Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **4**, November **2022**

IoT Service Description and Composition Method with Knowledge Graph

Zheng-Yi Tang

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
Fujian Provincial Key Laboratory of Information Processing and Intelligent Control
Minjiang University
Xiyuangong Road, Fuzhou City, Fujian Province, 350121, China
Key Laboratory of Hunan Province for Mobile Business Intelligence
Hunan University of Technology and Business
Yuelu Road, Changsha City, Hunan Province, 410205, China
tangzy84@126.com

Chuan Liu

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
liuchuan98@outlook.com

Zhi-Zhu Lian

Fujian Provincial Key Laboratory of Information Processing and Intelligent Control
Minjiang University
Xiyuangong Road, Fuzhou City, Fujian Province, 350121, China
lzz600@126.com

Jin-Shui Wang*

School of Computer Science and Mathematics
Fujian University of Technology
Xuefu South Road, Fuzhou City, Fujian Province, 350118, China
Key Laboratory of Hunan Province for Mobile Business Intelligence
Hunan University of Technology and Business
Yuelu Road, Changsha City, Hunan Province, 410205, China
wangjinshui@fjut.edu.cn

Md. Alamgir Hossain

Department of Management
Hajee Mohammad Danesh Science and Technology University
Dinajpur – 5200, Bangladesh
shamimru@gmail.com

*Corresponding author: Jin-Shui Wang

Received June 25, 2022, revised August 7, 2022, accepted October 8, 2022.

Abstract. *Service computing techniques have achieved remarkable results in solving the cohesiveness problem of Web applications and are therefore introduced into the IoT domain to break the trend of inlining IoT systems. However, compared with the traditional Internet, IoT has richer elements and semantic connotations, so the existing service description framework needs to be improved and extended. In this paper, we propose an ontology description framework for IoT services, design corresponding inference rules, and give a storage method for IoT services by integrating the knowledge graph technology for the strongly associated characteristics of IoT services. On this basis, an algorithm for combining IoT services based on graph planning is proposed. The experimental results show that the proposed method in this paper can provide richer entity semantics and effectively improve the efficiency and success rate of IoT service composition*
Keywords: IoT services, ontology, knowledge graph, Graph Plan, service composition

1. **Introduction.** With the development of the Internet of Things (IoT) industry, all aspects of society are becoming more and more intelligent, which is reflected in the fact that many common devices are embedding the ability to sense, execute, compute, and interact, thus enabling the management and organization of these common devices [1-4]. A large number of smart devices will inevitably generate a large amount of data, and for the security of these massive data scholars have introduced technologies such as cloud, fog computing and blockchain into the IoT [5-7]. However, for data sharing, the differences between devices and platforms make a large amount of data is divided independently in different IoT systems, which are closed and tightly coupled so that a large amount of data cannot be shared and integrated into other systems [8].

Service Oriented Computing (SOC) can effectively solve the problem of data reuse and sharing by unifying applications of heterogeneous platforms into services. In SOC, services are defined as self-describing, adaptive, and platform-independent autonomous computing units, and can be used to rapidly build distributed software systems and enterprise applications through service composition techniques. Services are published and interacted with a consistent standard, thus effectively solving the problem of interoperability of heterogeneous objects.

The success of SOC in the traditional Internet domain provides a path for the development of the Internet of Things (IoT), and researchers have applied the ideas and methods of SOC to the design of IoT systems, resulting in the concept of IoT services [9]: the core idea is to unify the functions provided by various objects (devices, applications, etc.) in the IoT and publish them as services to the outside world, which can be mutually call, carry out information transfer, and can be dynamically discovered and composed. From the perspective of services, the heterogeneity of IoT is stronger than that of the Internet, making the operation of IoT services more problematic; at the same time, IoT services interact with the physical environment, and their behavior and results are more difficult to control [10,11].

Therefore, the aim of the work in this paper is to construct an accurate IoT service model to improve the design quality and guarantee the reliable operation of the IoT application system; and to design a reasonable IoT service composition method to meet the complex requirements of users.

The main contributions of the work in this paper are: (1) for the characteristics and difficulties of IoT services, a set of semantic description framework of IoT services is proposed, semantic models of IoT services and entities are constructed, and relevant inference rules for IoT services are designed. (2) Discussed how to store IoT services under the knowledge graph using Neo4j tool, and on this basis, designed a graph planning based IoT service composition algorithm to improve the algorithm execution efficiency.

The rest of this paper is organized as follows. Section 2 introduces the related work from two aspects, IoT service modeling and architecture, and IoT service composition method; Section 3 gives the modeling method of IoT services and entities for the characteristics of IoT services; Section 4 gives the design storage method of IoT service knowledge graph; Section 5 gives the IoT service composition algorithm based on graph planning; Section 6 verifies the effectiveness of this paper through practical scenarios and simulation experiments Section 6 verifies the effectiveness of the method through practical scenarios and simulation experiments; Section 7 concludes the whole paper and discusses the subsequent research directions.

2. **Related Work.** A significant obstacle to the development of IoT is the existence of a large amount of heterogeneous data among various sensors and actuators. To efficiently parse these data and combine them into intelligent services according to certain rules, a set of standard description structures is needed to describe them, and the relevant data can be published in a machine-readable format. At present, there has been a lot of research on IoT services at home and abroad, but the semantic description of IoT services is still in the research stage, and a set of the unified standard has not yet been formed.

In terms of IoT service modeling and architecture framework, Agarwal et al. [12] proposed a FIESTA-IoT streamlined ontology model, which integrates ontologies including classical SSN, IoT-A, and IoT-Lite, and offers the M3-LITE classification system. This work starts from semantic sensor networks and focuses more on the modeling of sensor devices, and the portrayal of IoT services is still inadequate; De et al. [13] proposed a ternary model of device-entity-services, which is an earlier definition of IoT service models . This work defines the IoT service model as "real-world services" and describes the functions of physical devices as ontologies, but does not design further manipulation schemes for semantic data, such as querying and reasoning. Wei [14] elaborated on the description, discovery, and composition of IoT services as early as the nascent stage, illustrated the computational model of IoT services, and clarified the challenges that IoT services will face in the future. This work improves on the Web service ontology WSMO-Lite for IoT characteristics, but lacks a detailed portrayal of the physical environment to reflect the interaction between the service and the complex environment; Yu et al. [15] proposed a set of IoT service evaluation systems to test the effectiveness of IoT services for the management mechanism of the service to meet the needs of users; however, this work involves the same technology based on sensing networks and Web services, which cannot be directly applied in the IoT domain.

