# JPoT: Just another Populator of TBoxes

**Jean-Rémi Bourguet**

Università degli Studi di Sassari (UNISS)
Sassari, RAS – Italia
Universidade Federal do Espírito Santo (UFES)
Vitória, ES – Brasil

## Abstract

The OWL2's populators are currently an important support to empirically evaluate the reasoners and/or to escort the practitioners by leveraging instantiations in their knowledge bases. In this paper, after having evoked the closest approach describing their strengths and weakness, we present Just another Populator of TBOX (**JPoT**). This purely syntactic and domain independent populator is based on a random process of concept, role and data instantiations guaranteeing consistency of knowledge bases founded on TBOXes expressed in $\mathcal{ALCQ}^{(\mathcal{D})}$. Moreover, we demonstrate that data instantiations can be tricky when the expectation of the modeler is to obtain a sound knowledge base able to pass the equivalent of a Turing test. Finally, we evaluate the performances of **JPoT**.

## 1 Introduction

The rise of the domain ontologies in OWL2 compelled more and more practitioners to make their data available (Shadbolt et al., 2006). To interface with this potential myriad of instantiations (involving concepts, abstract and concrete roles) they will have to deal with bit by bit, one of their expectations is to have some leverages helping them to test their solutions in the context of big data before deploying in the real world. One solution often considered by default is a semi-automated nay manual population dependent on the proper terminological axioms (named TBOX) of the ontologies. Nonetheless, these solutions have the disadvantages of being time-consuming and prone to inconsistency. Furthermore, some approaches may need to dispose such steps of population in order to empirically perform the reasoning tasks (Bourguet and Pulina, 2013). For example, this situation occurs in the community of conceptual modeling where it is a frequent practice to design model alternatives and perform an empirical comparison between them (Batsakis et al., 2009). Finally, the reasoners engineers could devote more attention for ontology populations in the future. For example, during the first edition of the OWL Reasoner Evaluation (ORE) in 2012, a method was proposed to generate a benchmark and effectively evaluate semantic reasoners by generating realistic synthetic semantic data (Li et al., 2012). According to the international academic publisher IGI-global, the ontology population is defined as the process of creating instances (named ABOX) for an ontology *usually* involving the linking of various data sources to the elements of the ontology. However, this domain has been constantly hackneyed like the task of recognizing the new elements that should go into a domain ontology (Bedini and Nguyen, 2007). In fact, an Ontology Populator can behave either as an Extracted Data Generator (EDG), or as a Synthetic Data Generator (SDG). The SDGs can be split in two sets: i-the domain dependent SDGs i.e. those performing only populations of a given domain ontology and ii-the domain independent SDGs i.e. those performing populations of specific expressive fragments of computable languages.

Our proposal is to create a tool that is Just another Populator of TBOX (**JPoT**) performing populations like a domain independent SDG and guaranteeing consistency of the knowledge base (that is defined as the union of the TBOX and the ABOX). The subsequent parts of the article are as follows: Section 2 presents the related works, Section 3 presents **JPoT**, Section 4 performs an evaluation and Section 5 explores some perspectives.

## 2 Related Works

Very few SDGs have been developed in order to perform some stress tests before deploying an ontology in a Semantic Web application.

According to the creators, the Lehigh University Benchmark (Guo et al., 2005) was the first knowledge base generator. The idea of LUBM was to feature a university domain ontology with one statically predefined TBox, and allow different sizes of an artificial generated dataset i.e. an ABox. A set of 14 different queries was included in the benchmark in order to evaluate the perfomance of reasoners by processing the predefined queries on the different generated knowledge bases. Due to the growing need to better profile the behaviour of an ontology with regards to differing numbers and complexities of the axioms in the TBox, an extension of LUBM, the so-called University Ontology Benchmark (UOB) has been introduced (Ma et al., 2006).

Another remarkable tool named OTAGen (Ongenae et al., 2008) has the specificity to generate complete knowledge bases providing the capibility of specifying a large range of parameters characterising them, both on TBox as well as ABox level, the tool can also generate some corresponding queries.

Note that after this proposal, another approach proposed a purely TBox generation with different reasoning complexities resulting from the relative proportions of the design patterns of biomedical structures representation (Boeker et al., 2011).

Finally, the only approach that can handle a lack or inaccessibility of data in ABoxes when a TBox is already available is SKTI - a synthetic data generator (Chowdhury, 2012). This system generates synthetic instances based on a source ontology and user specifications. Note that to mimic the real world scenario the system also allows the insertion of noisy and erroneous instances into the dataset.

