Complete W3C-Semantic's Interpretations of AP-RDF

Dewi Wardani, IAENG, Member

***Abstract*- An attributed predicate RDF (AP-RDF) has been proposed as a new alternative model to represent complex information. AP-RDF augments the function of predicate to hold attributes to present any additional triples of the main triple. AP-RDF has shown its usefulness in the beginning work by showing much better performance in terms of query time. In this work, we focus on studying whether AP-RDF fulfils the default semantic interpretation of the W3C Recommendation. In the end, we can show that AP-RDF fulfils the default semantic interpretation by extending the interpretation of property with the interpretation of predicate type. The impact of semantic interpretation shows on the retrieval process, especially on complex queries. It returns better performance compared to default RDF.**

***Index Terms* - AP-RDF, rdf, semantic-interpretations, w3c-recommendation**

I. INTRODUCTION

RDF(Resource Description Framework) is the Semantic Web's standard data model [1][2]. The information is represented as a triple SPO. As an instance, *"NaCl is made of sodium"*. The subject (S) is composed of "NaCl", the predicate (P) is composed of "made", and the subject (O) is "sodium". Any additional information on the triple makes information becoming complex. For instance, *"NaCl consists of one part of Sodium or NaCl consists of one part of Sodium and one portion of Chloride"*. The information in the real world could be more complex than those examples.

The prominent approach, N-Quads [3], has been introduced to represent complex information as an RDF that has four elements. The fourth element manages the context issue. Named Graph [4] introduced the name of a triple. It is like the fourth element in N-Quads. The name of the graph arranges the provenance, trust and authority issues. Extra information on the main triple can be as context, provenance or other types of meta-data. They make the information more complex. In this work, the predicate has more functions than its default function. The proposed novelty of AP-RDF is to extend the functionality of a predicate. A predicate also can handle other additional information on the main statement. Additional information increases the complexity of the information. The previous work shows that the model has promising performance results concerning the needed time in the retrieval process [5].

As a new serialization model based on RDF, it is necessary to evaluate whether AP-RDF meets the standard semantic interpretation of the W3C Recommendation. Ability to answer the question, of whether a model which fulfils semantic interpretation does not influence the performances but will affect the extended processes, e.g. reasoning process for automatically finding new information. Therefore, making sure a new model fulfils the default semantic interpretation of the W3C Recommendation is an elementary issue. AP-RDF extends the function of a predicate to handle information which adds the complexity of fact. It means predicates in AP-RDF have additional semantics comparing default predicates in default RDF. In brief, our goal as well the contributions is to prove that AP-RDF fulfils the default semantic interpretation of the W3C Recommendation.

Previous work has proposed some methods which emphasized representing complex information in semantic web technology [6] [3] [4]. Reification, Named Graph, and N-Quads are all W3C recommendation models in representing information in the semantic web. Those models are all are a part of the W3C Recommendation. Therefore, they are assumed to have satisfied the default semantic interpretation. The other recent model [7] did not prove whether the model fulfils the default semantic interpretation of the W3C Recommendation. The latest document explains the default semantic interpretation of the W3C Recommendation [8]. It formalized precisely several interpretations divided into the normative and the non-normative interpretations. The earlier documents [9] [10] had introduced several explanations before they became the default of the W3C recommendation. The main interpretation is a normative interpretation which contains a simple interpretation (included data-type), RDF interpretation and RDFS interpretation. Each interpretation explains its exact entailment and its patterns. The entailment maps between syntaxes and their semantic interpretations. As the default of W3C Recommendation, all new models should satisfy this default. An AP-RDF has a different approach comparing the other models. It creates instances of predicate type as a new concept in representing information. Hence, the modified syntaxes need to be assessed whether it fulfils the default semantic interpretation.

The rest document is structured as follows: Part 2 summarizes some previous work. We briefly summarize the idea of AP-RDF in Part 3. Part 4 discusses that AP-RDF fulfils the default semantic interpretation of the W3C Recommendation and the last part for the conclusion and future work.

II. RELATED WORK

A. An Attributed Predicate RDF (AP-RDF)

Firstly, we are going to explain AP-RDF, which has been proposed in [5] [11] [12]. The previous work explained in detail how the graph model inspired its serialization

Manuscript received October 31, 2021; revised June 25, 2022.
Dewi Wardani is a lecturer at Informatics Department, Universitas Sebelas Maret, Indonesia. Email: dww_ok@uns.ac.id.

was created and was studied [13] [14] [15] [16] [17] [18]. RDF triples consist of S, P and O. There are no fixed characteristics for a type and its instances. Therefore, RDF is free to design. Nodes can function as types and as instances of other types. An example is shown below:
sh:Frog rdf:type sh:Animal .
sh:Tree_frogs rdf:type sh:Frog .
Properties have the other properties. Below is an example:
sh:own rdf:type sh:personAbility .
sh:hasVehicle rdf:type sh:own .
sh:Dina sh:own sh:Frog .
RDF composers are free to compose RDF. From the above example, we can see that *sh:Frog* can act as *S* or *O*. The term *sh:own* serves as *S*, *P* and *O*. The term *sh:personAbility* is a property, but in RDF it acts as *O*. *sh:own* itself is a property. Therefore, it means that the property itself can have a property. This flexibility influences AP-RDF's approach that triple elements can have multiple functions.

The idea of AP-RDF is to add the function of predicates. It also handles extra triples to the main RDF triple. Attributes of the predicate are extra triples. The predicate has two functions to manage attributes of the predicate. As the default predicate and as the subject in a different triple. In this work, for a clear explanation, we add a few definitions for AP-RDF as below:

DEFINITION 1. (A predicate type and predicate instances) *PIT*=$\{pIT \mid p \notin PT\}$, *PT*=$\{pT \mid p \in PIT\}$,*(pIT,pT)* $\rightarrow$ *rdf:type and (PT* $\cup$ *PIT)* $\subseteq$ *P are a set of predicate type PT and a set of instances of a predicate type PIT.*
Definition 1 defines predicate instances of a predicate type, as a new feature in AP-RDF.

DEFINITION 2. (An Attributed Predicate RDF) *TRDF* =$\{S,P,O\}$, *a set of AP-RDF triples TAP-RDF. TAP-RDF*=$\{S',P',O'\}$. *S'*=$\{s \mid s\in(S\cup PIT)\}$, *P'*=$\{p \mid p\in(P\cup PIT)\}$ *and O'*=$\{o \mid o\in(O\cup PT)\}$.
Definition 2 defines that an element of triples may have multiple functions.

B. W3C Default Semantic Interpretation

Secondly, we study the W3C's Recommendation of semantic interpretations [8]. There are several interpretations:
(i) Simple Interpretations,
(ii) Literals and Data-types Interpretations,
(iii) RDF Interpretations and,
(iv) RDFS Interpretations.