In terms of service composition, Hamzei and Navimipour [16] discussed the advantages and disadvantages of existing service composition techniques in the process of providing IoT services to users under the service-oriented idea. Androec [17] designed a mechanism for semi-automatic service composition after semantic annotation of ontologies using JSON-LD. However, this approach models IoT services as simple inputs and outputs in the same way as Web services and does not represent the complex nature of IoT services well. Baker et al. [18] designed an IoT service composition algorithm to search for the minimum number of services and proposed a transformation model for user requirements. This work performs service composition from the perspective of saving energy for IoT systems but again does not distinguish IoT services from traditional Web services in the composition process.

In summary, the existing related research results can bring interoperability to heterogeneous data between IoT platforms, but there are still the following problems: (1) few IoT ontologies constructed with the concept of services are seen, and most of the mature research results are based on sensing networks. (2) Most of the ontologies centered on IoT

services lack the consideration of the environment, on which IoT services are dependent. (3) Most of the composed approaches of IoT services are based on Web services, which cannot portray the specificity of IoT services.

3. Semantic Description Framework for IoT Services.

3.1. IoT entity.

Firstly, the characteristics of entities in an IoT system are elaborated. The biggest difference between IoT services and traditional Web services lies in the degree of dependence on the environment; IoT services are often bundled with physical devices that can access things in the physical world to the information world. Therefore, it is necessary to classify entities as physical entities and information entities. Physical entities represent entities that exist in the physical world and information entities represent entities that exist in the information world. For example, for the same smart car, the self-test service of the car and the traffic service of the city focus on modeling them as physical entities and information entities, respectively. Moreover, IoT entities are also highly dynamic, for example, if a service and an entity are not in the same physical location, then the two should not be able to influence each other. In this paper, in order to describe this nature of entities, the concept of entity properties is abstracted, and the difference in the state of property can indicate whether an entity property is available or whether it is imposed by a service.

In this paper, IoT entities are defined as:

Definition 3.1. *(IoT entity). An IoT entity IoTET = (EType, EId, Property), where*

•$IoTET \in \{PE, IE\}$ *.PE is a Physical Entity , IE is an Information Entity .*

•*Eid = (ECategory, EUri). ECategory is the category identifier of the environment entity, EUri is the environment entity identifier.*

•*Property = (PUri, Value, State). State = (ust, ast),* $ust \in \{available, unavailable\}$, $ast \in \{null, affected, unaffected\}$*.PUri is the unique identifier of the property, Value is the property value of the property, and State is the state the property is in, where ust is the available state of the property, indicating whether the property is available when the IoT service is executed, and ast is the affected state of the property, indicating whether the property has been applied with effect by the IoT service. The affected state is only meaningful for the properties of physical entities, and the affected state of information entities always takes null*

Physical and information entities cannot exist in isolation; when adding a new physical (information) entity, its corresponding information (physical) entity should also be added.

For example, EnvironmentLight is an ambient light entity with a light intensity property, initialized as described below:

Example 3.1. *EnvironmentLight:*

EType = PE
Eid:
ECategory = P-Environment_Light
EUri = P-Light_01
Property:
PUri = P-Brightness
Value = null
State:
ust = unavailable
ast = unaffected

EType = IE
Eid:
ECategory =I-Environment_Light
EUri = I-Light_01
Property:
PUri = I-Brightness
Value = null
State:
ust = unavailable
ast = null

3.2. **IoT service.** An IoT service is a service that interacts with an entity according to the user's wishes, and its behavior is either sensing and acquiring the parameters of the physical environment, or it artificially adjusts the parameters of the physical environment by applying some effect to the entity after internal data processing. According to these behavioral characteristics of IoT services, four service behavioral elements can be summarized: sense, effect, input and output.

In this paper, IoT services are defined as:

Definition 3.2. *(IoT service)An IoT service Srv = (SId,SProfile),where*

•*SId = (SUri, SDescription), SUri is the unique identifier of the service and SDescription is the text describing the functionality of the service.*

•*SProfile = (input, output, sense, effect),The SProfile quadruplet corresponds to the input, output, sense, and effect behaviors of the IoT service, respectively. And there are also the following restrictions: the elements that interact with the service in sense and effect are the physical entity properties in definition 3.1, and the elements that interact with the service in input and output are the information entity propertiess in definition 3.1.*

For example, Brightsense is an ambient light intensity awareness service that provides information on ambient light intensity, described as follows:

Example 3.2. *Brightsense:*
SId :
SUri = Bright_sense
SDescription = "sense the light intensity then output value."
SProfile:
input = ∅
output = I-Brightness
sense = P-Brightness
effect = ∅

IoT services can be divided into three categories according to their functions: sensing services, computing services, and effecting services. The sensing service, once started, continuously acquires the property values of physical environment entities and transforms them into information entity properties that can be understood by other services, the computing service only processes the information entity properties, and the effecting service, after receiving the information entity properties, applies effects to physical entities and changes the property values of physical entities. Combining the characteristics of the above services with the Sprofile section in Definition 3.2, the IoT services can be further classified as follows:

Sensing service: $Srv.Profile = (sense, \emptyset, \emptyset, output)$
Computing services: $Srv.Profile = (\emptyset, input, \emptyset, output)$

Effecting services: $Srv.Profile = (\emptyset, input, effect, \emptyset)$

In addition, to reflect the dynamic nature of IoT services, this paper defines service enablement and service execution as follows:

Definition 3.3. *(Service enablement)*

For an IoT service Srv, Srv.Profile = (input, output, sense, effect), Srv is said to be enabled if it satisfies $\forall p \in input : p.ust = available$*, denoted as enable(Srv).*

Definition 3.4. *(Service execution)*

The effect of executing an IoT service Srv is: $\forall p \in output : p.ust = available$ *and* $\forall p \in sense \cup effect : p.ast = affected$*, denoted as execute(Srv).*

3.3. **IoT service ontology.**

3.3.1. *Ontology construction.* According to Nacer and Aissani [19], the ontology aims to construct a model to indicate the precise definition of knowledge concepts between specific domains and the associations between them and to provide a formal description of the model. Based on the relevant definitions above, a graphical representation of the IoT service description framework is shown in Figure 1.