This last solution is the closest related work with **JPoT** as a domain independent SDG that can populate TBoxes provided by users. Next, we will describe the heuristics **JPoT** follows to populate these TBoxes by guaranteeing consistency.

## 3 Synthetic Data Generation

Formally, every Description Logic is based on some finite sets: a set $\mathbf{C_A}$ of concepts names, a set $\mathbf{D_T}$ of datatype names, a set $\mathbf{R_A}$ of abstract role names and a set $\mathbf{R_T}$ of concrete role names. Baader and Nutt introduced the notion of interpretation in first-order logic (Baader and Nutt, 2003).

Note that the definition we introduce below is an adaptation of the interpretation presented in the OWL2 direct semantic[1].

**Definition** (Interpretation). *An interpretation $\mathcal{I}$ is a tuple $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \Delta_{\mathcal{D}}, \cdot^{\mathcal{I}} \rangle$ where:*
*- $\Delta^{\mathcal{I}}$ is the domain, i.e. a set of individuals,*
*- $\Delta_{\mathcal{D}}$ is a data-type domain disjoint with $\Delta^{\mathcal{I}}$,*
*- $\cdot^{\mathcal{I}}$ is the interpretation function which maps:*
  *- each $A \in \mathbf{C_A}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,*
  *- each $r \in \mathbf{R_A}$ to a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,*
  *- each $D \in \mathbf{D_T}$ to a value space $D^{\mathcal{I}} \subseteq \Delta_{\mathcal{D}}$,*
  *- each $t \in \mathbf{R_T}$ to a relation $t^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{D}}$.*

**JPoT** can actually parse and populate guaranteeing consistency the expressive power of the fragment $\mathcal{ALCQ}^{(\mathcal{D})}$ described below:

$$\begin{aligned}
\mathbf{R} &::= \top_{\mathbf{R}} \mid \mathbf{R_A} \mid \mathbf{R_T} \\
\mathbf{D} &::= \mathbf{C} \mid \mathbf{D_T} \\
\mathbf{C} &::= \mathbf{C_A} \mid \neg\mathbf{C} \mid \top_{\mathbf{C}} \mid \bot_{\mathbf{C}} \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \\
&\quad \exists \mathrm{R}.\mathbf{D} \mid \forall \mathrm{R}.\mathbf{D} \mid \leq \mathrm{nR}.\mathbf{D} \mid \geq \mathrm{nR}.\mathbf{D} \mid = \mathrm{nR}.\mathbf{D}
\end{aligned}$$

**Definition** (TBox). *Given the concepts $C, D \in \mathbf{C}$, we call a* TBox *a finite set of i-concept equivalences i.e. 'C ≡ D' or ii-concept inclusions, i.e. 'C ⊑ D'. An interpretation $\mathcal{I}$ is a model of a* TBox *iff for all the axioms $\varphi \in$ TBox, $\mathcal{I} \Vdash \varphi$, with:*
  *- $\mathcal{I} \Vdash (C \sqsubseteq D)$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*
  *- $\mathcal{I} \Vdash (C \equiv D)$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$*

**JPoT** deals only with unequivocal TBox (Baader and Nutt, 2003) as a finite set of equivalence or inclusions for which the left-hand side of each axiom is an atomic concept and for every atomic concept there is at most one axiom where it occurs on the left-hand side.

**Definition** (ABox). *Given $a, b \in \Delta^{\mathcal{I}}$, $d \in \Delta_{\mathcal{D}}$, $C \in \mathbf{C}$, $r \in \mathbf{R_A}$, $t \in \mathbf{R_T}$, we call an* ABox *a finite set of class, role or data assertions. An interpretation $\mathcal{I}$ is a model of an* ABox *$\mathcal{A}$ iff for all the assertions $\varphi \in \mathcal{A}$, $\mathcal{I} \Vdash \varphi$, with:*
  *- $\mathcal{I} \Vdash C(a)$ iff $a \in C^{\mathcal{I}}$* Concept Assertion
  *- $\mathcal{I} \Vdash r(a, b)$ iff $(a, b) \in r^{\mathcal{I}}$* Role Assertion
  *- $\mathcal{I} \Vdash t(a, d)$ iff $(a, d) \in t^{\mathcal{I}}$* Data Assertion