There is also extra interpretations at the informative state. It is not legally the default. They are like extra information of default semantic interpretations. Those interpretations are blank nodes interpretation, skolemization, entailment rules, finite interpretations, proofs of some results and RDF reification. In this work, we emphasized investigating some parts that probably will be extended based on the AP-RDF model. The W3C has pointed out that any additional semantic that is called semantic extension must not cancel the more solid interpretations.

In short, the interpretation maps all parts of a triple of RDF. Those elements such as IRIs and Literals adds some constraints upon the set and its mapping process. The minimal interpretation is the Simple Interpretation as for the minimal truth condition. All possible extensions of the Simple Interpretations must conform to all minimal truth conditions. Any extension of interpretations can add the Simple Interpretations but cannot negate or modify them. Figure 1 describes the default RDF has property *P*. The property is used by multiple triples of RDF. Below, is an example of triples (in Turtle format):
"Indonesia capital Jakarta"
"Russia capital Moscow"
"PRC capital Beijing"
The property or the predicate "capital" is used by a couple of resources. Either as a subject or as an object. Therefore, the interpretation of property in default RDF is the exponential of two resources. Now, let's see the below example of AP-RDF (in Turtle format):
"PRC capital_1 Beijing" (line 1)
"capital_1 rdf:type capital" (line 2)
"capital_1 since 1912" (line 3)

Line 1 shows the property "capital_1" only maps PRC and Beijing. Then, it also is used as a subject to manage the additional information since. These extensions are not covered by the simple interpretation. Therefore, we need to extend the simple interpretation accordingly based on those circumstances, without against the default Simple Interpretations.

III. Results

As a new RDF model, it's important to see if AP-RDF meets the W3C's default semantic interpretations [8]. The basic goal of AP-RDF is to build predicate type instances. Attribute triples are permitted for predicate type instances. This construct is a variation on the standard triple RDF. As a result, we must assess whether this expansion entails normative semantic interpretations.

That is, the predicate instance *IPI* is the default predicate in AP-RDF. *(IPI* $\cup$ *IPT)*$\subseteq$ *IP* obtained automatically. The goal is to ensure that a predicate instance's syntax and the predicate type's semantic interpretation are identical. Otherwise, it adheres to the W3C's conventional semantic interpretation [8].

DEFINITION 3. Simple Interpretations of AP-RDF
Except for the property mapping function, AP-RDF's simple interpretation has all of the elements of a typical semantic interpretation *(IEXT)*. The Simple Interpretation *I* cover:
• *IRI* resource interpretation function *IS:V*$\rightarrow$*IR* $\cup$ *IP*
• Interpretation function for typed literal *IL* and untyped literal *LV*.
• Enhanced interpretation of properties *IP*. *IEXT* of AP-RDF is *IP* $\rightarrow$ *IR x IR*.

Figure 2 also show that a simple interpretation of AP-RDF makes it impossible to distinguish between the "rdf:type" vocabulary because all IRIs are considered equally. The property "rdf:type" indeed has special semantics. Therefore, the simple interpretation is not enough to explain the AP-RDF interpretation.

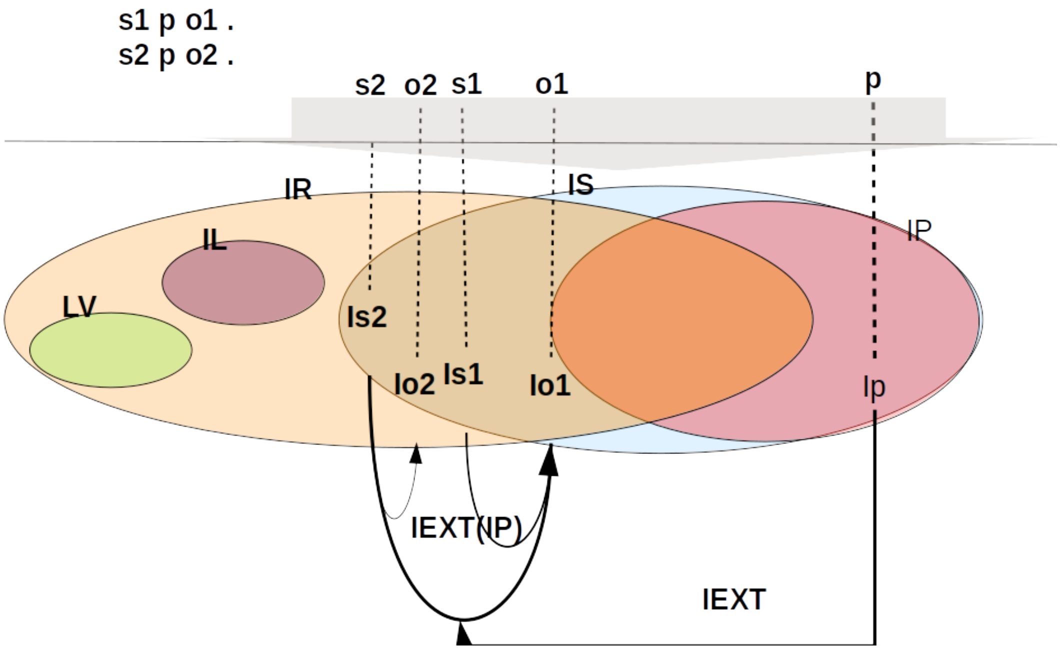

Fig. 1: A standard scheme for simple interpretation. Predicates map multiple pairs of resources. IP $\rightarrow 2^{IRxIR}$

DEFINITION 4. RDF Interpretation of AP-RDF

AP-RDF RDF interpretation is an extension of plain AP-RDF interpretation that adds a mapping function for the "rdf:type" property vocabulary to *IPT*, following all standard features of semantic RDF interpretation and the illustration is shown in Figure 3. *XIP* ∈ *IPT* if *IP(rdf:type)*→{⟨*X,XIP*⟩} and *X*∈*IPIT*.