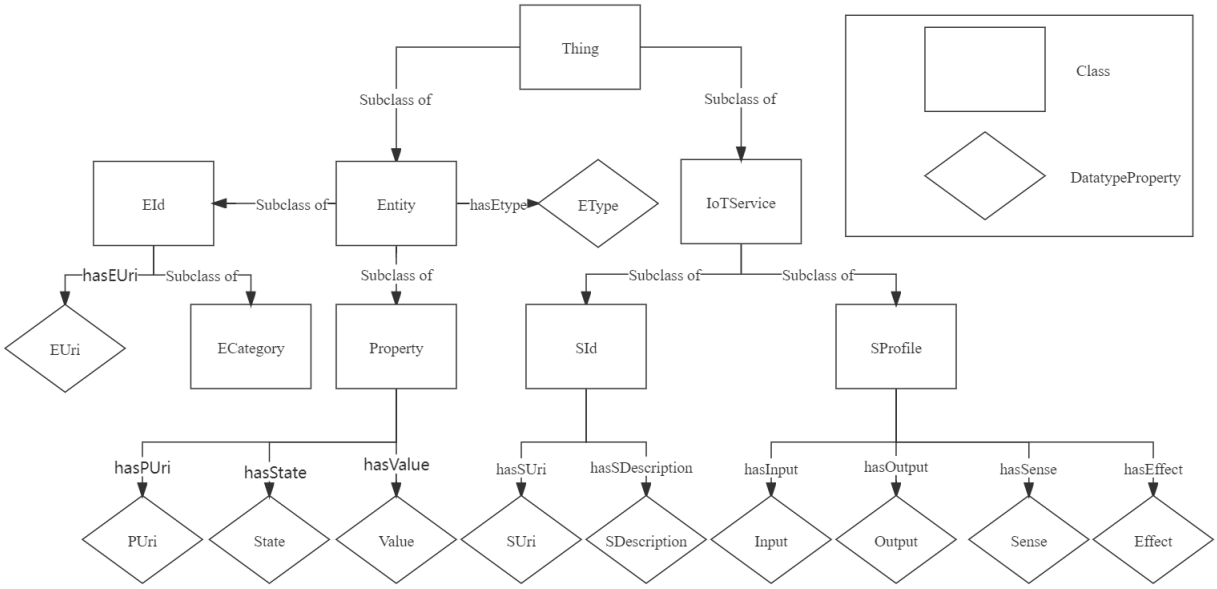

FIGURE 1. IoT Service Description Framework

In this paper, when constructing the IoT service ontology, the five basic principles proposed by Gruber [20] are observed: clarity, completeness, consistency, and extensibility, and the above IoT service description framework is modeled using the Protégé ontology construction tool, and the service and entity hierarchy is shown in Figure 2, and some of the core object properties are shown in Figure 3.

Table 1 is a description of some of the core properties of Figure 1 on the ontology of the IoT service domain.

3.3.2. *Rule-based inference.* To enhance the semantic information of the ontology, often the relevant SWRL rules will be defined when building the ontology [21]. The Schema of SWRL consists of four parts, Imp, Atom, Variable and Building, where the main restriction expressions of Atom are: 1. $C(x)$: C is the OWL description. 2. $P(x, y)$: P

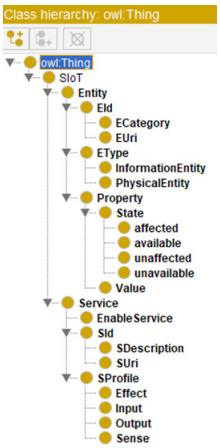

FIGURE 2. IoT Services and Entity Hierarchy

FIGURE 3. Some core object properties

TABLE 1. Description of some core properties of IoT services

Property	**Meanings**	**Domain**	**Range**
hsaEType	There are two types of each IoT entity	IoT entity	Entity type
hasProperty	Properties of each IoT entity	IoT entity	Entity property
hasEUri	Each IoT entity has a unique Uri	IoT entity	An entity URI
hasSUri	Each IoT service has a unique Uri	IoT service	An service URI
hasSDescription	Each IoT service has a functional description	IoT service	Text
hasInput	Each IoT service has input elements	IoT service	Information entity properties
hasOutput	Each IoT service has output elements	IoT service	Information entity properties
hasSenset	Each IoT service has sense elements	IoT service	Physical entity properties
hasEffectt	Each IoT service has effect elements	IoT service	Physical entity properties

is the attribute of OWL, x, y are variables, OWL individuals or OWL data value. This paper is written from the following two aspects:

(1) from the user, in service selection, the user may need to know whether the service has alternative services, and may care whether there is an association relationship between the services, and the relevant inference rules are defined as follows:

Rule1: $Service(?x) \wedge Service(?y) \wedge Property(?z) \wedge hasSense(?z, ?x) \wedge hasSense(?z, ?y)$ $\rightarrow$ $replaceable(?x, ?y)$

Rule2: $replaceable(?x, ?y) \rightarrow replaceable(?y, ?x)$

Explanation: Rule1, if both service x and service y perceive the same property z, then service x can be substituted by service y. Rule2, if service x can be substituted by service y, then service y can also be substituted by service x.

Rule3: $Service(?x) \wedge Service(?y) \wedge Property(?z) \wedge hasSense(?x, ?z) \wedge hasEffect(?y, ?z) \rightarrow related(?x, ?y)$

Rule4: $related(?x, ?y) \rightarrow related(?y, ?x)$

Explanation: Rule3, if service x perceives property z and service y applies effect to entity z, then service x is associated with service y. Rule4, if service x is associated with service y, then service y is also associated with service x.

(2) Starting from the aspect of ontology knowledge base, the definitions in the semantic framework of IoT services are mapped to the knowledge in the ontology knowledge base by writing rules and avoiding information omission when building the ontology manually, and the relevant inference rules are defined as follows:

Rule5: $Service(?x) \wedge Property(?y) \wedge hasinput(?x, ?y) \wedge hasState(?y, \text{“}available\text{”}) \rightarrow$ $enableservice(?x)$

Explanation: Rule5, service enablement corresponding to definition 3.3

Rule6: $Entity(?x) \wedge Property(?y) \wedge hasProperty(?x, ?y) \wedge hasState(?y, \text{“}unaffected\text{”}) \rightarrow PhysicalEntity(?x)$

Rule7: $Entity(?x) \wedge Property(?y) \wedge hasProperty(?x, ?y) \wedge hasState(?y, \text{“}affected\text{”}) \rightarrow$ $PhysicalEntity(?x)$

Explanation: Rule6, 7, Corresponding to the constraints in Definition 3.1 for physical entity properties.

