C1;C2)$ states that *C1* is the sub-concept of *C2*. *R* is a set of non-taxonomic relations described by their domain and rank constraints. *rel(R)* describes a hierarchy of relations. For example, *rel(R) = (C1; C2)* indicates that there is a relation R between *C1* and *C2*. A^o is the set of axioms. In [11], a new ontology-based approach to detect human activity from GPS data has been presented, aiming to detect cross-linguistic plagiarism. A framework called Multilingual Plagiarism Detection (MLPD) has been presented for cross-linguistic plagiarism analysis, aiming to detect plagiarism.

2.2. Reasoning and Descriptive Logic

Reasoning is considered to be at the heart of many domains, such as machine learning, system analysis, context-aware systems, search engines, and reasoning engines. In fact, reasoning provides the concept of understanding and intelligence in various applications. In particular, the concept of reasoning and perceptions between the data is defined in Ontology Web Language (OWL) and Resource Description Framework (RDF) standards. In other words, the goal of meaning and semantic reasoning is to use the semantic data and ontology in specific applications. Descriptive Logic (DL) provides a formal and logical definition for ontologies and the semantic web, in general, the DL models concepts, roles, individuals, and their correlations. An axiom (i.e., a logical statement related to the roles and/or concepts) is modeled in the most basic concepts in DL. DL uses different terminologies for nomination than first-order logic (FOL) and OWL. In general, ontology languages presented for the semantic web are a syntactic variant of DL [12]. The authors of [13] have proposed the BUNDLE algorithm to compute the probability of queries from DISPONTE knowledge bases that follow the ALC semantics. The explanations are encoded in a Binary Decision Diagram from which the query probability of the query is computed. The experiments performed by applying BUNDLE to probabilistic knowledge bases show that it can handle ontologies of realistic sizes. This reasoner does not support realization and fuzzy logic [14]. CEL is a reasoner for small description logic that can be used to compute the subsumption hierarchy induced by EL++ ontologies.

The most outstanding feature of CEL is that, unlike all the other modern DL reasoners, it is based on a polynomial-time subsumption algorithm, which allows it to process very large ontologies in a reasonable time. This reasoner does not support realization and fuzzy logic. In [15], DBOWL has been provided, which is a persistent and scalable OWL reasoner. Ontologies are stored in a relational database, where a description logic reasoner is used to precompute the class and property hierarchies to obtain all the ontology information (i.e.., properties, domain, and range), which is also stored in the database. Moreover, a simple but expressive query language has been implemented to query and reason about these ontologies. This reasoner does not support realization and fuzzy logic. [16] has described DELOREAN, the first ontology reasoner that supports fuzzy extensions of the standard languages OWL and OWL 2. In a strict sense, DELOREAN is not a reasoner, but a translator from fuzzy rough ontology languages to classical ontology languages. This allows using classical Description Logic inference engines to reason with the representation resulting from the transformation. With large ontologies posing a challenge for reasoners with ever-increasing data generation rates, large ontologies challenge the reasoners from both perspectives in terms of memory and computation power. In such cases , the distributed reasoners provide a viable solution. The authors of [17] have presented a distributed approach to EL + ontology classification, called DistEL, was where it was shown that the classifier can handle large ontologies and the classification time decreases as the number of nodes increases. However, this reasoner does not support fuzzy logic and satisfiability.

DRAGON makes systematic systematically using a general abstract model to represent each one of the knowledge sources necessary for automatic recognition of continuous speech recognition [18]. The model of a probabilistic function of a Markov process is very flexible, and leads to features that allow DRAGON to function despite high error rates of individual knowledge sources. The repeated use of a simple abstract model produces a system that is structurally simple, but powerful in capabilities. The main features of the DRAGON system are 1) delayed decisions; 2) generative form of the model;, 3) hierarchical system;, 4) integrated representation;, and 5) general theoretical framework. Various sources of knowledge are organized in a hierarchy of probabilistic functions of Markov processes. This reasoner does not support satisfiability, realization, and fuzzy logic. ELepHant reasoner, a consequence-based reasoner for the fragment of DLs, has been proposed in [19]. The optimizations, implementation details, and experimental results for classification of several large biomedical knowledge bases have been introduced. The consequence-based EL+ reasoner ELepHant, has also been presented, and the implementation details have been described and the experimental results have been presented. This reasoner does not support fuzzy logic and satisfiability. The authors of [20] have provided the main features of the fuzzyDL system (in terms of syntax and semantics), which is an expressive reasoner for fuzzy DLs. They also have shown that fuzzyDL significantly extends the fuzzy SHIF system by allowing several additional features [20]. The also showed two use cases, namely logic-based matchmaking and fuzzy control, which are not yet supported by any other fuzzy DL system. This reasoner does not support realization.

2.3. Fuzzy Theory Architecture

A fuzzy system is a suitable solution for uncertain environments, where the ambiguity probability is high. Figure 1 shows a general architecture of the systems based on the fuzzy theory. The fuzzy logic and fuzzy system architecture have been described in [21][22][23][24].

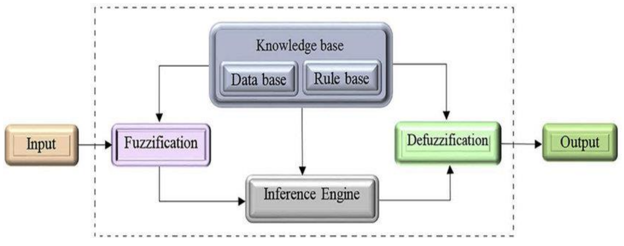


Figure 1. The architecture of the fuzzy theory system and its main components.

2.4. Colored Petri Nets (CPNs)

CPN is a discrete-event modeling language that combines the capabilities of Petri nets and a high-level programming language [25][26], and is used to build models related to isotropy and the analysis of its properties [27][28][29].

A formal definition of CPNs is as follows:

Definition 2: A CPN is defined as a 9-tuple (∑, P, T, A, N, C, G, E, I) [30], where:

∑: is a finite set of nonempty types called sets of color ;

P is a finite set of places;

T is a finite set of transitions;

A is a set of arcs such that $\cap T = P \cap A = T \cap A = \emptyset$;

N is a node function which is defined from to $P \times T \cup T \times P$.

C is a color function which is defined from P to ∑.

G is a guard function that is defined from T to an expression and $\forall t \in T: [Type(G(t)) = Bool \,\hat{}\, Type(Var(G(t)))]$;

E is an arc expression function; and

I is an initialization function.

2.5. Unified Modeling Language

Developments in software engineering have led to the emergence of more methods and tools for describing and documenting the software systems. Accordingly, many of the problems in the realization and maintenance of the software systems have been solved. UML is an important and practical by-product of software engineering with applications in many other areas. UML has various diagrams, that represents different views of the software system [31]. An important advantage of the UML diagrams is their extensibility, i.e., they can describe any feature using the annotation function of UML. In other words, if the diagram in question cannot describe a feature, it will describe that feature using the annotation concept [32].

3. Problem Definition and Solution

Ontology reasoning has clear limitations in its implementation. Therefore, this work aims to find a solution to improve it based on CPNs. The main limitation of ontology reasoning is its inability to solve the realization problem. So far, various solutions have been used to overcome this shortcoming. The approach used in this work is to separate the realization problem logically. In other words, instead of choosing a particular subcategory of ontology, this work finds a solution to the realization problem by decomposing it into some sub-categories, and then solving these sub-categories. In this way, computational complexity and inefficiency are avoided by minimizing the problem. In general, the realization problem can be divided into the following subcategories:

*** Satisfiability of concept:** Diagnosis of the concept considering the one to which the individual belongs, based on the description of the individual.

*** Subsumption of concept:** Determining whether the concept d follows from c, i.e. c is more general than d.

In contrast, the realization problem can be defined as "finding a concept considering to which individual has the most attachment." In order to solve either of the above problems, it is necessary to separate them and integrate their results. All the following formal definitions were taken from [33]. As currently the data and especially its hidden semantics are important, different descriptions have been offered for ontology, whose importance is specified from all viewpoints. Considering the purpose of this paper, a framework that can be used to obtain a representation based on CPNs from ontology is presented. Then the semantic reasoning, i.e. the realization problem, is analyzed.

Definition 3 (realization): Given an assertional box (Abox) of A, concept C, individual a, and a set of concepts, find C as the most specific concept from the set such that $A \vDash C(a)$.

An individual is called a and a collection of concepts are given. Find C (most specific concept) from the collection of concepts such that $A \vDash C(a)$.

Definition 4 (satisfiability): Given T, concept C is satisfiable if the model I exists in T, such that C^I is non-empty.

Definition 5 (subsumption): Let T be a terminological box (Tbox); then a concept C is subsumed by a concept D with respect to T if $C^I \subseteq D^I$ for every model I of T. In this case, we write

$C \sqsubseteq_T D$ or $T \vDash C \sqsubseteq D$.

1. Suppose that A is the collection of concepts in the realization and a is the individual.
2. The satisfiability problem for collection A is solved based on a, and the collection of concepts in collection R is assumed to satisfy a.
3. Subsumption is solved for all possible pairs of collection R, and *MSC(R, a)* is obtained.
4. As satisfiability is reducible to subsumption at the second level, subsumption is used in order to solve the satisfiability problem.