[1] https://www.w3.org/TR/owl2-direct-semantics/

The set of all the concept, role and data assertions are respectively denoted $A_c$, $A_r$ and $A_d$. **JPoT** is designed to populate TBox in function of

i - a number **n** of potential individuals[2]

$$\text{n}=|\Delta^{\mathcal{I}}|$$

ii - a total number **m** of assertions

$$\text{m}=|A_c \cup A_r \cup A_d|$$

iii - a ratio $\tau$ of the number of concept assertions on the total number of assertions

$$\tau = \frac{|A_c|}{|A_c \cup A_r \cup A_d|}$$

iv - a ratio $\rho$ of the number of role assertions on the number of data assertions

$$\rho = \frac{|A_r|}{|A_r \cup A_d|}$$

### 3.1 Concept Assertions

The process of concept assertions initiates with the computation of the set of all the disjoint class denoted $\mho_T$ such that $\mho_T = \{\{C, D\} | C \sqsubseteq^+ \neg D\}$ with $\sqsubseteq^+$ is the transitive closure of $\sqsubseteq$.

**Algorithm 1:** Concept Assertions (CAs)

```
Data: TBox, m, τ
Result: round(m · τ) CAs
x= 0;
while x ≤ round(m · τ) do
    δ_i = Draw(Δ^I);
    Θ=FALSE;
    while ¬Θ do
        C_j = Draw(C);
        if {{C, C_j}|δ_i ∈ C^I} ∩ ℧_T = ∅;
        then
            Θ=TRUE;
        end
    end
    C_j(δ_i);
    x++;
end
```

Next, each individual drawn will instantiate the first concept drawn that is not disjointed with the concepts the individual already instantiates. The Algorithm 1 describes the concept assertions phase of **JPoT**. Note that, the function Draw$(S)$ returns a randomly drawn element from a set $S$.

[2]Here "potential" means the possibility to have non drawn individuals that are not involved in any assertion.

### 3.2 Role Assertions

**JPoT** dealing only with unequivocal TBox, an axiom $t_i$ s.t. $t_i \in$ TBox can be defined like that:
$t_i = \{\mathtt{A}_i \equiv \bigsqcap_l \mathtt{B}_l \bigsqcap_{j,k} \forall q_j.\mathtt{D}_k \bigsqcap_{o,p} \exists r_o.\mathtt{C}_p \bigsqcap_{u,v} \lessgtr s_u.\mathtt{E}_v\}$
with $\bigsqcap \in \{\bigsqcap, \bigsqcup\}$, $\equiv \in \{\sqsubseteq, \equiv\}$ and $\lessgtr \in \{=, \geq, \leq\}$

We introduce a rearrangement of TBoxes to gather a set of axioms with concepts, datatypes, roles and constructors. The claim of **JPoT** is to populate TBoxes respecting the consistency but due to the Open World Assumption (OWA), we treat indifferently intersection and union.

A derivation of a TBox is defined like that: TBox$' = \{\mathcal{M}(ti) | t_i \in \text{TBox}\}$ with
$\mathcal{M}(t_i) = \{\bigcup_{j,k}\{\mathtt{A}_i \triangleq \forall q_j.\mathtt{D}_k\} \bigcup_{o,p}\{\mathtt{A}_i \triangleq \exists r_o.\mathtt{C}_p\} \bigcup_{u,v}\{\mathtt{A}_i \triangleq \lessgtr s_u.\mathtt{E}_v\}\}$

During this stage, **JPoT** draws abstract roles and try to instantiate them. The Algorithm 2 uses the following heuristic: once an abstract role is drawn, an individual subject is drawn and the concepts it already instantiates are confronted to the domain of the abstract role. In case of disjointness another individual is drawn. After this step, the concepts of the chosen individual are confronted with the universal quantification axioms implying the abstract role in order to build a set of mandatory concepts for the individual object that has to be drawn in the next step. In the event the drawn individual i-has a concept in common with the mandatory ones, ii-doesn't have concepts disjoint with the range of the abstract and iii-doesn't violate the max cardinality restriction, this object individual will participate to the instantiation with the abstract role and the subject individual. In case of a violation of the cardinality restriction, another abstract role will be drawn and the process of selection will restart. For example, in the TBox axiom "`Scholarship` $\sqsubseteq$ $\exists$managedBy.`Referrer` $\bigsqcup$ $\exists$supervisedBy.`Organisation`", the algorithm will randomly choose one of the element of the disjunction either involving the management or the supervision (or the both if the scholarship is drawn again). Moreover, in the axiom "`Scholarship` $\sqsubseteq$ $\forall$remunerates.`Researcher` $\bigsqcap$ $\exists$providesBy.`Organisation`", the algorithm will either ensure that if a scholarship remunerates someone it will be a researcher or suggest (in absence of an axiom concerning a range) that a scholarship is provided by an organisation.