Figure 4 shows the writing of the above custom rules done in Protégé's SWRL Tab operator panel.

	Name	Rule
☑	rule1	Service(?x) ^ Service(?y) ^ Property(?z) ^ hasSense(?z, ?x) ^ hasSense(?z, ?y) -> replaceable(?x, ?y)
☑	rule2	replaceable(?x, ?y) -> replaceable(?y, ?x)
☑	rule3	Service(?x) ^ Service(?y) ^ Property(?z) ^ hasSense(?x, ?z) ^ hasEffect(?y, ?z) -> related(?x, ?y)
☑	rule4	related(?x, ?y) -> related(?y, ?x)
☑	rule5	Service(?x) ^ hasState(?y, "available") ^ hasState(?z, "available") ^ hasInput(?x, ?y) ^ Property(?y) ^ hasSen...
☑	rule6	Entity(?x) ^ Property(?y) ^ hasProperty(?x, ?y) ^ hasState(?y, "unaffected") -> PhysicalEntity(?x)
☑	rule7	Entity(?x) ^ Property(?y) ^ hasProperty(?x, ?y) ^ hasState(?y, "affected") -> PhysicalEntity(?x)

FIGURE 4. SWRL rules

4. **IoT Service Storage with Knowledge Graph.**

4.1. **IoT Service Knowledge Graph Definition.** In Section 3, this paper gives a framework for the description of IoT services and constructs an ontology library with related rules under IoT services using Protégé. This chapter will explore the way of storing IoT services under a knowledge graph.

A knowledge graph is a semantic network that reveals the relationships between entities, and its generic representation is a triple $G = (E, R, S)$, where $E = \{e_1, e_2, ..., e_{|n|}\}$ is the set of entities in the knowledge base, which contains $|E|$ different entities; $R = \{r_1, r_2, ..., r_{|R|}\}$, which contains $|R|$ different relations, $S \subseteq E \times R \times E$ [22].

Mapping IoT services to traditional knowledge graphs lead to the definition of IoT service knowledge graphs:

Definition 4.1. *(IoT Service Knowledge Graph). An IoT service knowledge graph* $G = (S, P, R)$*, where*

•$S = \{s_1, s_2, ..., s_{|n|}\}$ *is the set of IoT services and the total number of services is n.*

•$P = \{p_1, p_2, ..., p_{|m|}\}$ *is the set of IoT entity properties and the total number of services is m.*

•$R = (input, output, sense, effect)$ *is the relationship between IoT service and entity properties.*

The above definition shows that IoT services under the knowledge graph no longer merely describe the service-to-service relationship, but indirectly link the services through the properties of IoT entities.

4.2. **Architecture of IoT Service Knowledge Graph Architecture.** There are two general-purpose knowledge graph construction methods: bottom-up and top-down approaches [23]. The former is to extract resources from publicly available datasets with the help of certain technical means and add them to the knowledge base after manual review. While the latter is to pre-define the ontology structure and then guide the data to be added to the knowledge base through the ontology structure. For IoT services, since there is relatively little relevant knowledge publicly available on the network, this paper adopts a top-down approach to construct the knowledge graph of IoT services. The detailed construction architecture is shown in Figure 5.

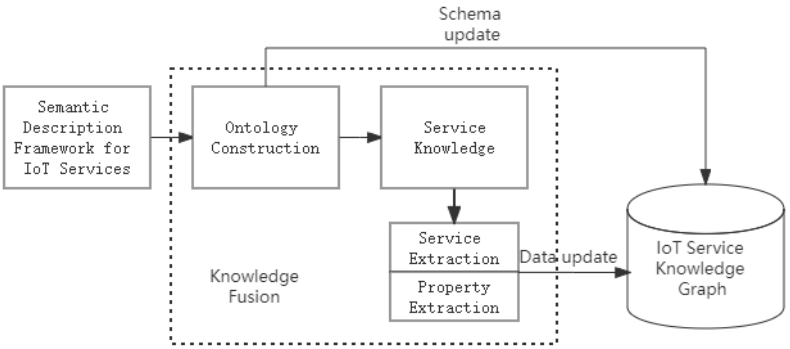

FIGURE 5. IoT ervice knowledge graph construction architecture

According to Figure 5, the detailed construction steps of the IoT service knowledge graph can be obtained as follows:

(1) Guide the ontology construction under the framework of semantic description of IoT services.

(2) Extract service information and related entity property information.

(3) Compare the property concept in (2) with the ontology library, and if it is an existing concept in the ontology library, connect the service with the property in the specified sense, input, effect, or output way; if there is no such concept, update the concept in the

ontology library and add it to the knowledge graph together with the service and entity properties.

Whenever a new service is added to the knowledge graph, steps (2)(3) are repeated to achieve an iterative update of the knowledge graph.

4.3. **Service and Property Extraction.** The services in OWL-TC4, the fourth version of the OWL-S service test set, are used as an example for service and property extraction. Figure 5 shows some information about BOOK_PRICE_SERVICE service.

```
<profile:Profile rdf:ID="BOOK_PRICE_PROFILE">
<service:isPresentedBy rdf:resource="#BOOK_PRICE_SERVICE"/>
<profile:serviceName xml:lang="en">
BookPrice
</profile:serviceName>
<profile:textDescription xml:lang="en">
Uses the ISBN to return the purchase price of a given book title.
</profile:textDescription>
<profile:hasInput  rdf:resource="#_BOOK"/>
<profile:hasOutput rdf:resource="#_PRICE"/>

. . .
<process:Input rdf:ID="_BOOK">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/books.owl#Book</process:parameterType>
</process:Input>

<process:Output  rdf:ID="_PRICE">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/concept.owl#Price</process:parameterType>
</process:Output >
```

FIGURE 6. Key part of the description of the book price service

The service name can be obtained from <profile:serviceName/>in the figure, where a unique Uri will be assigned to the service according to the description framework of this paper, and <sprofile:textDescription/>, <process:input/>and <process:output/>are also similar to it. It contains the service description and the ontology information required for input, output,sense,and effect.The service and property extraction can be done by simply matching the module ontology.

4.4. **Knowledge Graph Storage for IoT Services Based on Neo4j Graph Database.** Neo4j is a graph database that stores data in a property graph structure.

Property Graphs (PG) are the most common way of storing knowledge graphs with the following three basic data types nodes: Nodes, Edges, and Properties [24].

Nodes: are entities in the graph, marked with zero to multiple text labels (labels) indicating their types, equivalent to entities.