The main challenge is the efficiency of this method for collections with a high number of concepts. In what follows, an algorithm for reducing and optimizing the primary collection of concepts is presented. In other words, a solution to the ontology partitioning problem is derived. After partitioning the ontology into different sub-ontologies, solving the realization problem is performed on one of the sub-ontologies. In ontology partitioning, ontology O is partitioned into a collection of modules, which are not necessarily disjoint such that the union of all modules is equal to O:

Definition 6 (ontology partitioning function):

$Partition(O) \rightarrow$
$M = \{\{M_1, M_2, \ldots, M_n\} | \{M_1 \cup M_2 \cup \ldots \cup M_n\} = O\}$

This ontology procedure is converted into a multipartite graph so that the query result can exist in one or some parts of the graph. If the query result exists in one part of the graph, only that part of the graph is examined since it is independent from the other parts. Other parts of the ontology are not examined. The proof of the presented idea is as follows:

Suppose that, after partitioning the graph, the related part of ABox A (with Tbox T) to query q is p. Then, we solve the satisfiability problem in p:

$$\exists\ I \in p, C^I\ is\ not\ \perp \tag{1}$$

Hence, we find all C in p that are satisfiable, and put them all in R:

$$R = \{R_1, R_2, \ldots, R_n\}, R_i\ is\ satisfiable\ concept\ in\ p \tag{2}$$

Then, for all pairs of satisfiable concepts in R, we solve the subsumption problem as follows:

$$\forall (i1, i2) \in R, \forall\ I\ \in T \quad C^I \sqsubseteq D^I \tag{3}$$

If $i1$ is subsumable under $i2$, then $i1$ is removed from R. We apply relation (3) for every pair of $i1$ and $i2$ since there is no candidate $i1$ and $i2$, and call the new R as R'. Then we apply all concepts in R' to individual a such that:

$$\forall C \in R' \quad if\ A \nvDash C(a)\ then\ remove\ C\ from\ R' \tag{4}$$

Thus R' has a candidate msc(A,a) such that most real specific concept A and a is in R'.

On the other hand, according to theorem 1 and [33], we can solve the satisfiablity problem (relation (1)) only with the subsumption problem. Thus, we find the candidate solutions for the realization problem in a reasonable time, and use only the subsumption problem. The general problem-solving procedure is the same as shown in Algorithm 1. The pseudo-code of the algorithm is shown in Algorithm 2. In line 3 of Algorithm 2, we used the algorithm to partition the ontology based on the approach of [34] and applied the pseudo-code format in Algorithm 1. After partitioning the ontology into parts and selecting one or more relevant parts in the ontology partitioning phase, it is important to combine and merge the query answers from the relevant parts to obtain the final answer. In other words, we are required to prove that the final answer can be generated from instance checking by the independent Abox. Before defining the respective theorem, we should explain some relevant notions [35].

Algorithm 1 : Realization Solving

```
1: Procedure Ontology- Partitioning(A box A, Query q, Individual a)
2: Make Ontology Partitioning using Overlapped-Ontology- Partitioning(A)
3: Select dependent part p according to q from partitions
4: Solve Realization according to p
5: return solve – Realization (p,q,a)
6: end procedure
7: procedure solve – Realization(A box p, Query q, Individual a)
8: Solve Satisfiability(p,q,a) and store result in R
9: while no subsumption problem exists do
10: for each pair of items (i1,i2) in R do
11: solve subsumption(i1,i2) problem and store results in S
12: end for
13: end while
14: return S as a result
15: end procedure
```

Algorithm 2 : Ontology Partitioning

```
1: procedure Overlapped- Ontology- Partitioning (Ontology O)
2: Building the weighted dependency graph WG from an ontology O
3: Finding the common concepts in WG
4: Partitioning the weighted graph WG
5: Using rank removal algorithm for cluster components
6: Extracting partitioned ontologies as PO set
7: return PO
8: end procedure
```

Theorem 1: Independent Abox and instance checking

Two connected Abox $A_1$and $A_2$are given such that $A = A_1 \cup A_2$. If A_1, A_2 are independent then for each query realization, we have Φ and Tbox T:
$< T, A > \models \Phi$ if and only if $< T, A_1 > \models \Phi$ or $< T, A_2 > \models \Phi$.

Proof: (→)

Suppose that A_1 and A_2 are independent, and $\Delta^I_{A_1}, \Delta^I_{A_2}$are domains of $A_1$and A_2; then:

$$\Delta^I_{A_1} \cap \Delta^I_{A_2} = \emptyset \tag{5}$$

For any concept *C*, $C^I_{A_1} \cap C^I_{A_2} = \emptyset$which $C^I_{A_i}$is extended C in $\Delta^I_{A_i}$.

On the other hand, suppose that $< T, A_1 > \not\models Q$ and $< T, A_2 > \not\models Q$, which means:

$$\exists I_1; I_1 \models A_1, I_1 \models T, I_1 \models \neg Q \tag{6}$$

$$\exists I_2; I_2 \models A_2, I_2 \models T, I_2 \models \neg Q \tag{7}$$

where I_1 and I_2 are explanations of A_1 and A_2.

Since $A = A_1 \cup A_2$ and $\Delta^I_1 \cup \Delta^I_2 = \emptyset$, we can create an explanation from A like I, where $I = I_1 \cup I_2$. In other words, $I =< \Delta^I, O^I >$ that declare as follows:

(i) $\Delta^I = \Delta^I_1 \cup \Delta^I_2$

(ii) for any constant a, $a^I = \begin{Bmatrix} a^{I_1} \text{ if a occurs in } A_1 \\ a^{I_2} \text{if a occurs in } A_2 \end{Bmatrix}$

(iii) for any concept, C,$C^I = C^{I_1} \cup C^{I_2}$

(iv) for any role R, $R^I = R^{I_1} \cup R^{I_2}$

Thus, we can conclude from $I_1 \models \neg Q$, $I_2 \models \neg Q$ and (iii):

$$I \models \neg Q \tag{8}$$

which means:

$$(\neg Q)^I = (\neg Q)^{I_1} \cup (\neg Q)^{I_2} \tag{9}$$

where I is the explanation of A. On the other hand, we can conclude from (ii), (iii), and (iv) that:

$$(A)^I = (A_1)^{I_1} \cup (A_1)^{I_2} \tag{10}$$

As A_1 and A_2 are consistent, we just proof no intersection between them.

For Concept C from DL, we have:

$$C^I \subseteq C^I \tag{11}$$

Also, we have:

$$(\neg C)^I = (\Delta^I \backslash \Delta^I) \subseteq \Delta^I \tag{12}$$

Then, for *C*, we have:

$$C^{I_1} \subseteq \Delta^I_1 \text{ and } (\neg C)^{I_2} \subseteq \Delta^I_2 \tag{13}$$

Due to $\Delta^I_1 \cap \Delta^I_2 = \emptyset$,:

$$C^{I_1} \cap (\neg C)^{I_2} = \emptyset \tag{14}$$

This means they have no intersection. Since, we have explanation *I* from *A*:

$$(A)^I \neq Q \text{ and } (\neg Q)^T \neq Q \tag{15}$$

then:

$$I \models A \cap I \models \neg Q \tag{16}$$

which is a $A \not\models Q$ definition.

Therefore, $A \models Q$ if $A_1 \models Q$ or $A_2 \models Q$ that result is $<T,A> \models Q$ if $<T,A_1> \models Q$ or $<T,A_2> \models Q$.

(←)

We assume that $<T,A_1> \models Q$ or $<T,A_2> \models Q$. In both cases we have:

$$<T,A_1 \cup A_2> \models Q \tag{17}$$

where $<T,A> \models Q$. ■

3.1. Ontology Description using UML Diagram

Table 1 shows the concept of ontology and its representative elements in the class diagram. Thus, with the help of the UML class diagram, the ontology structure can be represented along with its efficiency characteristics. However, using the UML class diagram alone, the reasoning execution property is not achieved. Therefore, an executive model is required, for which CPNs have been used in this work. In the following, a model based on CPNs from ontology description based on class diagrams is explained.

Table 1. Mapping in Ontology Description.

Element of the class diagram	Element of ontology
Class	Class
Association Relationship	Object Property
Sub Class	Sub Class
Generalization	Generalization
Attribute	Data Property

3.2. Describing Executive Ontology Reasoning Model based on CPNs

The description of the ontology structure alone is not sufficient for query execution. Therefore, CPNs are used to apply the reasoning executive properties to the ontology structure. For this purpose, a mapping from the structure description based on the UML class diagram to CPNs will be established. Accordingly, some hypotheses for mapping the structure description based on the UML class diagram to the executive model based on CPNs are made, which include the following:

Hypothesis 1: Each class element in the class diagram is converted to a place element in the CPN.

Hypothesis 2: Each relationship element (association and generalization) in the class diagram is converted into a transition element with a sub-network in the CPN.

Hypothesis 3: Each sub-class element in the class diagram is converted into a place element in the CPN.