The *IPIT* and *IPT* interpretations must be defined. In addition, we define *IRT*. The interpretation of the resource type is known as *IRT*. All axioms from rdf: vocabularies must be mapped automatically. Figure 2 depicts those axioms in a series of syntaxes for ease reading. As below is AP-RDF that coverages RDF Interpretations:

• Using the RDF vocabulary to add *IS* should also adhere to the RDF axioms. RDF Semantic Interpretation [8] contains a detailed collection of RDF axioms. With or without adequate input, the ability to understand XMLLiteral *IL*. *IL*∈ *LV* is a well-typed XMLliteral if *IL* is a well-typed XMLliteral. *IL*∉*LV* if *IL* is entered improperly XMLliteral

• Using RDF semantics, add *IP*. *rdf:Property* indicates each potential predicate in the RDF data set. *X*∈ *IP* is just for ⟨*X, I (rdf: Property)*⟩ ∈ *IEXT (I (rdf: type))*. The predicate type is defined by *IPT*, and the resource type is defined by *IRT*. The resource's *rdf:type* interpretation is *IRT*, while the property's *rdf:type* interpretation is *IPT*. As a result, *(IPT* ⊆ *IP)* and *(IRT* ⊆ *IR)* are automatically generated. *IP(rdf: type)* → {⟨*X,XIPT*⟩ }, where *X* ∈ *IPIT* and *XIPT* ∈ *IPT*. *IR(rdf: type)* → {⟨*Y,YIR*⟩}, where *YIR* ∈ *IRT* and *YIR* ∈ *IR*.

• Using the *IPT* and *IRT* extensions, add *IP* extensions. There is also a property called *IPT*. As a result, *IEXT(IPT)* and *IEXT(IPT)* ⊆ *IEXT* are both extensions of *IPT*. There's also *IRT* and *IEXT (IRT)* ⊆ *IEXT*, in addition to *IPT*. As a result, {*IEXT(IPT)* | *IPIT(rdf:type)*} → {⟨*X,XIPT*⟩} ∧ {⟨*X*∈ *IPIT*⟩}, {⟨*IEXT(IRT)* | *IR(rdf: type)*⟩} → {⟨*Y,YIR*⟩} ∧ {⟨*Y,IR*⟩}, *and ((IEXT(IPT)* ∪ *IEXT(IRT)) IEXT)* ⊆ *IEXT)*

All of the AP-RDF interpretations are defined in RDF interpretation, as seen in Figure 3. As a result, RDF interpretation is the bare minimum of AP-RDF interpretation. This job does not go into detail about how to understand the RDFS. This is because the RDFS semantic interpretation is identical to the conventional W3C semantic interpretation. If the reader wishes to learn more about the RDFS interpretation, W3C Recommendation [8] is a good place to start.

DEFINITION 5. RDFS Interpretations of AP-RDF

AP-RDF's RDFS (RDF Schema) interpretation is an extension of AP-RDF's RDF interpretation, adding the *rdfs:property* mapping function according to all the characteristics of the standard RDFS W3C semantic interpretation. {*T:*⟨*T,Z*⟩} *IEXT(I(rdf: type))* is the definition of the class extension *ICEX(Z)*. A *Z class* extension is what it's called.The illustration of this representations is shown in Figure 4

For a more detailed explanation of how AP-RDF meets the standard W3C semantic interpretation, the following example is used to explain the interpretation.

"Marie has set up a new shop called Pinked with a budget of $100 million. The StartupPost writes that the creation process took place in Hongkong. This triple was inserted by Jims. The founding process. led by Lanny, started on January 10, 2013, and ended on February 1, 2013. Then continue led by Lisa until 6 weeks. Marie also founded another shop called Dustypinked".

Shop.ttl has been renamed from the previous example. Although we present our proving process in this paper, *Shop.ttl* can be utilized with a variety of data sets. *Shop.ttl's* primary and secondary lines are examples of the AP-RDF primary extension. In the first line, *sh:set_1* is an example of *PT sh:set* as the bogus of the *PT* itself. Then, utilizing

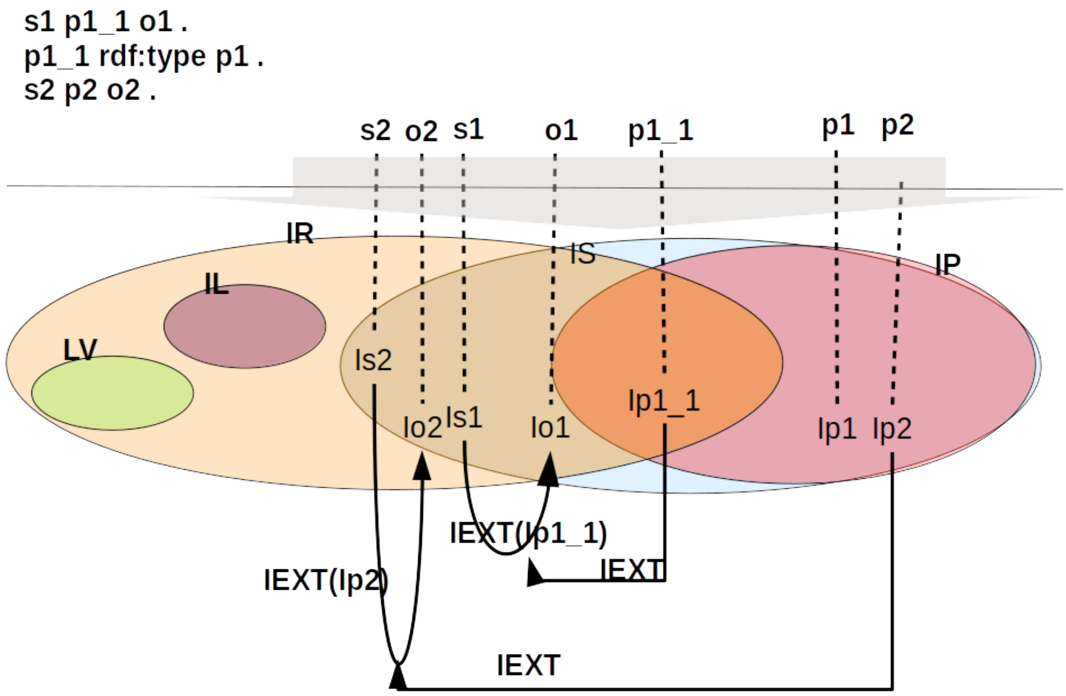

Fig. 2: A simple AP-RDF interpretation scheme. Each predicate maps only a small number of resources. $IP \rightarrow IR \times IR$