Edges: are directed links between nodes, also called relationships. The corresponding "from node" is called the source node and the "to node" is called the target node. Edges are directed and each edge has a type, they can be navigated and queried in any direction. It is equivalent to a relationship between entities.

Properties: are key-value pairs where vertices and edges have properties.

The storage of the IoT service library can be accomplished through Neo4j's Cypher for JAVA API. The detailed steps after completing the extraction of service and entity property information in Section 4.3 are as follows.

(1) Save the extracted services as <SUri, SDescription>as a service.csv file.

(2) Save the extracted properties as <PUri, EType, State>as property.csv file.

(3) Save the relationship between service and property as <PUri, Relation,SUri>as relationship.csv file.

Take the BOOK_PRICE_SERVICE service in Figure 6 as an example, the data initialized in each of the three files for this service are:

(1)<http://127.0.0.1/services/1.1/book_price_service, This service returns a list of current purchase prices of a given book title. The prices include both new and used versions of the book.>

(2)<http://127.0.0.1/ontology/books.owl#Book, PhysicalEntityProperty, unavailable><http://127.0.0.1/ontology/concept.owl#Price, InformationEntityProperty, unavailable>

(3)<http://127.0.0.1/ontology/books.owl#Book, input, http://127.0.0.1/services/1.1/book_price_service><http://127.0.0.1/services/1.1/book_price_service, output, http://127.0.0.1/ontology/concept.owl#Price>

After the above-mentioned extraction of service and entity properties, the IoT service knowledge graph can be created on this basis. Taking three simple service nodes as an example, the relationship diagram between these three service nodes is first given below, and some of their descriptive information is given in Table 2.

Table 2. Some service information

SUri	Input	Output	Sense	Effect
Airconditioner_Service	I-Time, I-RoomATemp	∅	∅	P-RoomATemp
Time_Service	∅	I-Time	∅	∅
Tempsense_Service	∅	I-RoomATemp	P-RoomATemp	∅

Figure 7 shows the representation of some of the service and property relationships in Neo4j. After storing the knowledge graph of IoT services using Neo4j, several query

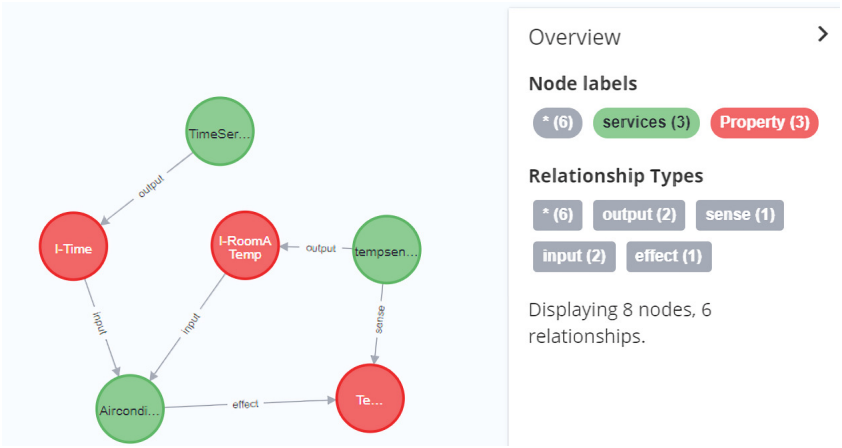

Figure 7. Partial property and service relationship in Neo4j

interfaces can be encapsulated using the query function of Cypher for JAVA API for subsequent calls to the service composition algorithm. Table 2 shows the functional description of some interfaces.

5. IoT Service Composition Based on Graph Planning.

TABLE 3. Functional description of some interfaces

Neo4j interface	**Function**
GetInput (srv)	Query the input node of srv
GetOutput (srv)	Query the output node of srv
GetSense (srv)	Query the sense node of srv
GetEffect (srv)	Query the effect node of srv
Check (prop)	Query the state of property
Check (goal)	Query whether all properties have reached the goal states

5.1. **IoT Service Composition Model.** Due to the limited functionality of individual IoT services, in order to meet complex business requirements, a composition of different services is required to obtain a greater granularity of services [25]. The service composition request consists of three components,supply, demand, and composition constraints. In the IoT service model developed in this paper, the supply is described as the input element, and the demand is described as the output element and the effect element. The composition constraints can be described as requirements for service quality metrics (cost, reliability, responsiveness, etc.) and can be easily extended to the IoT service model established in this paper. And the composed service obtained by composing is an execution sequence of services: $csrv =< srv_1, srv_2, ...srv_n >$. The service composition is defined as follows:

Definition 5.1. *(service composition) For a service composition request* $req = (P^R, Q^R, A^R)$ *with* $\forall p \in P^R : p.ust = available$*, find a composed service csrv such that after csrv is executed, it satisfies* $\forall p \in Q^R : p.ust = available \wedge \forall p \in A^R : p.ast = affected$.

On the basis of Definition 5.1, a service composition request $req = (P^R, Q^R, A^R)$ can be converted into a state transition system defined as follows:

Definition 5.2. *(state transition system of service composition) For a service composition request* $req = (P^R, Q^R, A^R)$ *can be convert to a state transition system* $STS = (Z, z_0, Z^e, T, E)$*,where*

•$Z = \{z | z =< (p_0.ust, p_0.ast), (p_1.ust, p_1.ast), \dots >\}$*,where* p_i *is entity property, Z is a set of states,and a state z is a vector of all entity property state values.*

• $z_0 \in Z$ *is the initial state, with* $\forall p \in P^R : p.ust = available$

• $Z^e \subseteq Z$ *is the set of terminable states and a terminable state Z satisfies* $(\forall p \in Q^R : p.ust = available) \wedge (\forall p \in A^R : a.ast = affected)$.