Hypothesis 4: Each class or sub-class properties is converted into a closet representing tokens of each place.

Hypothesis 5: there is a place element in each sub-net that refers to each association to represent the Object Property information so that the association information is complete for the ontology structure. Therefore, this element has a color set proportional to the tags of stereotype, where each record that contains the name of the domain sample and a list of the range samples.

Hypothesis 6: All samples are in the last sub-classes.

Considering the given rules for mapping the class-based description to an executive model based on CPNs, the mapping samples are shown in Figure 2.

3.3. Converting Ontology based on UML Class Diagram to Executive Model based on CPNs

This section presents an algorithm that can be used to obtain an executive model based on CPNs from an ontology description based on UML. First, algorithm 1 is presented to solve the realization problem optimally for reasoning in an ontology, and then algorithm 3 is applied to model the problem to form CPNs.

Algorithm 3: *Constituting a CPN from the UML class diagram*

```
Input : UML class diagram
Output : CPN model
Stepe 1  1. CList = CList[(ID, As)] //UML Class Diagram Classes with id and attribute list
         2. AList = AList[(C1ID, C2ClassIDList, ObjectPropertyInfoStr, PerformanceStr)] //UML Class Diagram
            Associations
         3. GList = GList[(C1ID, C2ClassIDList, PerformanceStr)] //UML Class Diagram Generalizations
         4. SCList = SCList[(ID, As)] //UML Class Diagram SubClasses with id and attribute list
Stepe 2  5. PList = PList[(ID,Type)]:=new list[]  //CPN Place List
         6. T = (ID, InList, OutList,Time)]  //CPN Transition
         7. SubCPN = (ID,PList,T)  //CPN Sub Model
         8. STList = STList[(ID, InList, OutList,SubCPN)] := new List[] //CPN SubTransition List
         9. CPNModel = (ID, PList, STList)
Stepe 3  10. for each c ∈ CList
         11. PList.Add(c.ID, c.As)
         12. End for each
Stepe 4  13. for each a ∈ AList
         14. SPList := SPList[(ID,Type)] = [PList(a.C1ID), PList(a.C2ID), ("ObjectProperty", a. ObjectPropertyInfo)]
         15. T:= (a.ID, [PList(a.C1ID), PList(a.C2ID), ("ObjectProperty")], [PList(a.C1ID), PList(a.C2ID),
             ("ObjectProperty")], a.Performance.Demand )
         16. SubCPN := (a.ID, SPList, T)
         17. STList.Add(a.ID, InList.AddList([a.C1ID, a.C2ID]), OutList.AddList([a.C1ID, a.C2ID]), SubCPN)
         18. End for each
Stepe 5  19.  for each g ∈ GList
         20.  SPList := SPList[(ID,Type)] = [PList(g.C1ID)]^^[PList(g.C2ClassIDList)]
         21.  T := (g.ID, SPList, SPList, g.Performance.Demand )
         22.  SubCPN := (g.ID, SPList, T)
         23.  STList.Add(g.ID, InList.AddList(SPList), OutList.AddList(SPList), SubCPN)
         24. End for each
Stepe 6  25. CPNModel := ("Main", PList, STList)
```

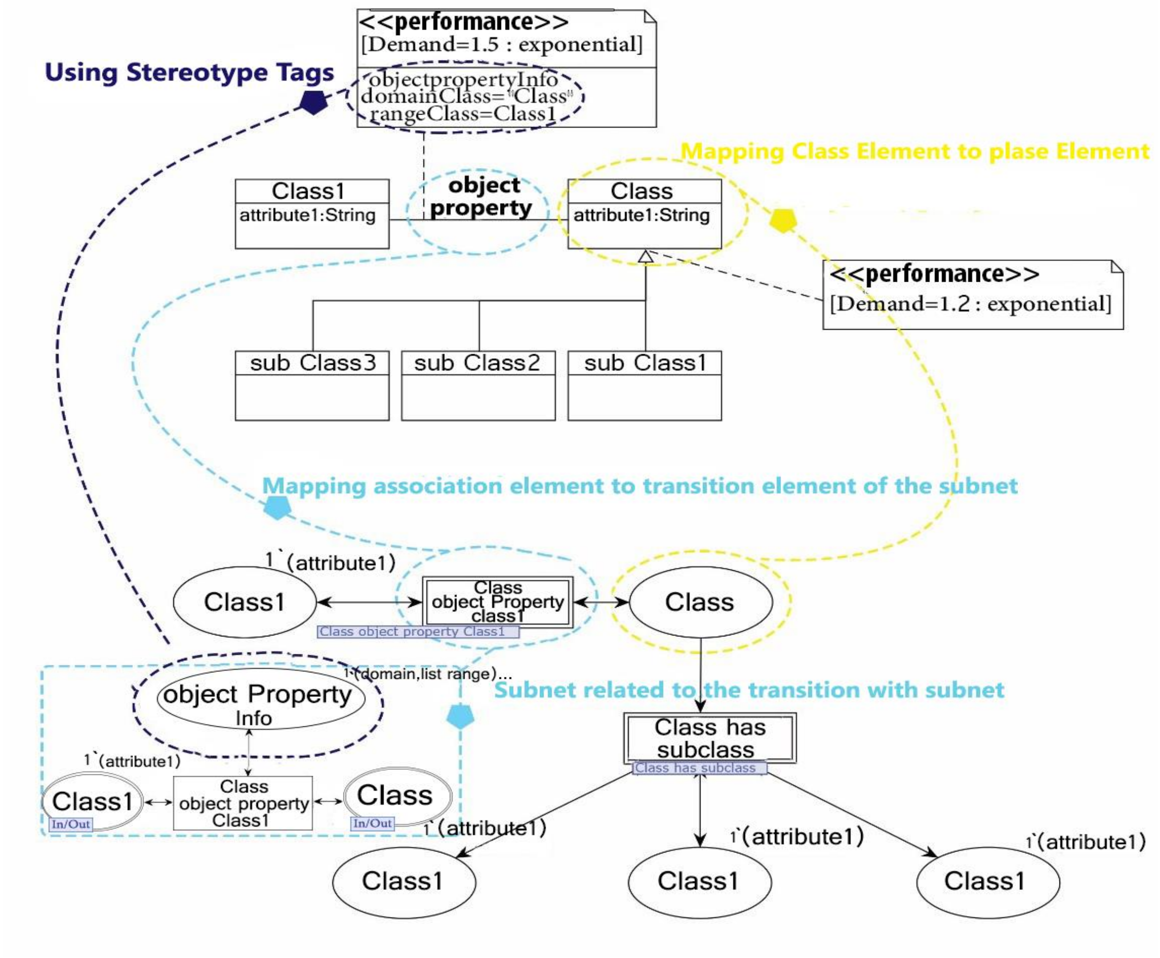


Figure 2. Mapping the class diagram to the model based on CPN.

In step 1, the input of the algorithm includes the elements of the class diagram, which includes a list of classes, a list of relations with respect to associations, a list of relations with respect to generalization, and a list of sub-classes. The elements of the colored Petri net including the list of places, transitions, subCPNs, and subtransitions are created in step 2 and form a model of CPN. In step 3, a place is created for each class in the CList according to hypothesis 1. In step 4, a mapping is created from the association in the UML class diagram to its equivalent in CPN. The relation of the associations is a relation in which an object property is present. For example, the class of the first side of the associations is automobile, while the class of the second side of the associations is feature. These two classes are related by an object property called has_a. Figure 10 shows a class diagram, where this relationship is obvious. In step 5, we create a mapping from the generalization in the UML class diagram to its equivalent in CPN, which are shown in Figure 11. The final model of CPN with model ID, the list of places, and the list of transitions with a sub-network are obtained in step 6.

3.4. Fuzzy Concepts in Colored Petri Nets

In order to obtain more complete and real results in the presence of uncertainties, the fuzzy theory should be used. This work applies fuzzification, inference, and defuzzification to fuzzy-colored Petri nets. Moreover, the steps involved in applying the fuzzy system to the colored Petri net are presented completely for two inputs and one output, which can be extended to other inputs and outputs. The first input is denoted as A1, the second as A2, and the output as A3. A trapezoid is used to represent the membership functions. The input membership functions consist of low, medium, and high membership functions, which are shown in Figure 3. According to this method, the weight of each rule, which is its effectiveness in the inputs and outputs, is calculated. Our rules are applied to the inputs, and the weight is calculated for each rule. The range of each rule at the output is obtained by calculating the weights.

The inputs of the fuzzy system are the tokens present in places and can be any kind of language variable, which are in this section the maintenance

cost and fuel cost. When a fuzzy-colored Petri net is asked, the reasoning is performed considering the question demand. The transition that performs reasoning retrieves the information from the tokens available at places considering its requirement. In order to represent the membership function, a trapezoidal membership function is used. Each trapezoid is composed of four points.
Membership function of the first linguistic input

variable A1;
val A1Low_range = (0,0,a_1,b_1) ;
val A1 Medium_range = (c_1, d_1, e_1, f_1) ;
val A1 High _range = (g_1, h_1, 100, 100)
Membership function of the second input

linguistic variable A2 ;
val A2Low_range = (0,0,a_2,b_2) ;
val A2 Medium_range = (c_2, d_2, e_2, f_2) ;
val A2 High _range = (g_2, h_2, 100, 100) ;
Index i is used in the figure so that it can be extended to an arbitrary number of inputs.