**Algorithm 2:** Role Assertions (RAs)

**Data**: TBox, TBox′, m, $\tau$, $\rho$
**Result**: round$(m \cdot (1-\tau) \cdot \rho)$ RAs
y= 0;
**while** $y \leq$round$(m \cdot (1-\tau) \cdot \rho)$ **do**
  $\Xi$ =False;
  **while** $\neg\Xi$ **do**
    $\Xi$ =True;
    $r_k$ = Draw($\mathbf{R_A}$);
    $P_\exists = \{(C,D) | C \triangleq \exists r_k.D\}$;
    $(A,B)$ = Draw($P_\exists$);
    **if** domain$(r_k) = \emptyset$ **then** $I = \{A\}$;
    **else** $I$ = domain$(r_k)$;
    $\Theta$=False;
    **while** $\neg\Theta$ **do**
      $\delta_i$ = Draw($\Delta^\mathcal{I}$) s.t. $\exists C.\ \delta_i \in C^\mathcal{I}$;
      **if** $\{\{C,I\} | \delta_i \in C^\mathcal{I}\} \cap \Omega_T = \emptyset$
      **then** $\Theta$=True;
    **end**
    $P_\forall = \{(C,D) | C \triangleq \forall r_k.D\}$;
    **foreach** $c \in \{C | \delta_i \in C^\mathcal{I}\}$ **do**
      **if** $\exists (E,F) \in P_\forall$ *s.t.* $c \sqsubseteq E$ **then**
      $F \in W$;
    **end**
    **if** range$(r_k) = \emptyset$ **then** $J = \{B\}$;
    **else** $J$ = range$(r_k)$;
    $\Theta$=False;
    **while** $\neg\Theta$ **do**
      $\delta_j$ = Draw($\Delta^\mathcal{I}$) s.t. $\exists C.\ \delta_j \in C^\mathcal{I}$;
      **if** $\{\{C,J\} | \delta_j \in C^\mathcal{I}\} \cap \Omega_T = \emptyset$
      AND $W \subseteq \{C | \delta_j \in C^\mathcal{I}\}$ **then**
      $\Theta$=True;
    **end**
    $T$=$\{\langle C,D,l\rangle | C \triangleq \leq l r_k.D \vee C \triangleq = l r_k.D\}$;
    $l = 0$;
    **foreach** $c \in \{C | \delta_i \in C^\mathcal{I}\}$ **do**
      **if** $\exists \langle E,F,L\rangle \in T$ *s.t.* $c \sqsubseteq E$ **then**
        **foreach**
        $d \in \{D | \delta_h \in D^\mathcal{I} \wedge (\delta_i, \delta_h) \in r_k\}$
        **do**
          **if** $d \subseteq F$ **then** $l$++;
        **end**
        **if** $l = L$ **then** $\Xi$ =False;
      **end**
    **end**
  **end**
  $(\delta_i, \delta_j) \in r_k$;
  y++;
**end**

### 3.3 Data Assertions

The populations performed by **JPoT** follow heuristics that guarantee consistencies of the knowledge bases only on the basis of the axioms in the TBoxes, in other words without any consideration of a specific domain. The produced Aboxes are somewhere semantically consistent because the population respects the semantic rules of the world present in the TBoxes. While the URIs of the individuals are all based on an integer in a selected range, the concepts they instantiate (concept assertions) and more the relation in which they are involved (role assertions) can represent an artificial but apparently real world. Let's imagine the equivalent of the Turing test for a SDG model in which the Aboxes should appear as real 70% of the time to succeed the test. Even if we didn't lead this experiment, we can objectively assume that a **JPoT**'s population implying only concept and role assertions could pass this test.

The test gets much more complicated to pass when **JPoT** has to deal with concrete roles due to a set of issues concerning the creation of data values. First, the usage of datatype has to be tackled with caution under a penalty of undecidability. owl2 solved this problem by recommending datatypes defining a datatype map[3] which lists the datatypes that can be used in the knowledge bases. Even restricting a population to this subset of datatypes, **JPoT** could fail a Turing test for SDG by strictly following the heuristic described in the Algorithm 3 due to a lack of semantic soundness in the generation of the data values for the data assertions. For example, let's imagine a TBox with the following axioms: each person has exactly one age that is an integer, an author has exactly one number of citations and hindex that are integers and if a person has a name it is always a string. As it is, **JPoT** could give an inhuman age for a person, assert a number of citations inferior to the hindex squared for a same author and provide an absurd name that would just be a random sequence of characters.