• $T = \{srv\}$ *is the set of transition labels, and a transition label represents an IoT service.*

• $E \subseteq Z \times T \times Z$ *is the set of transitions, for a transition* $e = \{(z, srv, z')\}$*,In state z. srv is enabled, and the state changes to z' after srv is executed.*

5.2. **Algorithm of Creating IoT Service Planning Graph.** Graph planning method is proposed by Blum and Furst [26], which is a method to solve the planning problem in the form of a graph . Considering the characteristics of IoT services, the IoT service planning graph is different from the traditional planning graph in that there are no negative effect edges as well as mutually exclusive services, so the IoT service planning graph in this paper contains two types of nodes, one is the service node in the service layer and the

other is the node of state change triggered by the service in the state layer. To apply the IoT service composition to the graph planning domain, the data structure of the IoT service planning graph is first defined as follows:

Definition 5.3. *(IoT service planning graph) An IoT service planning grpah IoTSPG = (Z, T, L), where*

•*L is the current number of levels of planning graph.*

•*Z is the state level of planning graph,* $Z = (Z_{information}, Z_{physical})$*,* $Z_{information}$ *records changes of information entity properties in the current level,* $Z_{physical}$ *records changes of physical entity properties in the current level*

•*T is the service level of planning graph, records the services that executed in the current level.*

Algorithm 1 describes the specific creation process of the IoT service planning graph Algorithm 1 generates and extends the IoT service planning graph, first transforming it

Algorithm 1 Create IoT Service Planning Graph

Input: A Compose Request $STS = (Z, z_0, Z^e, T, E)$, A max num of levels in planning graph n

Output: A planning graph $G = (Z_1, T_1, Z_2, T_2, \ldots, Z_n, T_n)$

```
1: Z_1 = z_0
2: for i ∈ [0, n] do
3:   if (Z_i ⊄ Z^e) ∧ (Z_i ≠ Z_{i−1}) then
4:     for each srv ∈ T do
5:       if enable(srv) then
6:         T_i ← srv
7:         Z_{i+1} ← Z_{i+1} + execute(srv)
8:       end if
9:     end for
10:  end if
11:  G ← (Z_i, T_i)
12: end for
```

into the state transition system $STS = (Z, z_0, Z^e, T, E)$ based on the composition request $req = (P^R, Q^R, A^R)$, and creating the first state level Z_1 based on z_0, obtaining the set of enabling services by judging the state of the information entity properties in Z_1, and adding the set of services to the service level T_1, while execute the service and add the resulting property changes to the next state level Z_2, and repeat this process within the set maximum number of levels until the state level of the planning graph reaches the terminable state, or the state level reaches the stable level, i.e., the execution of any service does not trigger a new properties state the change, at which time the expansion of the IoT service planning graph ends.

5.3. **IoT Service Planning Solution Extraction.** In the graph planning based IoT service composition solving process, if the state reaches terminable state, then the graph planning algorithm enters the solution extraction phase, and the specific process of service composition solution extraction is described in Algorithm 2 and 3.

First, for a composition request $req = (P^R, Q^R, A^R)$, let $target = Q^R \cup A^R$, for any entity property $a \in target$, use the query interface opened by Neo4j in Chapter 4 to find the services that satisfy the influence on this entity property in the service layer above the planning graph srv,and put it into the solution sequence $TStemp$ of the current level, and

Algorithm 2 TranslistExtract

Input: A planning graph G, A set of targets $targets$, Current num of level L
Output: A Translist $TS =< z_0, srv_0, z_1, srv_1, z_2, \ldots, Z_n >$ or $null$

if $L = 0$ **then**
  return TS
else
  StepSearch$(G, target, L)$
end if

Algorithm 3 StepSeach

Input: A planning graph G, A set of targets $targets$, Current num of level L
Output: A Translist $TS =< z_0, srv_0, z_1, srv_1, z_2, \ldots, Z_n >$ or $null$

if $target = null$ **then**
  for each $srv \in TStemp$ **do**
    $input \leftarrow GetInput(srv)$
  end for
  TranslistExtract$(G, inputs, L-1)$
  return TS
else
  for each $a \in target$ **do**
    for each $srv \in T_L$ **do**
      $result \leftarrow GetOutput(srv) + GetSense(srv) + GetEffect(srv)$
      if $a \in result$ **then**
        $TStemp \leftarrow TStemp + srv$
      end if
    end for
  end for
  if $TStemp = null$ **then**
    return $null$
  else
    choose a random $srv \in TStemp$
    $TS \leftarrow TS + TStemp$
    $result \leftarrow GetOutput(srv) + GetSense(srv) + GetEffect(srv)$
    return $StepSearch(G, target - result, L)$
  end if
end if

repeat the above process until the initial state P^R of the composition request is reached, and extract all the composed sequence TS.

6. Experimental Analysis.

This section evaluates the rule-based reasoning for IoT service ontologies and the composed approach for IoT services, respectively.

6.1. Rule Reasoning Evaluation.

The rule inference is verified in the context of smart home. The user can get service A by searching, and service A has alternative service B, and service B has service C related to it, and the function of service C is exactly what the user is interested in. Through the joint reasoning of Rule1, 2, 3, and 4 in Section 3.3.2, the information about service C can be shown to the user.

Another example is that when the user inputs the concept query of light A, the user can be shown the detailed information of the instances of light A existing in the constructed

ontology knowledge base, such as light A has the property of light intensity and the status of this property is unavailable, etc. If the user can provide a set of properties with the status of available, the currently available services can be automatically inferred through Rule5.

The SWRL Tab operation panel in Protégé runs inference based on the written rules. Figure 8 shows that the inference machine generated 238 new axioms from ontology knowledge with custom SWRL rules.

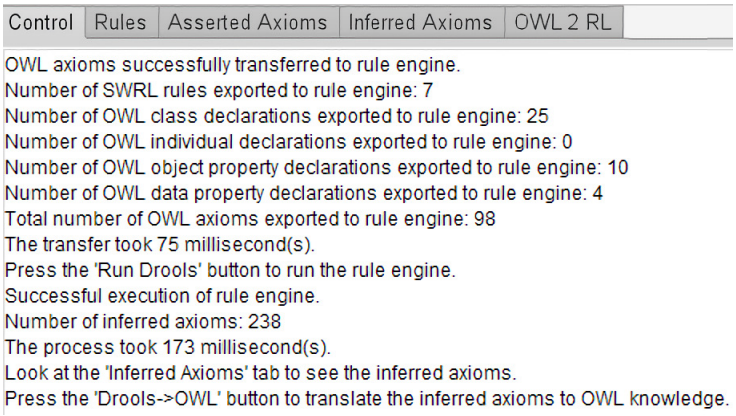

FIGURE 8. Inference Axioms

The new knowledge generated by inference is displayed in Protégé with a yellow background, and the inference process can be viewed by clicking on the question mark to the right of the inferred knowledge. Figure 9 shows the inference of the inference of the Lightsense service to the enabled service and its process.