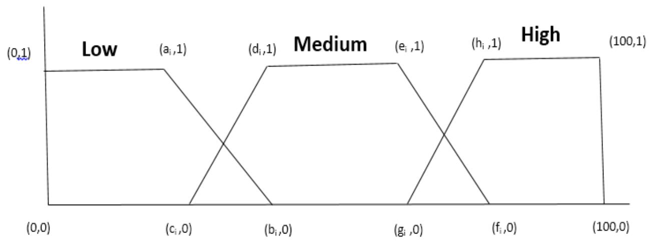


Figure 3. Membership function of input values.

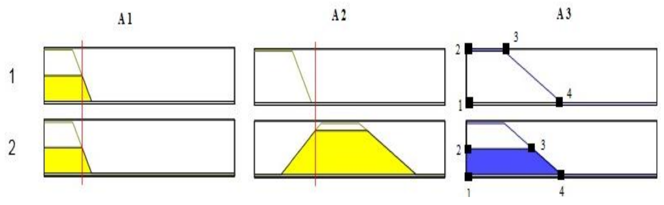


Figure 4. Range of weights after applying the rules.

In this section, considering the rule weights obtained in the previous section, the new form of the membership function shape is obtained, which has a new coordinate. Since the trapezoidal function is used, our range has four points shown in Figure 4. In order to obtain the points, the line equation and the line slope equation are used, shown in Figure 5.

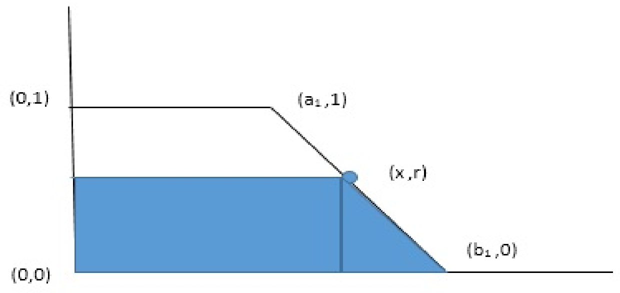


Figure 5. Determining the weight range.

Line slope equation:

$$m = \frac{y-y0}{x-x0} = \frac{1-0}{a1-b1} \tag{18}$$

Line equation:

$$y\text{-}0 = \frac{1-0}{a1-b1}(x-b1) \tag{19}$$

$$r = \frac{1-0}{a1-b1}(x-b1) \tag{20}$$

$$x = r\,(a1 - b1) + b1 \tag{21}$$

val r1_range = (0.0, 0.0, ((#1(r1_w))* $\underbrace{\text{((\#3(A3 Low\_range))}}_{}$ ...

$$\text{val r1\_range} = (0.0,\ 0.0,\ ((\#1(\text{r1\_w}))*\ \overset{a1}{\overline{((\ \#3(\text{A3 Low\_range}))}} - \overset{b1}{\overline{(\#4(\text{A3 Low \_range}))}})) + \overset{b1}{\overline{(\#4(\text{A3 Low \_range}))}}\ ,\ (\#4(\text{A3 Low \_range}))) ;$$

In this step, considering the points obtained in the previous step, the area under the curve of each trapezoid is calculated at the output.

Area of the trapezoid

$$S = \frac{total\ of\ the\ bases * height}{2} = [(c-b) + (d-a) * r] * 0.5 \tag{22}$$

$$\text{val r1\_area} = 0.5*(((\overset{d}{\overline{\#4(\text{r1\_range})}}) - \overset{a}{\overline{(\#1(\text{r1\_range}))}}) + ((\overset{c}{\overline{\#3(\text{r1\_range})}}) - \overset{b}{\overline{(\#2(\text{r1\_range}))}}))) * \overset{r}{\overline{(\#1(\text{r1\_w}))}} .$$

Calculating the area obtained by applying rule a.
After obtaining the area of all rules, the center of each rule is calculated, and the total mean area of all rules is obtained using the center law.

$$\text{center of the area} = \frac{\sum X_i A_i}{\sum X_i} \tag{23}$$

4. Illustration Example

A case study is conducted for each proposed method considering the importance of the issue to evaluate the proposed method. Accordingly, in order to demonstrate the performance of the proposed method, a sample ontology in the context of automobile information is considered, which includes information of the manufacturers of the automobile, tire, and ring, as shown in Figure 6. It also includes information about automobiles of different classes and information about facilities, motor type, and devices of the automobiles. In the further course of the proposed method, one should achieve a description based on the UML class diagram of the ontology. Thus, a complete and standard description is obtained based on the proposed method. In the following, an executive model is developed according to the presented algorithm for developing the executive reasoning model based on CPNs and using the given hypotheses.

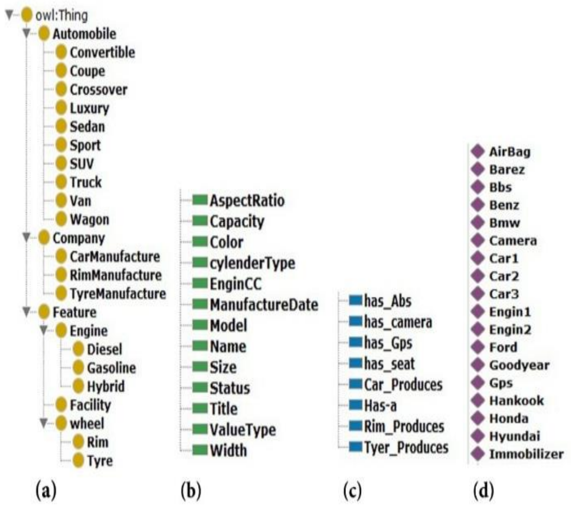


Figure 6. An excerpt from car manufacturing indicating (a) classes, (b) data type properties, (c) object properties, and (d) individuals.

Since the inference process is long, the sub-nets used in the inference steps are not shown.

First Execution: The purpose of the execution is to answer the question "Which companies made which automobiles that cost less than 50?"

Query 1: "?Company has ?Automobiles has Cost Less Than 50.0"

There is a price concept in neither the ontology nor its corresponding colored Petri net, and the fuzzy-colored Petri net is used to obtain the price. Two parameters, Maintenance Cost Fuzzy Value (MCFV) and Fuel Cost Fuzzy Value (FCFV), are used as inputs in order to determine the price. In the first step, the query entered into the automobile has a sub-class sub-net to find the automobiles that cost less than 50. This bus is fuzzy, so the two parameters of Maintenance Cost Fuzzy Value (MCFV) and Fuel Cost Fuzzy Value (FCFV) are considered as the inputs, and the cost values that are not present in the place data are returned as the output. After firing a transition, the automobiles that cost less than 50 are selected, and the price of each one is given in the token information, as shown in Figure 7. Finally, two automobiles that cost less than 50 and are manufactured by Kia and Peugeot are found, as shown in Figure 8. By applying the above query, the following result is found. It refers to Kia, which manufactured Car1, and Peugeot, which manufactured Car3. The execution is completed in 8.65 ms.

1 ["Company = Kia","Automobile = Car1"]
1`["Company = Peugot","Automobile = Car3"]

Second Execution: The purpose of the secend question is to answer the question "which companies have manufactured which automobiles whose ring is Bbs?"

Query: 2`"?Company has ?Automobile has_a ?Rim Rim_Producer Rim_Manufacture is Bbs"

Since the reasoning procedure of the second question is long and more sub-networks are used, only the last section that answers the query is shown. The sub-networks used in the reasoning steps are not shown. In this section, CPN has formed satisfiability set, i.e. the samples are found and the Transition called Get Result transfers the token obtained from executing the sub-networks located in a place called things to the decision. Three samples are found for the second question. The transition called Get Process gives each sample to the item section one by one. Subsumption operation is performed for each sample, separately. After firing the transition called Get Subsumption for the samples, the sub-sumption operation is performed. where the first, second and third responses for the query and its reasoning result and the time taken to perform reasoning for the samples are shown in Figure 9.