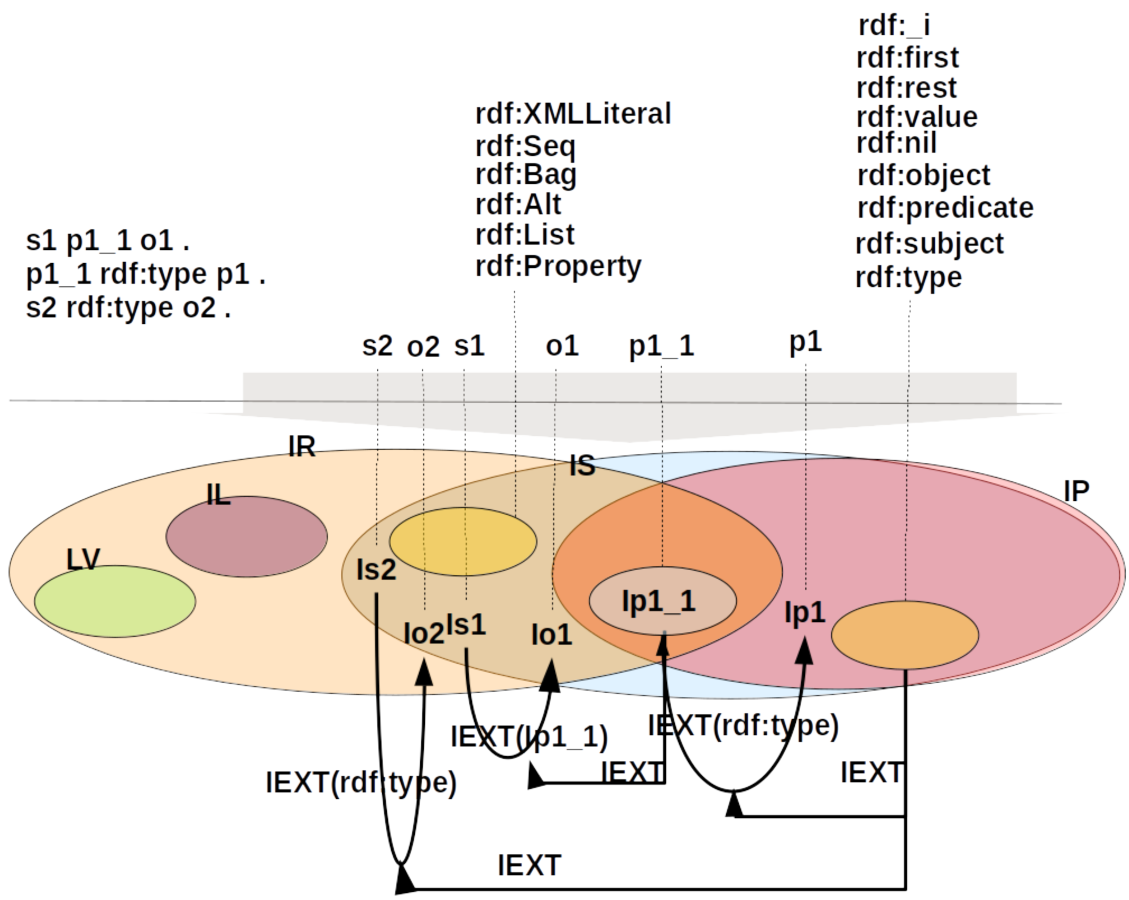

Fig. 3: The schema of AP-RDF RDF Interpretations. The predicate example is on the intersection of *IR* and *IP*

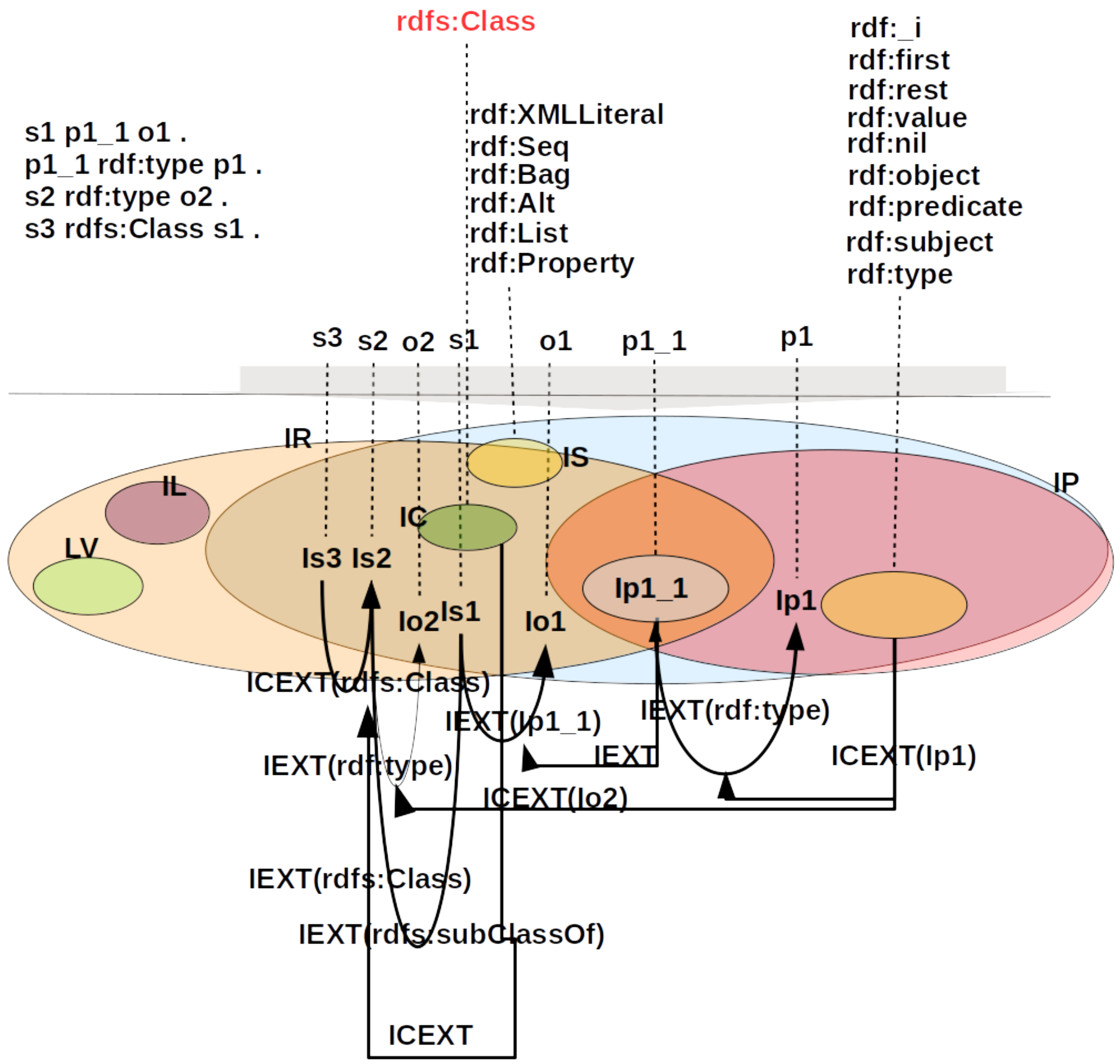

Fig. 4: The schema of AP-RDF RDFS Interpretations. It shows not much different comparing to the default one

the *PT*, we establish predicate attributes on the primary triples, as demonstrated in line two. In most cases, RDF will use the *PT sh:set*. The predicates are semantically related to the properties, while the PIT are semantically linked to their *PT*.