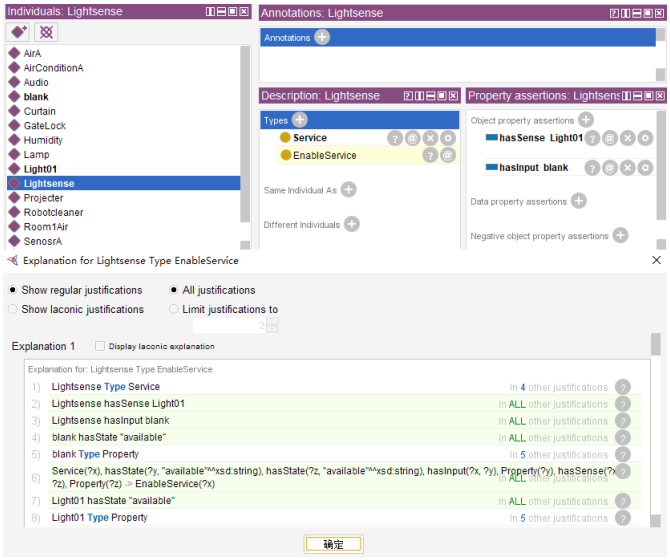

FIGURE 9. inference about LightSense

6.2. **Composition Method Evaluation.** The feasibility and effectiveness of the proposed IoT service composition method is verified by comparing the service composition method using knowledge graph and graph planning with the service composition method using only graph planning, and recording the success rate of the composition and the average running time of the algorithm. In addition, the two are denoted as K-IOTSPG

and IOTSPG respectively in order to present the results. The experimental environment for this paper is Win10 64-bit operating system, Intel(R) Core(TM) i7-10700k CPU @ 3.80GHZ, 16G RAM.

As there is no relevant standard dataset, this paper uses a random generation method to simulate entities, entity properties, services, and user requirements. In particular, 1300 entity properties are randomly generated, with 1/2 of each physical entity propertiy and 1/2 of each information entity property; the input, output, sense, effect, and initial and target states of the service and user requirements are randomly selected from these 1300 entity properties.

The size of the set of input, output, sense, and effect properties of the service is chosen randomly between 10–20, the initial state properties of the user requirements is chosen randomly between 10-30 and the target state is chosen randomly between 100–200. The number of simulated services is increased in steps of 25, from 50 up to 1000, and 100 service requests are randomly generated for each size of service separately, and finally the success rate and average time spent on the 100 composed requests are calculated. The above experiments were conducted for each of the two methods. The success rates of the compositions are shown in Figure 10.

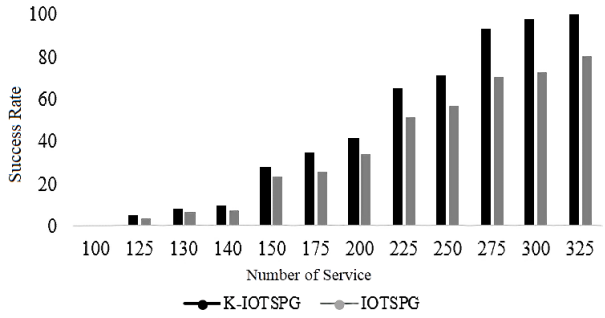

FIGURE 10. Rate of successful service composition

As can be seen from Figure 10, under the same conditions, the success rate of the composition of the two methods increases as the number of services increases, and the composition success rate of K-IOTSPG is consistently higher than that of IOTSPG, which is because IOTSPG is mainly based on keywords when performing service matching, while K-IOTSPG combines the semantic ontology of IoT services and expands the matching scope and therefore has a higher success rate. The forward expansion time, backward search time, and total planning time for the two methods are shown in Figures 11, 12, and 13.

As can be seen in Figure 13, the total algorithm time for both methods increases as the number of services increases under the same conditions. Although whether or not the knowledge graph is applied has little impact on the algorithm running time when the number of services is small, as can also be seen by looking at the forward expansion time in Figure 11, sometimes when the knowledge graph interface is not applied, the planning graph is expanded faster instead. However, as the number of services increases, the execution efficiency of K-IOTSPG is significantly higher than IOTSPG. This is because K-IOTSPG invokes the search interface provided by the Neo4j graph database in both the forward expansion and backward search phases of the graph planning phase, effectively reducing the time required to find and match between services and entity attributes.

It can also be seen that the algorithm's time increases at a faster rate until the number of services reaches 350, while after 350 the time increases at a slower rate and is not strictly

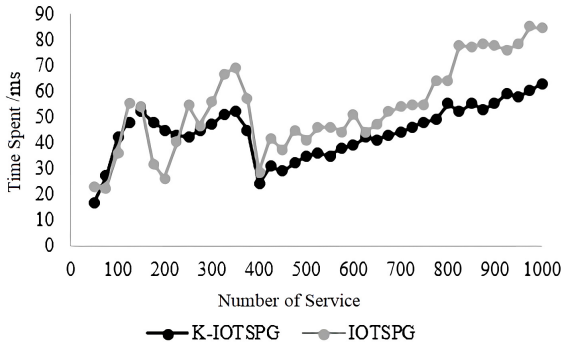

FIGURE 11. forward expansion time

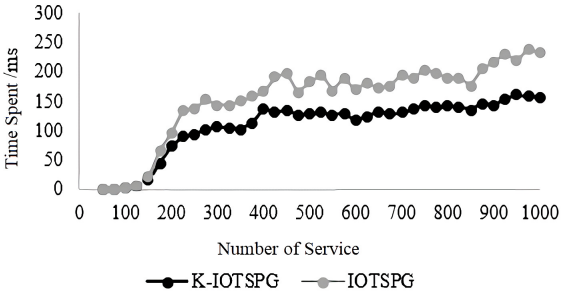

FIGURE 12. backward search time

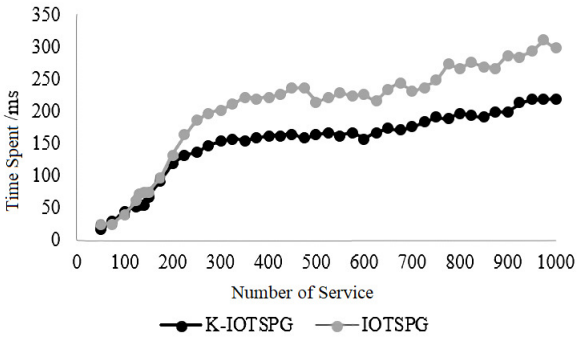

FIGURE 13. total planning time

monotonically increasing. By looking at the results of the service compositions and the forward search time in Figure 11, it can be seen that the reason for this phenomenon is that the size of the services is so large that, if the number of services exceeds a certain threshold, a large number of services can be enabled by the initial conditions given by the user alone, so that the target state is reached by only one layer of services in the planning graph, making the forward expansion phase of the planning graph take less time instead. After this threshold, as the number of services continues to increase, the increase in time for the forward expansion of the planning graph is faster, but the increase in time for the backward search tends to level off, and the former accounts for a lower proportion of the overall algorithm process time, so that the total running time of the algorithm also increases more slowly, indicating that the IoT service composition method is also suitable for large-scale service scenarios.