Reasoning Result:

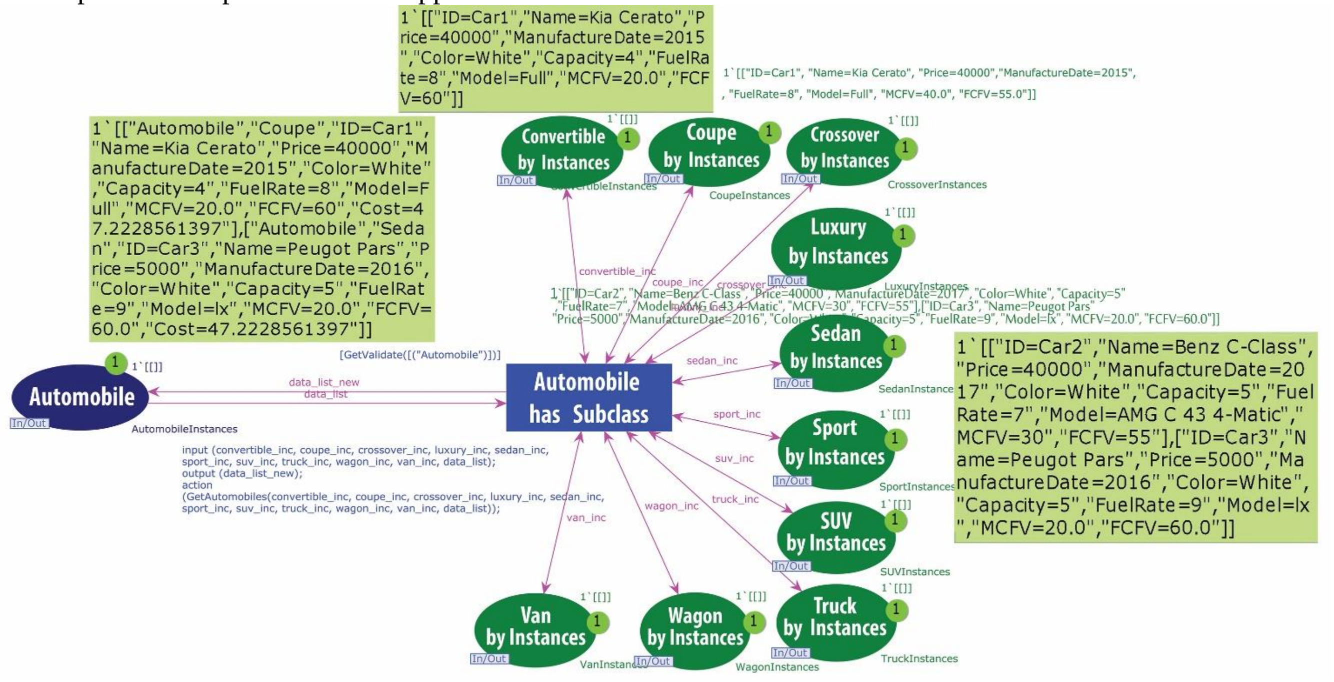
By applying the above query, the reasoning result is obtained in which three samples related to Benz, Kia and Peugeot, which are the manufactures of Car2, Car1 and Car3, and are found, respectively; The execution is completed in 19.48 ms.

1`["Company = Benz","Automobile = Car2","Rim = Rim2"]++
1`["Company = Kia","Automobile = Car1","Rim = Rim1"]++
1`["Company = Peugot","Automobile = Car3","Rim = Rim2"]

Accordingly, the obtained reasoning results are based on the data recorded in ontology, and if the recorded data of the companies is completely defined in the ontology, more real results are obtained and different queries can be applied to the model that process them dynamically and the proposed CPN represents the results of the query after reasoning.

5. Result Analysis

The significance of the current work lies in its proposal of an optimal method for semantic reasoners. Moreover, the reasoner is modeled using the colored Petri net, and all the reasoner steps can be observed and evaluated. Table 4 compares the capabilities and supported elements of the proposed methods with eight reasoner engines.



```
1`[["ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60"]]
```

```
1`[["ID=Car1", "Name=Kia Cerato", "Price=40000","ManufactureDate=2015",
, "FuelRate=8", "Model=Full", "MCFV=40.0", "FCFV=55.0"]]
```

```
1`[["Automobile","Coupe","ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60","Cost=47.2228561397"],["Automobile","Sedan","ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0","Cost=47.2228561397"]]
```

```
1`[[]]
```

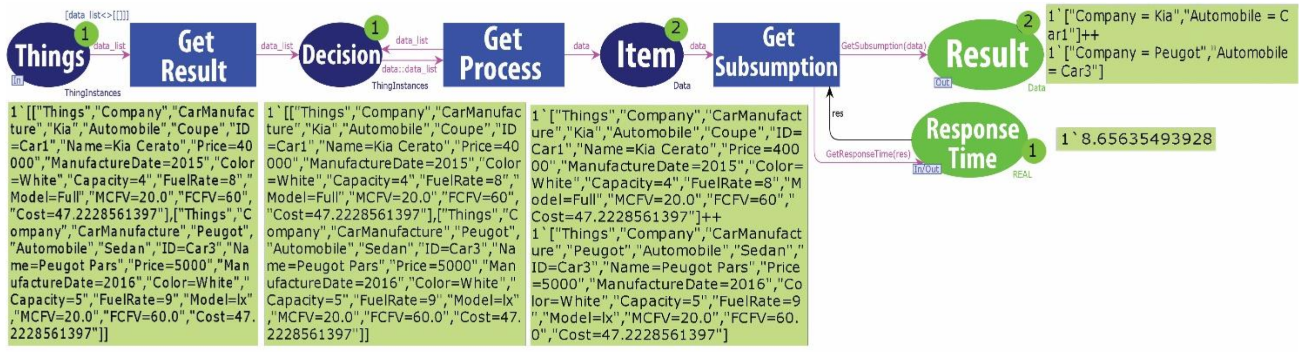
```
[GetValidate([("Automobile")])]
```

```
1`[["ID=Car2", "Name=Benz C-Class", "Price=40000", ManufactureDate=2017", "Color=White", "Capacity=5"
,"FuelRate=7", "Model=AMG C 43 4-Matic", "MCFV=30", "FCFV=55"],["ID=Car3", "Name=Peugot Pars"
"Price=5000","ManufactureDate=2016", "Color=White", "Capacity=5", "FuelRate=9", "Model=lx", "MCFV=20.0", "FCFV=60.0"]]
```

```
1`[["ID=Car2","Name=Benz C-Class","Price=40000","ManufactureDate=2017","Color=White","Capacity=5","FuelRate=7","Model=AMG C 43 4-Matic","MCFV=30","FCFV=55"],["ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0"]]
```

```
input (convertible_inc, coupe_inc, crossover_inc, luxury_inc, sedan_inc,
sport_inc, suv_inc, truck_inc, wagon_inc, van_inc, data_list);
output (data_list_new);
action
(GetAutomobiles(convertible_inc, coupe_inc, crossover_inc, luxury_inc, sedan_inc,
sport_inc, suv_inc, truck_inc, wagon_inc, van_inc, data_list));
```

Figure 7. Fuzzy transition that makes the reasoning.



```
[data_list<>[[]]]
```

```
1`[["Things","Company","CarManufacture","Kia","Automobile","Coupe","ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60","Cost=47.2228561397"],["Things","Company","CarManufacture","Peugot","Automobile","Sedan","ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0","Cost=47.2228561397"]]
```

```
1`[["Things","Company","CarManufacture","Kia","Automobile","Coupe","ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60","Cost=47.2228561397"],["Things","Company","CarManufacture","Peugot","Automobile","Sedan","ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0","Cost=47.2228561397"]]
```

```
1`["Things","Company","CarManufacture","Kia","Automobile","Coupe","ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60","Cost=47.2228561397"]++
1`["Things","Company","CarManufacture","Peugot","Automobile","Sedan","ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0","Cost=47.2228561397"]
```

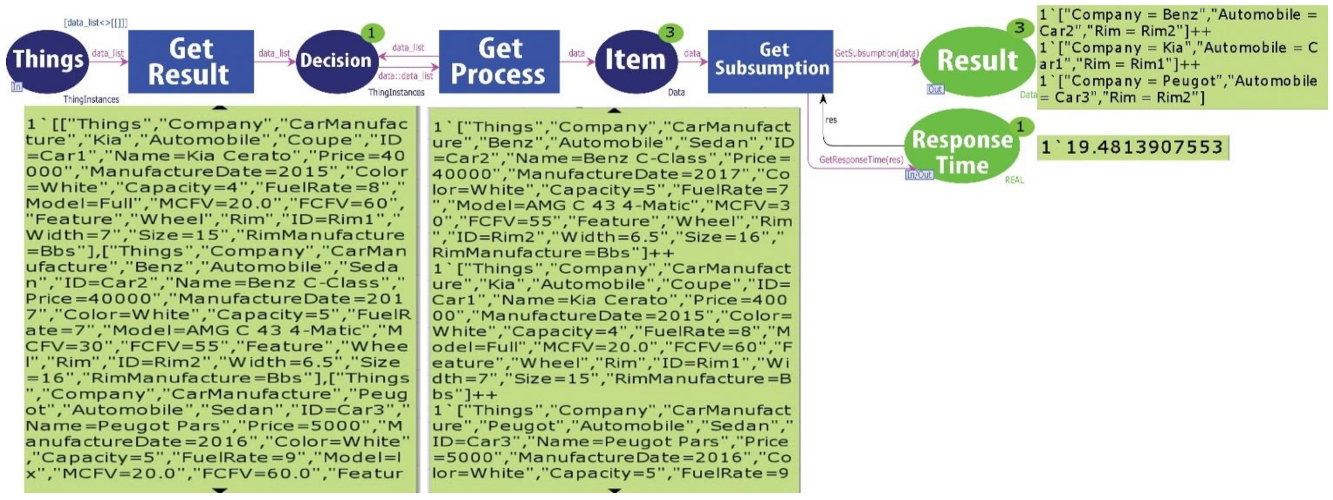
```
1`["Company = Kia","Automobile = Car1"]++
1`["Company = Peugot","Automobile = Car3"]
```

```
1`8.65635493928
```

Figure 8. Query and the result related to the first execution.