Shop.ttl:
@PREFIX e: <http://example.org/shop >.
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >.
sh:Marie sh:set_1 sh:Pinked .
sh:set_1 sh:city_1 sh:Hongkong .
sh:set_1 sh:invest "US$100M" .
sh:set_1 sh:head_1 sh:Lanny .
sh:set_1 sh:head_2 sh:Lisa .
sh:set_1 sh:news_1 sh:The StartupPost .
sh:head_1 sh:begin "2013-01-01"^^xsd:date .
sh:head_1 sh:stop "2013-02-01"^^xsd:date .
sh:head_2 sh:term "6"^^xsd:integer .
sh:news_1 sh:reportBy sh:Jims .
sh:Marie sh:set sh:Dustypinked .
sh:set_1 rdf:type sh:set .
sh:city_1 rdf:type sh:city .
sh:head_1 rdf:type sh:head .
sh:head_2 rdf:type sh:head .
sh:news_1 rdf:type sh:news .

A simple interpretation *I* of *Shop.ttl* based on Definition 3 is explained as follows:

- IS is obtained as follows: *sh:Marie→ x1; sh:Pinked → x2; sh:Hongkong → x3; sh:Lanny → x4; sh:The StartupPost → x5; sh:Jims → x6; sh:city → x7; sh:head → x8; sh:news → x9; sh:Dustypinked → x10; sh:Lisa → x11; sh:set_1 → x12; sh:invest → x13; sh:city_1 → x14; sh:head_1 → x15; sh:begin → x16; sh:stop → x17; sh:news_1 → x18; sh:reportBy → x19; sh:head_2 → x20; sh:term → x21; sh:set → x22*
- *IL* is a typed literal function. *"2013-01-01" ^^xsd: date → L0; "2013-02-01" ^^xsd: date → L1; "6" ^^xsd:integer → L2. IL* = {*L0, L1, L2*}
- *LV* is a partial mapping of untyped literals. *"US$100M" → L3. LV* = {*L3*}.

• *IR* is a set of non-empty resources, the world of *I*. The *IR* is obtained as follows: $IR = \{(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21) \cup IL \cup LV\}$
• *IP* is a set of properties. The *IP* is obtained as follows: $IP = \{ x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22\}$
• *IEXT* is a mapping function for each property. *IEXT* is obtained as follows: $IEXT = x12 \rightarrow \{\langle x1,x2\rangle\}$; $x13 \rightarrow \{\langle x11,L3\rangle\}$; $x14 \rightarrow \{\langle x11,x3\rangle\}$; $x15 \rightarrow \{\langle x11,x4\rangle\}$;$x16 \rightarrow \{\langle x15,L0\rangle\}$; $x17 \rightarrow \{\langle x15,L1\rangle\}$; $x18 \rightarrow \{\langle x14,x5\rangle\}$; $x19 \rightarrow \{\langle x14,x12\rangle\}$; $x20 \rightarrow \{\langle x13,x11\rangle\}$; $x21 \rightarrow \{\langle x21,L2\rangle\}$; $x22 \rightarrow \{\langle x1,x11\rangle\}$

The assessment of simple interpretations has a few flaws:
(i) Some properties, such as *sh:head_1*, *sh:head_2*, are instances of predicate type. They are derived from the sh:head predicate type. A property is an instance of the predicate type. As a result, the predicate instance is also a property.
(ii) The property includes the predicate type. It should be defined as a property as an extension of the property. As a result, simple interpretations are insufficient. More semantic interpretations for AP-RDF are required.

The RDF Interpretations extension comes next. RDF Interpretations is a collection of simple and further RDF interpretations from vocabularies with the prefix rdf: [8].
The following are the RDF interpretations of *Shop.ttl* based on Definition 4:
• The list of RDF axiom as below will be added into *IS*. We obtain *IS* as below:
sh:Marie $\rightarrow$ *x1*; *sh:Pinked* $\rightarrow$ *x2*; *sh:Hongkong* $\rightarrow$ *x3*; *sh:Lanny* $\rightarrow$ *x4*; *sh:TheStartupPost* $\rightarrow$ *x5*; *sh:Jims* $\rightarrow$ *x6*; *sh:city* $\rightarrow$ *x7*; *sh:head* $\rightarrow$ *x8*; *sh:news* $\rightarrow$ *x9*; *sh:Dustypinked* $\rightarrow$ *x10*; *sh:Lisa* $\rightarrow$ *x11*; *sh:set_1* $\rightarrow$ *x12*; *sh:invest* $\rightarrow$ *x13*; *sh:city_1* $\rightarrow$ *x14*; *sh:head_1* $\rightarrow$ *x15*; *sh:begin* $\rightarrow$ *x16*; *sh:stop* $\rightarrow$ *x17*; *sh:news_1* $\rightarrow$ *x18*; *sh:reportBy* $\rightarrow$ *x19*; *sh:head_2* $\rightarrow$ *x20*; *sh:term* $\rightarrow$ *x21*; *sh:set* $\rightarrow$ *x22*
The rest follows the default interpretation [8]
• $IL = \{L0, L1, L2, L3\}$.
• Add *IR* with the vocabulary of RDF. We obtain *IR* as below:
$IR=\{(x1, x2, x3, x4, x5, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, A, Cx, Dy \mid x \in \{0,...,12\}, y \in N) \cup IL\}$
• Based on Definition 4, $IPT=\{x8,x9,x10,x11\}$ and $IRT= \emptyset$. We obtain *IP* as follows: $IP=\{(x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, A, C2, C3, C4, C7, C8, C12, Dy \mid y \in N) \cup IPT\}$ • In the example, some *IP* in RDF vocabularies do not exist, therefore $IEXT(C2) = IEXT(C3) = IEXT(C4) = IEXT(C7) = IEXT(C8) = IEXT(C12) = IEXT(Dy)$ *for* $y \in N$. All of the XMLLiterals in *IL* are ill-typed. As a result, *IEXT(IL)* equals *IEXT(I(rdf:type))*. The following is how we get $IEXT = x12 \rightarrow \{\langle x1,x2\rangle\}$; $x13 \rightarrow \{\langle x12,L3\rangle\}$; $x14 \rightarrow \{\langle x12,x3\rangle\}$; $x15 \rightarrow \{\langle x12,x4\rangle\}$; $x16 \rightarrow \{\langle x15,L0\rangle\}$; $x17 \rightarrow \{\langle x15,L1\rangle\}$; $x18 \rightarrow \{\langle x14,x5\rangle\}$; $x19 \rightarrow \{\langle x14,x12\rangle\}$; $x20 \rightarrow \{\langle x12,x13\rangle\}$; $x21 \rightarrow \{\langle x21,L2\rangle\}$; $x22 \rightarrow \{\langle x1,x11\rangle\}$; $A \rightarrow \{\langle x12,x7\rangle, \langle x15,x8\rangle, \langle x16,x9\rangle, \langle x19,x10\rangle\}$; $C2= C3= C4= C7= C8= C12= Dy= \emptyset$ *for* $y \in N$