7. **Conclusions.** In this paper, we construct a semantic framework for the description of IoT services and their entities, build an IoT service ontology using Protégé, design a SWRL rule language for automatic reasoning about the ontology, mine part of the hidden knowledge, avoid the missing information of the ontology caused by manual building of the ontology, and implement the storage of the knowledge graph of IoT services in Neo4j, and finally implement the IoT service composition based on Graph Plan. The next work will try to automatically build IoT service ontology by machine instead of manual definition by domain experts; further generalize and design more relevant IoT service rules to enrich the knowledge in the ontology library; further improve the composition method to improve the efficiency of service composition.

Acknowledgment. This work is partially supported by Natural Science Foundation of Fujian Province (No. 2020J01877), Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF-IPIC202207), Open Research Fundation for Key Laboratory of Hunan Province (No. 2015TP1002)

REFERENCES

[1] L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504–59513, 2019.

[2] E.-K. Wang, R.-P. Sun, C.-M. Chen, Z.-D. Liang, S. Kumari and M.-K. Khan, "Proof of X-repute blockchain consensus protocol for IoT systems," *Computers and Security*, vol. 95, 101871, 2020.

[3] C.-M. Chen, X.-T. Deng, W.-S. Gan, J.-H. Chen and S. Islam, "A secure blockchain-based group key agreement protocol for IoT," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 9046–9068, 2021.

[4] C.-M. Chen, X. Li, S.-S. Liu, M.-E. Wu and S. Kumari, "Enhanced Authentication Protocol for the Internet of Things Environment," *Security and Communication Networks*, vol. 2022, 8543894, 2022.

[5] T.-Y. Wu, X.-L. Guo, Y.-C. Chen, S. Kumari and C.-M. Chen, "SGXAP: SGX-Based Authentication Protocol in IoV-Enabled Fog Computing," *Symmetry*, vol. 14, no. 7, 1393, 2022.

[6] T.-Y. Wu, Q. Meng, S. Kumari and P. Zhang, "Rotating behind Security: A lightweight authentication protocol based on IoT-enabled cloud computing environments," *Sensors*, vol. 22, no. 10, 3858, 2022.

[7] T.-Y. Wu, T. Wang, Y.-Q. Lee, W.-M. Zheng, S. Kumari and S. Kumar, "Improved authenticated key agreement scheme for fog-driven IoT healthcare system," *Security and Communication Networks*, vol. 2021, 6658041, 2021.

[8] B.-H. Yu, "Research on Key Technologies of semantic Internet of things," Ph.D. dissertation, University of Chinese Academy of Sciences, 2021.

[9] M.-J. Lian, "Research on Key Technologies of Semantic Service-oriented IoT Middleware ," Ph.D. dissertation, University of Chinese Academy of Sciences, 2021.

[10] F. -C. Chang and H. -C. Huang, "A Survey on Intelligent Sensor Network and Its Applications," *Journal of Network Intelligence*, vol. 2016 ,no. 1, pp. 1–15, 2016.

[11] S. Fang, L.-D. Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan and Z. Liu, "An Integrated System for Regional Environmental Monitoring and Management Based on Internet of Things," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 10, no. 1, pp. 72–80, 2019.

[12] R. Agarwal R, D.-G. Fernandez, T. Elsaleh, et al., "Unified IoT ontology to enable interoperability and federation of testbeds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 70–75.

[13] S. De, P. Barnaghi, M. Bauer, et al., "Service modelling for the Internet of Things," in *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2011, pp. 949–955.

[14] Q. Wei, "Research on key technologies for semantic-based IoT service discovery and provisioning," Ph.D. dissertation, University of Chinese Academy of Sciences, 2014.

[15] J. Yu, M. Wang, J. Liu , et al., "Service management mechanisms in the internet of things: an organized and thorough study," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2021, pp. 1–12, 2021.

[16] M. Hamzei and N.-J. Navimipour, "Toward efficient service composition techniques in the internet of things," *IEEE Internet of Things Journal*, vol. 2018, no. 5, pp. 3774–3787, 2018.
[17] D. Androec, "Using JSON-LD to Compose Different IoT and Cloud Services," *arXiv preprint* 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1809.08233
[18] T. Baker, M. Asim, H. Tawfik, et al., "An energy-aware service composition algorithm for multiple cloud-based IoT applications," *Journal of Network & Computer Applications*, vol. 2017, no. 89, pp. 96–108, 2017.
[19] H. Nacer and D. Aissani, "Semantic web services: Standards, applications, challenges and solutions," *Journal of Network and Computer Applications*, vol. 2014, no. 44, pp. 134–151, 2014.
[20] T.-R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 1993, no. 5, pp. 199–220, 1993.
[21] D. Runumi, M. Deepti, et al., "SWRL reasoning on ontology-based clinical dengue knowledge base," *International Journal of Metadata, Semantics and Ontologies*, vol. 14, no. 1, pp. 39–53, 2020.
[22] Z.-L. Xu, Y.-P. Sheng, L.-R. He and Y.-F. Wang. "Review on Knowledge Graph Techniques," *Journal of University of Electronic Science and Technology of China*, vol. 45, no. 4,pp. 589–606, 2016.
[23] Q. Liu, Y. Li, H.Duan, et al., "Knowledge graph construction techniques," *Journal of Computer Research and Development*, vol. 53,no. 3, pp. 582–600, 2016.
[24] G. Li and W. Li. "Research on storage method for fuzzy RDF graph based on Neo4j," *Evolutionary Intelligence*, 2022. [Online]. Available: https://doi.org/10.1007/s12065-022-00715-0
[25] Z.-J. Ding and Z.-X. Zhou "Survey on Web Service Composition Testing," *Journal of Software*, vol. 29, no. 2, pp. 299–319, 2018.
[26] A.-L. Blum and M.-L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, no. 2, pp.281–300, 1997.