Five features are important in this comparison. The first column represents the satisfiability and whether the reasoner engines can perform it. The second column represents the reasoner process during the execution time, and models the reasoner.

This feature is an important point of this work, which is performed using the colored Petri net. The third column compares the fuzzy property of this work with other reasoner engines. The fourth column discusses whether the execution can be stopped at each step of the reasoner process. Finally, the fifth column considers the optimal solution of the realization problem and compares different inference engines. HermiT is a reasoner for ontologies written using the Web Ontology Language (OWL). Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify the subsumption relationships between classes, and more.



```
1`[["Things","Company","CarManufac
ture","Kia","Automobile","Coupe","ID
=Car1","Name=Kia Cerato","Price=40
000","ManufactureDate=2015","Color
=White","Capacity=4","FuelRate=8","
Model=Full","MCFV=20.0","FCFV=60",
"Feature","Wheel","Rim","ID=Rim1","
Width=7","Size=15","RimManufacture
=Bbs"],["Things","Company","CarMan
ufacture","Benz","Automobile","Seda
n","ID=Car2","Name=Benz C-Class","
Price=40000","ManufactureDate=201
7","Color=White","Capacity=5","FuelR
ate=7","Model=AMG C 43 4-Matic","M
CFV=30","FCFV=55","Feature","Whee
l","Rim","ID=Rim2","Width=6.5","Size
=16","RimManufacture=Bbs"],["Things
","Company","CarManufacture","Peug
ot","Automobile","Sedan","ID=Car3","
Name=Peugot Pars","Price=5000","M
anufactureDate=2016","Color=White"
,"Capacity=5","FuelRate=9","Model=l
x","MCFV=20.0","FCFV=60.0","Featur
```

```
1`["Things","Company","CarManufact
ure","Benz","Automobile","Sedan","ID
=Car2","Name=Benz C-Class","Price=
40000","ManufactureDate=2017","Co
lor=White","Capacity=5","FuelRate=7
","Model=AMG C 43 4-Matic","MCFV=3
0","FCFV=55","Feature","Wheel","Rim
","ID=Rim2","Width=6.5","Size=16","
RimManufacture=Bbs"]++
1`["Things","Company","CarManufact
ure","Kia","Automobile","Coupe","ID=
Car1","Name=Kia Cerato","Price=400
00","ManufactureDate=2015","Color=
White","Capacity=4","FuelRate=8","M
odel=Full","MCFV=20.0","FCFV=60","F
eature","Wheel","Rim","ID=Rim1","Wi
dth=7","Size=15","RimManufacture=B
bs"]++
1`["Things","Company","CarManufact
ure","Peugot","Automobile","Sedan","
ID=Car3","Name=Peugot Pars","Price
=5000","ManufactureDate=2016","Co
lor=White","Capacity=5","FuelRate=9
```

```
1`["Company = Benz","Automobile =
Car2","Rim = Rim2"]++
1`["Company = Kia","Automobile = C
ar1","Rim = Rim1"]++
1`["Company = Peugot","Automobile
= Car3","Rim = Rim2"]
```

```
1`19.4813907553
```

Figure 9. Query and results of the first, second, and third samples for the secend execution.

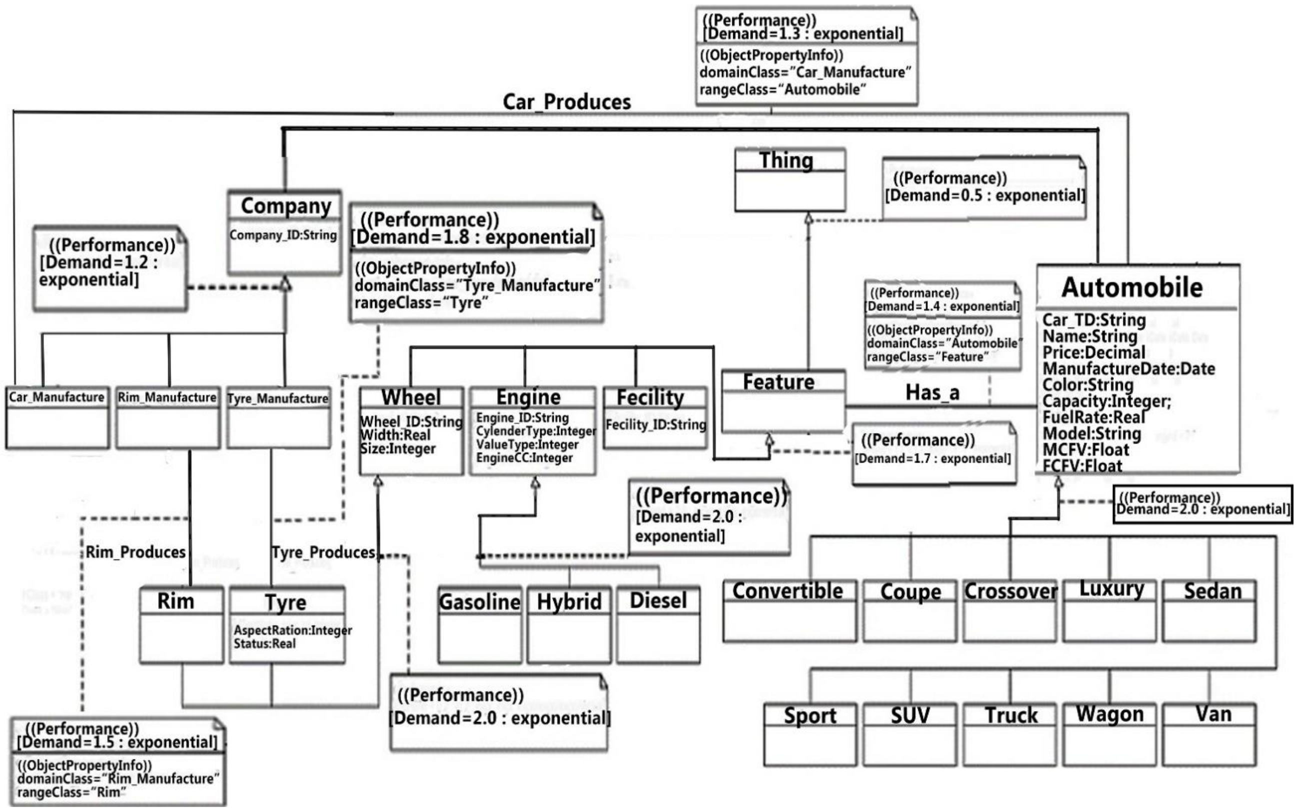


Figure 10. Description of the case study ontology using a class diagram together with the associated annotations.

HermiT uses direct semantics and passes all OWL 2 conformance tests for direct semantics reasoners [36]. FaCT++ is a tableaux-based reasoner for expressive Description Logics. It covers OWL and OWL 2 (lacking support for key constraints and some data types) DL-based ontology languages. Now it is used as one of the default reasoners in the Protege 4 OWL editor [37]. The proposed method is compared with HermiT and FaCT++ reasoners in terms of speed and accuracy, and the results are provided in Tables 2 and 3. The scores in the tables show the accuracy of the answers given by the reasoner. In other words, a score of 253 out of 264 means that 253 correct answers were given out of 264 questions. The error rate shows the incorrect answers given by the reasoner. The tables clearly show that the proposed method outperforms other methods.

The dataset is used in the framework created by the ontology Reasoner Evaluation Workshop in 2019 [38], which has a standard structure for evaluating ontology reasoning in order to evaluate the proposed method.

Table 2. Comparison of different methods based on the Realization parameter.

Rank	Reasoner	Score	Error	Time(s)
1	Proposed Reasoner	253/264	11	545.68 s
2	FaCT++	172/264	92	1111.3 s
3	HermiT	163/264	101	2934.9 s
4	HermiT-OA4	162/264	102	3022.5 s

Table 3. Comparison of different methods based on the classification parameter.

Rank	Reasoner	Score	Error	Time(s)
1	Proposed Reasoner	292/306	14	1318.18 s
2	HermiT-OA4	237/306	69	5808.2 s
3	HermiT	236/306	70	5416.4 s
4	FaCT++	200/306	106	1361.3 s

6. Conclusion

In this paper, a visual method for modeling and solving the realization problem based on the subsumption and satisfiability problems in ontology using fuzzy-colored Petri nets was proposed. The main issue in solving the realization problem is its computational complexity, which has been solved by introducing the proposed algorithm. For an easier implementation, an executive reasoning model was developed in different steps and a mapping algorithm for simpler transformations was presented. RDFS was used to better represent the ontology, and the UML class diagram was used in order to provide a standard description of the ontology. Thus, using the ontology information, an executive reasoning model of the ontology could be presented using the proposed method. This method tries to develop an executive reasoning model that can receive various queries in a simpler format and offer the optimal result and evaluate the results. Moreover, this model could respond to the query that was not previously defined at any point, where each token was fuzzy using the colored Petri net.