We implemented functions Gen$(D,t)$ generating data values in adequation with a datatype D and with a concrete role $t$ facing with different issues concerning the automatic population.

[3] https://www.w3.org/TR/owl2-syntax/#Datatype_Maps

In the following, we will describe three issues for this generation of data values corresponding to three kinds of data assertions illustrating through the concrete roles hasAge (3.3.1), citations/hindex (3.3.2) and hasName (3.3.3).

**Algorithm 3:** Data Assertions (DAs)

**Data**: TBox, TBox′, m, $\tau$, $\rho$
**Result**: round$(m \cdot (1-\tau) \cdot (1-\rho))$ DAs
z= 0;
**while** $z \leq$ *round*$(m \cdot (1-\tau) \cdot (1-\rho))$ **do**
  $\Xi$ =FALSE;
  **while** $\neg\Xi$ **do**
    $\Xi$ =TRUE;
    $t_k$ = Draw$(\mathbf{R_T})$;
    $P_\exists = \{(C,D) | C \triangleq \exists t_k.D\}$;
    $(A,B)$ = Draw$(P_\exists)$;
    **if** domain$(t_k) = \emptyset$ **then** $I = \{A\}$;
    **else** $I$ = domain$(t_k)$;
    $\Theta$=FALSE;
    **while** $\neg\Theta$ **do**
      $\delta_i$ = Draw$(\Delta^\mathcal{I})$ s.t. $\exists C.\ \delta_i \in C^\mathcal{I}$;
      **if** $\{\{C,I\} | \delta_i \in C^\mathcal{I}\} \cap \Omega_T = \emptyset$
      **then** $\Theta$=TRUE;
    **end**
    $P_\forall = \{(C,D) | C \triangleq \forall t_k.D\}$;
    **foreach** $c \in \{C | \delta_i \in C^\mathcal{I}\}$ **do**
      **if** $\exists (E,F) \in P_\forall$ *s.t.* $c \sqsubseteq E$ **then**
      $F \in W$;
    **end**
    **if** range$(t_k) = \emptyset$ **then** $J = \{B\}$;
    **else** $J$ = range$(t_k)$;
    $d_j$ = Gen$(J, t_k)$;
    $T$=$\{\langle C,D,l\rangle | C \triangleq \leq l t_k.D \vee C \triangleq = l t_k.D\}$;
    $l = 0$;
    **foreach** $c \in \{C | \delta_i \in C^\mathcal{I}\}$ **do**
      **if** $\exists \langle E,F,L\rangle \in T$ *s.t.* $c \sqsubseteq E$ **then**
        **foreach**
        $d \in \{D | d_h \in D^\mathcal{I} \wedge (\delta_i, d_h) \in t_k\}$
        **do**
          **if** $d \subseteq F$ **then** $l$++;
        **end**
        **if** $l = L$ **then** $\Xi$ =FALSE;
      **end**
    **end**
  **end**
  $(\delta_i, d_j) \in t_k$;
  z++;
**end**

### 3.3.1 Using a facet space

The usage of facet spaces is the royal road for the modeler to obtain a knowledge base capable of passing the SDG Turing test. The facet space was introduced as a set of pairs of the form (F,v) where F is a constraining facet and v a constraining value. Each such pair is mapped to a subset of the value space of the datatype. Thus, the data range of concrete roles can be restricted using a *datatypeRestriction* which restricts the value space of a datatype by a constraining facet. In the example of hasAge, the modeler can use a *datatypeRestriction* that would restrict the datatype xsd:nonNegativeInteger by using a singleton facet space i.e. the pair (xsd:maxExclusive,"123"^^xsd:nonNegativeInteger) that corresponds to the limit for an age never reached in the history of the humanity. In addition in **JPoT**, we used a Gaussian distribution (with an expectation of 42 and a standard deviation of 10) in order to simulate an age distribution of researchers following a pyramidal shape.

### 3.3.2 Using a linear equation

Ensuring the soundness of data assertions restricting dataranges using *datatypeRestrictions* can still produce knowledge bases incapable of passing the SDG Turing test. In fact, one can say that the value of the subject of a data assertion has to be an integer less than 123, but one cannot say that the value of the subject of one data assertion is less than that of another data assertion. In the example of citations and hindex, the modeler doesn't have the ability with the current expressivity of OWL2 to constraint the TBox with the fact that the total number of citations of an author is greater than the hindex-squared. A proposition of extension for OWL2[4] allows the expression of linear equations but not of polynomial equations.