The default property IP and the property type *IPT* are included in the AP-RDF. Property instances *IPI* and property types *IPT* that do not have instances are included in the default property *IP*. *IP* is a portion of *IR*, and all properties are subsets of *IP*. The above explanation demonstrates that AP-RDF in *Shop.ttl* meets RDF Interpretation requirements. If the AP-RDF data collection contains triples that use the prefix-based *rdfs:*, the RDFS Interpretations are only required. The RDFS Interpretations will simply follow the default RDFS semantic interpretations without any further extension mapping because the modification is based on a *rdf:type* extension. *Shop.ttl* does not require RDFS semantic interpretations since the AP-RDF in *Shop.ttl* does not have any *rdfs:vocabulary*.

As far as that property of predicate is not the default term in RDF and that a predicate can have such an instance of it in RDF, it needs a bit tricky way to present AP-RDF with the default syntax. Moreover, there is not yet an engine that can represent AP-RDF. The way to represent AP-RDF is by using a surrogate class. The surrogate is used to help in tackling the predicate type and the instance of the predicate. Below is the OWL file which is composed by Protégé.

```
@prefix : <http://example.org/ex#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@base <http://example.org/ex>.
<http://example.org/ex>rdf:type owl:Ontology .
### http://www.w3.org/1999/02/22-rdf-syntax-ns#type
rdf:type rdf:type owl:AnnotationProperty .
### http://www.w3.org/2002/07/owl#qualifiedCardinality
owl:qualifiedCardinality rdf:type owl:AnnotationProperty .
### http://example.org/ex#city
:city rdf:type owl:ObjectProperty ;
rdfs:range [ rdf:type owl:Restriction ;
owl:onProperty :city ;
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
owl:onClass :City
] .
### http://example.org/ex#city_1
:city_1 rdf:type owl:ObjectProperty .
### http://example.org/ex#head
:head rdf:type owl:ObjectProperty ;
rdfs:subPropertyOf owl:topObjectProperty .
### http://example.org/ex#head_1
:head_1 rdf:type owl:ObjectProperty .
### http://example.org/ex#head_2
:head_2 rdf:type owl:ObjectProperty .
### http://example.org/ex#insertedBy
:insertedBy rdf:type owl:ObjectProperty .
### http://example.org/ex#news
:news rdf:type owl:ObjectProperty .
### http://example.org/ex#news_1
:news_1 rdf:type owl:ObjectProperty .
```

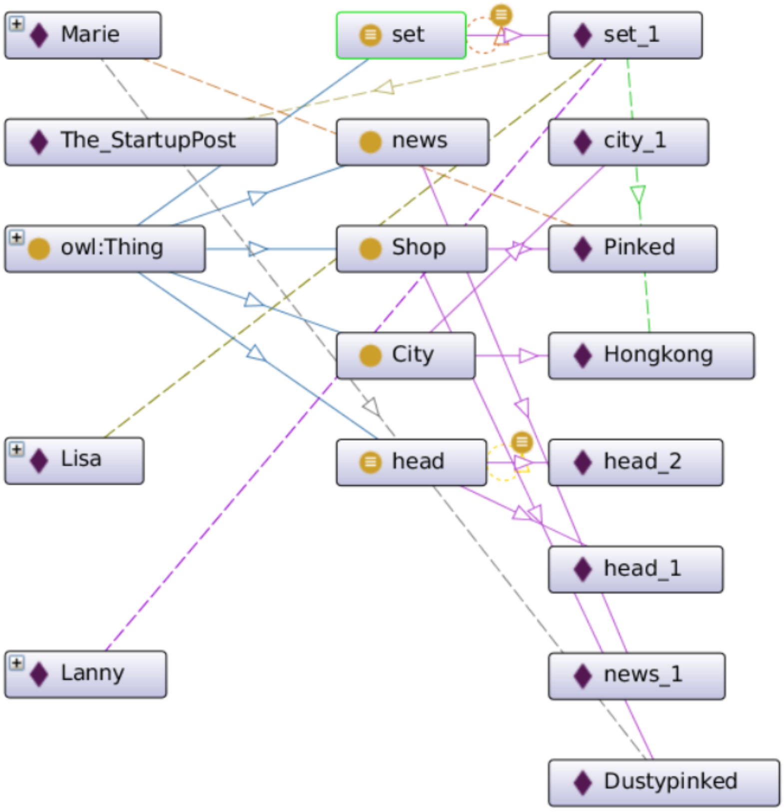

Fig. 5: The OWL graph of AP-RDF of *Shop.ttl*

http://example.org/ex#reportBy
:reportBy rdf:type owl:ObjectProperty .
http://example.org/ex#set
:set rdf:type owl:ObjectProperty ;
rdfs:subPropertyOf owl:topObjectProperty ;
rdfs:domain [rdf:type owl:Restriction ;
owl:onProperty :head ;
owl:allValuesFrom :head
] ;
rdfs:range [rdf:type owl:Restriction ;
owl:onProperty :set ;
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
owl:onClass :Shop
] .
http://example.org/ex#set_1 :set_1 rdf:type owl:ObjectProperty ;
rdfs:domain [rdf:type owl:Restriction ;
owl:onProperty :set_1 ;
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
owl:onClass :Shop
] ;
rdfs:range [rdf:type owl:Restriction ;
owl:onProperty :set_1 ;
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
owl:onClass :Shop
] .
http://example.org/ex#begin
:begin rdf:type owl:DatatypeProperty .
http://example.org/ex#duration
:duration rdf:type owl:DatatypeProperty ;
rdfs:range xsd:integer .
http://example.org/ex#endDate
:endDate rdf:type owl:DatatypeProperty ;
rdfs:range xsd:dateTime .
http://example.org/ex#hasBudget
:hasBudget rdf:type owl:DatatypeProperty ;
rdfs:range xsd:integer .
http://example.org/ex#invest
:invest rdf:type owl:DatatypeProperty .
http://example.org/ex#startDate
:startDate rdf:type owl:DatatypeProperty ;
rdfs:range xsd:dateTime .
http://example.org/ex#stop
:stop rdf:type owl:DatatypeProperty .
http://example.org/ex#time
:time rdf:type owl:DatatypeProperty .
http://example.org/ex#City
:City rdf:type owl:Class .
http://example.org/ex#Shop
:Shop rdf:type owl:Class .
http://example.org/ex#head
:head rdf:type owl:Class ;
owl:equivalentClass [rdf:type owl:Restriction ;
owl:onProperty :head ;