Table 4. List of reasoners with their supported services.

Reasoner	Institution	Details	Satisfiability	Displays the reasoning process at runtime	Fuzzzy	Stop the inference process at each stage of execution	Realization
BUNDLE [13]	University of Ferrara	Probabilistic reasoner based on Pellet	Yes	No	No	No	No
CEL [14]	Technische Universität Dresden	Lisp-based reasoner	Yes	No	No	No	No
DBOWL [15]	University of Malaga	scalable reasoner for OWL ontologies with very large Aboxes	Yes	No	No	No	No
DeLorean [16]	Not given	Fuzzy rough Description Logic reasoner	Yes	No	Yes	No	Yes
DistEL [17]	Wright State University	Distributed reasoner that runs on a cluster of machines	No	No	No	No	Yes
DRAGON [18]	University of Paris 8, IUT of Montreuil	OWL reasoner that supports distributed reasoning over a networked ontologies	No	No	No	No	No
ElepHant [19]	Not given	Consequence-based reasoner that currently supports part of the OWL 2 EL fragment for the reasoning tasks classification, consistency and realization.	No	No	No	No	Yes
FuzzyDL [20]	ISTI–CNR	Free Java/C++ based reasoner for fuzzy SHIF with concrete fuzzy concepts	Yes	No	Yes	No	No
F-CPN Tools	Semnan University	Fuzzy-Colorcolored Petri Netsnets-based reasoner	Yes	Yes	Yes	Yes	Yes

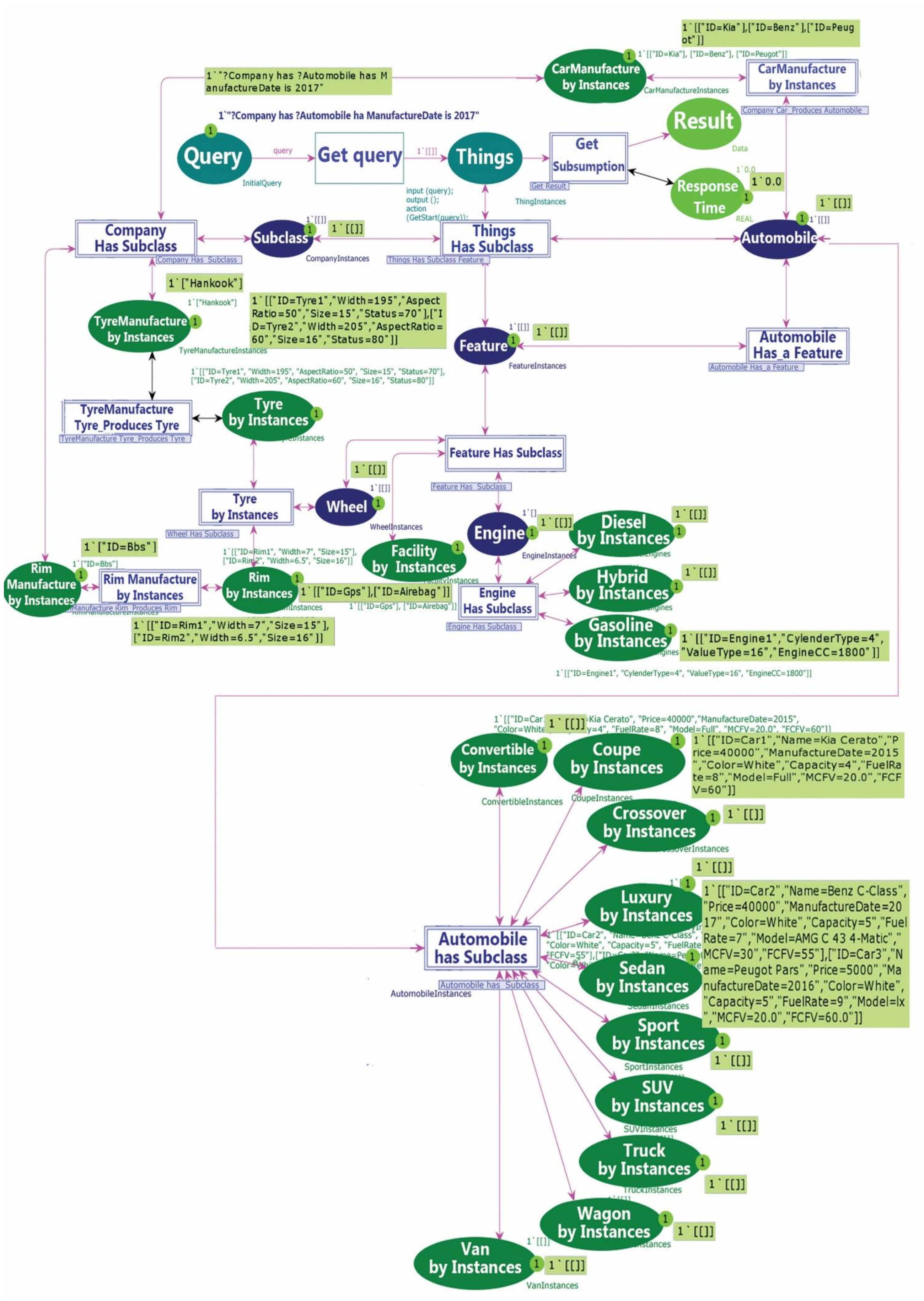

1`[["ID=Kia"],["ID=Benz"],["ID=Peugot"]]
CarManufacture by Instances
CarManufacture by Instances
CarManufactureInstances
Company Car_Produces Automobile
1`"?Company has ?Automobile has ManufactureDate is 2017"
1`"?Company has ?Automobile ha ManufactureDate is 2017"
Query
InitialQuery
query
Get query
1`[[]]
Things
Get Subsumption
Get Result
ThingInstances
input (query); output (); action (GetStart(query));
Result
Data
Response Time
1`0.0
REAL
Automobile
1`[[]]
Company Has Subclass
Company Has_Subclass
Subclass
1`[[]]
CompanyInstances
Things Has Subclass
Things Has Subclass Feature
1`["Hankook"]
TyreManufacture by Instances
TyreManufactureInstances
1`[["ID=Tyre1","Width=195","AspectRatio=50","Size=15","Status=70"],["ID=Tyre2","Width=205","AspectRatio=60","Size=16","Status=80"]]
Feature
1`[[]]
FeatureInstances
Automobile Has_a Feature
Automobile Has_a Feature
TyreManufacture Tyre_Produces Tyre
TyreManufacture Tyre_Produces Tyre
Tyre by Instances
Feature Has Subclass
Feature Has_Subclass
1`[[]]
Tyre by Instances
Wheel Has Subclass
Wheel
WheelInstances
Engine
1`[[]]
EngineInstances
Diesel by Instances
1`[[]]
1`["ID=Bbs"]
Rim Manufacture by Instances
Rim Manufacture by Instances
Manufacture Rim_Produces Rim
1`[["ID=Rim1","Width=7","Size=15"],["ID=Rim2","Width=6.5","Size=16"]]
Rim by Instances
Facility by Instances
1`["ID=Gps"],["ID=Airebag"]]
Engine Has Subclass
Engine Has Subclass
Hybrid by Instances
1`[[]]
Gasoline by Instances
1`[["ID=Engine1","CylenderType=4","ValueType=16","EngineCC=1800"]]
Convertible by Instances
ConvertibleInstances
1`[[]]
Coupe by Instances
CoupeInstances
1`[["ID=Car1","Name=Kia Cerato","Price=40000","ManufactureDate=2015","Color=White","Capacity=4","FuelRate=8","Model=Full","MCFV=20.0","FCFV=60"]]
Crossover by Instances
CrossoverInstances
1`[[]]
Luxury by Instances
1`[[]]
1`[["ID=Car2","Name=Benz C-Class","Price=40000","ManufactureDate=2017","Color=White","Capacity=5","FuelRate=7","Model=AMG C 43 4-Matic","MCFV=30","FCFV=55"],["ID=Car3","Name=Peugot Pars","Price=5000","ManufactureDate=2016","Color=White","Capacity=5","FuelRate=9","Model=lx","MCFV=20.0","FCFV=60.0"]]
Automobile has Subclass
Automobile has_Subclass
AutomobileInstances
Sedan by Instances
SedanInstances
Sport by Instances
SportInstances
1`[[]]
SUV by Instances
SUVInstances
1`[[]]
Truck by Instances
TruckInstances
1`[[]]
Wagon by Instances
1`[[]]
Van by Instances
VanInstances
1`[[]]

Figure 11. Executive reasoning model based on CPNs from the case study ontology.

References

[1] P. Křemen and Z. Kouba, "*Ontology-driven information system design*," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 42, pp. 334-344, 2012.

[2] T. Murata, "*Petri nets: Properties, analysis and applications*," Proceedings of the IEEE, Vol. 77, pp. 541-580, 1989.

[3] J. Yim, J. Joo, and G. Lee, "*Petri net-based ontology analysis method for indoor location-based service system*," International Journal of Advanced Science and Technology, Vol. 39, pp. 75-92, 2012.

[4] L. Zhu and W. Wang, "*UML diagrams to hierarchical colored petri nets: an automatic software performance tool*," Procedia Engineering, Vol. 29, pp. 2687-2692, 2012.

[5] Z. Xiao and Z. Ming, "*A method of workflow scheduling based on colored Petri nets*," Data and Knowledge Engineering, Vol. 70, pp. 230-247, 2011.

[6] A. Agostini, C. ,Bettini, and D. Riboni, "*Online ontological reasoning for context-aware internet services*". In 2nd International Workshop on Contexts and Ontologies: Theory, Practice and Applications, Collocated with the 17th European Conference on Artificial Intelligence, ECAI 2006 (Vol. 210).

[7] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "*SWRL: A semantic web rule language combining OWL and RuleML*," W3C Member submission, Vol. 21, pp. 1-31, 2004.

[8] M. Rahimi and M. Zahedi, "*Query expansion based on relevance feedback and latent semantic analysis*," Journal of AI and Data Mining, Vol. 2, pp. 79-84, 2014.

[9] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, *et al.*, "*A survey of context modelling and reasoning techniques*," Pervasive and Mobile Computing, Vol. 6, pp. 161-180, 2010.

[10] A. Maedche and S. Staab, "*Ontology learning for the semantic web*," IEEE Intelligent systems, Vol. 16, pp. 72-79, 2001.

[11] A. Mousavi, A. Sheikh Mohammad Zadeh, M. Akbari, and A. Hunter, "*A New Ontology-Based Approach for Human Activity Recognition from GPS Data*," Journal of AI and Data Mining, Vol. 5, pp. 197-210, 2017.

[12] I. Horrocks, and S. Ulrike ,"*Ontology reasoning in the SHOQ (D) description logic*." IJCAI. Vol. 1. No. 3. 2001.

[13] F. Riguzzi, E. Bellodi, E. Lamma, and R. Zese, "*BUNDLE: A reasoner for probabilistic ontologies*," in International Conference on Web Reasoning and Rule Systems, pp. 183-197, 2013.

[14] F. Baader, C. Lutz, and B. Suntisrivaraporn, "*CEL—a polynomial-time reasoner for life science ontologies*," in International Joint Conference on Automated Reasoning, pp. 287-291, 2006.

[15] M. del Mar Roldan-Garcia, and J. F. Aldana-Montes, "*DBOWL: Towards a Scalable and Persistent OWL reasoner*," in Third International Conference on Internet and Web Applications and Services, 2008, pp. 174-179, 2008.

[16] F. Bobillo, M. Delgado, and J. Gómez-Romero, "*DeLorean: A reasoner for fuzzy OWL 2*". Expert Systems with Applications, 39(1), 258-272, (2012).

[17] R. Mutharaju, P. Hitzler, P. Mateti, and F. Lécué, "*Distributed and scalable OWL EL reasoning*". In European Semantic Web Conference (pp. 88-103). Springer, 2015.

[18] J. Baker, "*The DRAGON system--An overview*," IEEE Transactions on Acoustics, speech, and signal Processing, Vol. 23, pp. 24-29, 1975.

[19] B. Sertkaya, "*The ELepHant Reasoner System Description*", OWL Reasoner Evaluation (ORE) workshop 2013.

[20] F. Bobillo and U. Straccia, "*fuzzyDL: An expressive fuzzy description logic reasoner*," in International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), pp. 923-930, 2008.

[21] E. Inelmen, and A. Ibrahim, "*A new approach to teaching fuzzy logic system design*". In International Fuzzy Systems Association World Congress (pp. 79-86). Springer, Berlin, Heidelberg , 2003.

[22] K. Tanaka, "*An introduction to fuzzy logic for practical applications*", Springer, New York ,1997.

[23] G. Klir and B. Yuan, "*Fuzzy sets and fuzzy logic*" vol. 4: Prentice Hall New Jersey, 1995.

[24] H.-J. Zimmermann, "*Fuzzy set theory—and its applications*" Springer Science and Business Media, 2011.

[25] C. Petri, "*Kommunikation mit Automaten*", Bonn: Institut fur Instrumentelle Mathematik, Schriften des IIM Nr. 3, also, English translation," Communication with Automata," Tech. Rep. RADC-TR-65-377, 1966.

[26] W. Reisig and P. Nets, "An Introduction *EATCS*" Monographs on Theoretical Computer Science, Vol. 4, 1985.

[27] K. Jensen, "*Book Review: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*" (volume 1) by Kurt Jensen: SIGOPS Oper. Syst. Rev., Vol. 28, pp. 1-2, 1994.

[28] K. Jensen, "*Coloured Petri nets: basic concepts, analysis methods and practical*" Vol. 1: Springer Science and Business Media, 2013.

[29] J. Kurt, "*Coloured Petri nets: Basic concepts, analysis methods and practical use*," EATCS Monographs on Theoretical Computer Science. 2nd edition, Berlin: Springer-Verlag, 1997.

[30] K. Jensen and L. M. Kristensen, "*Coloured Petri nets: modelling and validation of concurrent systems*" Springer Science and Business Media, 2009.

[31] G. Booch, "*The unified modeling language user guide*" Pearson Education India, 2005.

[32] F. Aquilani, S. Balsamo, and P. Inverardi, "*Performance analysis at the software architectural design level*," Performance Evaluation, Vol. 45, pp. 147-178, 2001.

[33] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi,"*The description logic handbook: Theory, implementation and applications*" Cambridge University Press, 2003.

[34] K. Etminani, A. R. Delui, and M. Naghibzadeh, "*Overlapped ontology partitioning based on semantic similarity measures*," in 5th International Symposium on Telecommunications, pp. 1013-1018. 2010.

[35] P. Pothipruk and G. Governatori, "*A formal ontology reasoning with individual optimization: a realization of the semantic web*," in International Conference on Web Information Systems Engineering, , pp. 119-132, 2005.

[36] B. Glimm, I. Horrocks, B. Motik, and G. Stoilos, "*HermiT: reasoning with large ontologies*, *Computing Laboratory*", Oxford University, (2009).

[37] D. Tsarkov, I. Horrocks, "*FaCT++ description logic reasoner: System description, in: International Joint Conference on Automated Reasoning*", pp. 292-297. (Springer, 2006).

[38] N. Pour, A. Algergawy, R. Amini, D. Faria, I. Fundulaki, I. Harrow, ... and L. Zhou, "*Results of the ontology alignment evaluation initiative*". In Proceedings of the 15th International Workshop on Ontology Matching , 2019.

ارائه مدل اجرایی برای بهبود استنتاج و حل مساله ادراک و فهم در آنتولوژی با استفاده از شبکه‌های پتری رنگی فازی (FCPN)

مجتبی شکوهی نیا، عباس دیدبان* و فرزین یغمایی

دانشکده مهندسی برق و کامپیوتر، دانشگاه سمنان، سمنان، ایران.

ارسال ۲۰۲۱/۰۴/۱۲ ؛ بازنگری ۲۰۲۱/۰۶/۱۱ ؛ پذیرش ۲۰۲۱/۱۱/۰۱

چکیده:

امروزه، روش‌های زیادی برای ارائه دانش، استنتاج و استخراج نتایج از میان دانش مربوطه ارائه شده است. با وجود موفقیت آنتولوژی در ارائه دانش، اما نحوه استنتاج آن در این روش با چالش‌هایی روبرو است. مهم ترین چالشی که در استنتاج در روش‌های مبتنی بر آنتولوژی وجود دارد، بهبود حل مساله ادراک و فهم (Realization) در فرآیند استنتاج است. از طرفی، وجود عدم قطعیت در این مشخصه‌ها یک واقعیت انکار ناپذیر بوده و مستلزم چارچوبی می باشد که بتواند مشخصه‌های عدم قطعیت را در سطح داده‌ها بیان کند. در این مقاله هدف مدلسازی و بهبود استنتاج و حل مساله ادراک و فهم در آنتولوژی با استفاده از شبکه‌های پتری رنگی فازی می‌باشد. برای رسیدن به این هدف، ابتدا الگوریتمی جهت حل بهینه مساله ادراک و فهم ارائه داده شده است. سپس مفاهیم فازی در شبکه‌های پتری رنگی معرفی و ارائه شده است. در مرحله بعد الگوریتمی ارائه شده که با استفاده از آن، توصیف آنتولوژی مبتنی بر نمودار کلاس UML به یک مدل اجرایی مبتنی بر شبکه‌های پتری رنگی فازی تبدیل می‌شود. توسط این راه کار، یک روش برای تشکیل مدل اجرایی و استنتاج مبتنی بر شبکه‌های پتری رنگی فازی از آنتولوژی ایجاد می‌شود که می‌توان با اعمال پرس و جوهای مختلف به نتایج موردنظر دست یافت. در انتها نیز کارایی روش ارائه شده از نظر عملکرد مورد ارزیابی قرار گرفته است که نتایج نشان دهنده عملکرد بهینه روش ارائه شده و ارزیابی کارایی از جنبه های مختلف می‌باشد.

کلمات کلیدی: استنتاج، آنتولوژی، زبان مدل‌سازی یکپارچه، شبکه‌های پتری رنگی فازی.