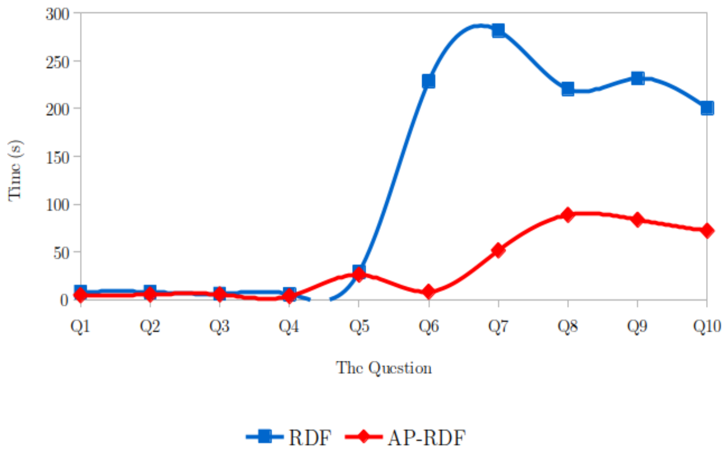

Fig. 6: The performance's result of World bank data-set

Fig. 7: The performance's result of World bank data-set 2 comparing the other models

```
owl:allValuesFrom :head
] .
### http://example.org/ex#news
:news rdf:type owl:Class .
### http://example.org/ex#set
:set rdf:type owl:Class ;
owl:equivalentClass [ rdf:type owl:Restriction ;
owl:onProperty :set ;
owl:allValuesFrom :set
] .
### http://example.org/ex#Dustypinked
:Dustypinked rdf:type owl:NamedIndividual ,
:Shop .
### http://example.org/ex#Hongkong
:Hongkong rdf:type owl:NamedIndividual ,
:City .
### http://example.org/ex#Jims
:Jims rdf:type owl:NamedIndividual .
### http://example.org/ex#Lanny
:Lanny rdf:type owl:NamedIndividual .
### http://example.org/ex#Lisa
:Lisa rdf:type owl:NamedIndividual .
### http://example.org/ex#Marie
:Marie rdf:type owl:NamedIndividual ;
:set :Dustypinked ;
:set_1 :Pinked .
### http://example.org/ex#Pinked
:Pinked rdf:type owl:NamedIndividual ,
:Shop .
### http://example.org/ex#The_StartupPost
:The_StartupPost rdf:type owl:NamedIndividual .
### http://example.org/ex#city_1
:city_1 rdf:type owl:NamedIndividual ,
:City .
### http://example.org/ex#head_1
:head_1 rdf:type owl:NamedIndividual ,
:head ;
:begin "Januari"^^xsd:string ;
```

:stop "February"^^xsd:string .
http://example.org/ex#head_2
:head_2 rdf:type owl:NamedIndividual ,
:head ;
:time "6"^^xsd:int .
http://example.org/ex#news_1
:news_1 rdf:type owl:NamedIndividual ,
:news ;
:reportBy :Jims .
http://example.org/ex#set_1
:set_1 rdf:type owl:NamedIndividual ,
:set ;
:city_1 :Hongkong ;
:head_1 :Lanny ;
:head_2 :Lisa ;
:news_1 :The_StartupPost ;
:invest "100"^^xsd:int .
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger .

It can be explained of the snippets bellow:

:set rdf:type owl:ObjectProperty ; (1)
:set_1 rdf:type owl:ObjectProperty ; (2)
:set rdf:type owl:Class ; (3)
owl:equivalentClass [rdf:type owl:Restriction ;
owl:onProperty :set ;
owl:allValuesFrom :set
] .
:set_1 rdf:type owl:NamedIndividual ,
:set ; (4)
:head_1 :Lanny ;
:head_2 :Lisa ;
:city_1 :Hongkong ;
:news_1 :The StartupPost ;
:invest 100 .

The graph's illustration is shown in Figure 5. The predicate type is represented as well by using the surrogate class. The surrogate is used only to handle the predicate instances. The surrogate class is shown with a loop link to show that it has the same property name which points to itself. The real snippets of the example from the implementation with Protégé, as shown above, is the proof that AP-RDF can be directly used in the practical application.

(i). The property *:set* is composed as a property.
(ii). The property *:set_1* is formed as well as the instance of the property *:set*
(iii). The surrogate class *:set* is composed and also the constraint of the equivalency with the property *:set*
(iv). Then, the instance of property can be used to generate the attribute of property.

It is also vital to measure the model's genuine performance as a new serialization model. The semantic interpretation of AP-RDF supports the model's performance. At glance, the Property Instance helps to perform the function of a predicate. Therefore, the question analysis works better compared to the default RDF. The predicate instance which represents property in the real world cuts the matching process time. The result of the experiment is shown in Figure 6. The ten queries over the World bank data-set formed in AP-RDF. The Q1 until Q6 are simple queries and the rest are complex queries. It shows that complex queries which usually involve more triples return much better results. Meanwhile, the results are fairly obtained for simple queries. Those queries are as below:

Q1: Please state the indicators of Topic <t>that have increased in the country during the last four years until 2012.
Q2:Indicators for all subjects that increased in the last four years in all countries except the country <c>are listed.
Q3: Lists all countries except those in the Topic <t>and Indicator <i>that have increased in the last four years until 2012.
Q4: In the four years leading up to 2012, which country <c>indicators increased or decreased?
Q5: From four years to 2012, compile a list of countries that have declined in all indicators in topic <t>.
Q6: Indices for topics <t>that increased over four years to 2012, except CZE (Czech Republic), but fell in TUR (Turkey).
Q7: On the other hand, the indicator theme of agriculture in countries <c>fell for four years until 2012, while all indicators of another topic, except indicators that increased for four years until 2012, increased for four years. The names of the people are listed.
Q8: List of nations in the same condition in terms of whether the TUR (Turkey) topic <t>index of all these indicators <i>climbed or dropped in the four years leading up to 2012, except the CZE (Czech Republic), which increased.
Q9: A list of indicators <i>that declined in the CZE (Czech Republic) for the four years leading up to 2012 has been enlarged for all nations except TUR for the four years leading up to 2012. (Turkey).
Q10: On the other hand, until 2012, the indicator theme I of 5 nations <cA>, <cB>, <cC>, <cD>, <cE>increased for 4 years, while all other indicators of other themes are listed within 4 with all countries except these 5 countries. By 2012, the indicator's name has been shortened.

The other experiment's result is shown in Figure 7. This experiment focuses on comparing AP-RDF to the other recent model, Name Graph[4] and N-Quads[3]. It returns a similar result to the first experiment shown in Figure 6. AP-RDF performance surpasses the other model, especially for the complex query. The list of questions is below:

Q1: Please list all countries and their names
Q2: Please enter the subject and its indicator number.
Q3: Maximum indicator number of the topic
Q4: Middle and low-income countries in Europe and Central Asia Q5: A country in Central Europe, but more than 50% of the agricultural land.
Q6: All countries are displayed, and the topic <t>index has increased for 4 years until 2012.
Q7: Displays all countries and indicators of themes <t>that climbed for four years till 2012, however, these indicators declined in Japan at the same time.although they fell in Japan at the same time.
Q8: All nations are shown save the Czech Republic, and the theme <t>indicators climbed for four years until 2012,
Q9: Displays all Japanese subject indicators that have

increased or dropped over the last four years, up to 2012.
Q10: All indicators that rose in the four years leading up to 2012 are shown in Subject and Subject for Japan and Turkey.
Q11: Show all the indicators that have increased in the four years leading up to 2012 in the Topic of Japan and Turkey.
Q12: Show all the indicators in the topic that climbed in Turkey for four years till 2012, but fell in Japan at the same period.
Q13: Show how, in the four years leading up to 2012, all indicators of Topic and Topic increased and fell simultaneously in Japan.
Q14: Shows all indicators for the three subjects <tA>, <tB>, and <tC>that had increased for four years in Japan by 2012.
Q15: In the topic <t>, show the country and index name. In certain nations, the indicator declined for four years until 2012, whereas in others, it climbed for four years until 2012.

IV. Conclusions and Future Work

We demonstrated that AP-RDF meets the W3C Recommendation's default semantic interpretation. We demonstrated that AP-RDF meets the W3C Recommendation's default semantic interpretation. The RDF Interpretation is the bare minimum of normative semantic interpretations. The experimentals showed AP-RDF surpassed default RDF, NG-RDF and NQ-RDF. The work shortly will focus on the other key aspect of an AP-RDF formal model. The previous of our work has shown that the graph approach helps to understand complex information. Up to now, we assume that the formal model is related to the formal graph as well. The graph has a very strong root in math. Therefore, if an AP-RDF can be interpreted as well as a graph, we argue that more extensions of this work will be useful in some services. The middle-level future work is to study whether this model is useful in driving a new direction of information retrieval. The predicate in triples AP-RDF should be given more attention in the retrieval process. Next middle layer, we will evaluate the usefulness of what we have proposed especially in the reasoning process within some real applications. How the proposed concepts influence extended tasks, e.g., graph mining or linked data mining should be studied further in the future.

References

[1] W. W. W. Consortium and others, "RDF 1.1 Primer," World Wide Web Consortium, 2014.

[2] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 concepts and abstract syntax," W3C Recommendation, vol. 25, pp. 1–8, 2014.

[3] G. Carothers, "RDF 1.1 N-Quads: A line-based syntax for RDF datasets," W3C Recommendation, 2014.

[4] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler, "Named Graphs, Provenance and Trust," in Proceedings of the 14th International Conference on World Wide Web, New York, NY, USA, 2005, pp. 613–622.

[5] D. W. Wardani and J. Küng, "An Attributed Predicate RDF," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, 2015, pp. 410–416.

[6] F. Manola, E. Miller, and B. McBride, "RDF primer. W3C recommendation," World Wide Web Consortium, 2004.

[7] S. Das, J. Srinivasan, M. Perry, E. I. Chong, and J. Banerjee, "A Tale of Two Graphs: Property Graphs as RDF in Oracle.," in EDBT, 2014, pp. 762–773.

[8] P. Hayes and P. F. Patel-Schneider, "RDF 1.1 Semantics," W3C recommendation, vol. 25, 2014.

[9] M. Arenas, E. Prud'hommeaux, and J. Sequeda, Direct mapping of relational data to RDF. W3C Working Draft 24 March 2011. .

[10] C. Gutierrez, C. Hurtado, and A. O. Mendelzon, "Foundations of Semantic Web Databases," in Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, NY, USA, 2004, pp. 95–106.

[11] D. Wardani and J. Küng, "Property Hypergraphs as an Attributed Predicate RDF," in On the Move to Meaningful Internet Systems: OTM 2015 Conferences, vol. 9415, C. Debruyne, H. Panetto, R. Meersman, T. Dillon, G. Weichhart, Y. An, and C. A. Ardagna, Eds. Springer International Publishing, 2015, pp. 329–336

[12] D. W. Wardani and J. Küng, "An Attributed Predicate RDF," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, Oct. 2015, pp. 410–416. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.58.

[13] D. Wardani, "The Evaluation of Semantic Mapping," J. Phys. Conf. Ser., vol. 1500, p. 012101, Apr. 2020, doi: 10.1088/1742-6596/1500/1/012101.

[14] D. W. Wardani and J. Küng, "THE EFFECTIVENESS OF THE SEMANTIC MAPPING RELATIONAL TO GRAPH MODEL," pp. 107–114, 2016.

[15] D. W. Wardani, "Rich-link oriented graph model for improving query performance," in 2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Oct. 2017, pp. 52–56. doi: 10.1109/IC3INA.2017.8251739.

[16] D. W. Wardani and J. Küng, "Semantic mapping relational to graph model," in Computer, Control, Informatics and Its Applications (IC3INA), 2014 International Conference on, Oct. 2014, pp. 160–165. doi: 10.1109/IC3INA.2014.7042620.

[17] D. W. Wardani and J. Küng, "Semantic Mapping Relational to a Directed Property Hypergraph Model," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, Oct. 2015, pp. 152–159. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.24.

[18] D. W. Wardani and J. Küng, "Mapping RDB TO RDF with Higher Semantic Abstraction," presented at the 15th International Conference on WWW/INTERNET 2016, Mannheim, Germany, Oct. 2016.

Dewi Wardani received her doctoral degree from Johannes Kepler University in December. She finished her master degree at National Cheng Kung University, ROC Taiwan. Now, she is a lecturer and a researcher at Informatics Department, Universitas Sebelas Maret, Surakarta, Central Java, Indonesia. Her research interest is information system especially in knowledge management with semantic web approach. Her research area is included all aspects of it. Either fundamental layer, the middle layer or upper layer - use cases. She also has interest in language computation which supports the semantics issue in knowledge management.