### 3.3.3 Using an API

As we described for a numerical datatype, it is also possible to restrict dataranges using a *datatypeRestriction* for String with a pattern (well known as a regular expression). This expressivity can be usefull to generate knowledge bases capable of passing the SDG Turing test. For example, a model that would represent a social security number could use such a pattern.

[4] https://www.w3.org/2007/OWL/wiki/Data_Range_Extension:_Linear_Equations

But when the concrete role is hasName, even the usage of space facet doesn't prevent to produce knowledge ineligible for a SDG Turing test. In this case, the only solution is to use another generator or API to produce a soundness value. Then we used the API JaNaG (Java Name Generator) which is a random name generator based on a name fragment database that creates relatively reasonably sounding names from different cultures/influences.

## 4 Performances of JPOT

We performed an evaluation of the populator **JPOT** (the version of **JPOT** based on the OWL-API 4.1.0) using a simplified model of the scholar domain illustrated in Figure 1. We highlight here that the ontological choices made at this example are intuitive, but arguable. Our aim was to describe the performance of **JPOT**, not to propose an ontological analysis of the scholar domain.

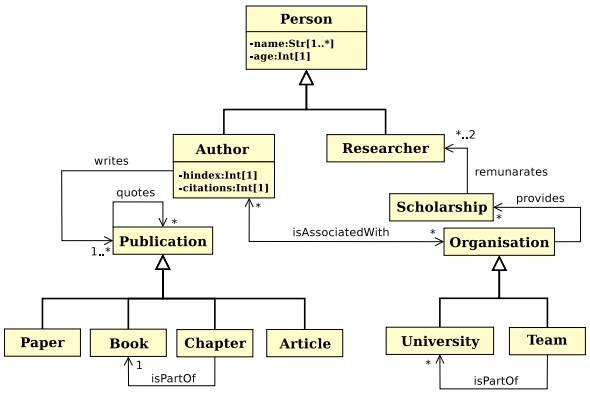



Figure 1: UML diagram of the scholar domain

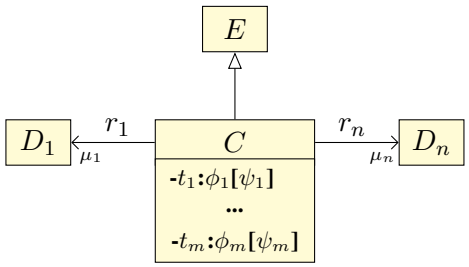
In this domain, Authors (which are Persons with names) write Publications, which can be classified into Papers, Articles, Chapter or Books. A Publication can be quoted by another Publication, and Authors have a total number of citations and an hindex. A Researcher can be remunerated by Scholarships provided by Organisations in such a way that an Organisation can provide several Scholarships, and a Scholarship can remunerate a maximum of two Researchers (e.g. organisations authorizing to change one time the "owner" of a scholarship). Authors and Organisations can be associated. There are two kinds of Organisations: Team and University. A Team can be part of Universities.

We introduce here a UML interpretation for the fragment $\mathcal{ALCQ}^{(\mathcal{D})}$. We denote $\mathbf{C_A}$ a set of concept names, $\mathbf{R_A}$ a set of abstract role names, $\mathbf{R_T}$ a set of concrete role names, $\mathbf{D_T}$ a set of datatype names and $S$ a set of symbols for Descriptions logics interpreted in first order logic (Baader and Nutt, 2003), $\Omega$ a set of UML cardinalities and a function $\ell(\Omega \rightarrow S)$ such that $\ell(*) \mapsto \forall$, $\ell(n) \mapsto \forall_{=n}$, $\ell(*..n) \mapsto \forall_{\leq n}$ and $\ell(n..*) \mapsto \forall_{\geq n}$ (with $n > 0$).

Note that we use the following notations:

- $\forall_{=n} r.C \equiv \forall r.C \sqcap {=}nr.C$
- $\forall_{\geq n} r.C \equiv \forall r.C \sqcap {\geq}nr.C$
- $\forall_{\leq n} r.C \equiv \forall r.C \sqcap {\leq}nr.C$

**Definition** (UML Interpretation U)**.** *Let* $\{C, D_1, \ldots, D_n, E\} \subseteq \mathbf{C}$, $\{r_1, \ldots, r_n\} \subseteq \mathbf{R_A}$, $\{t_1, \ldots, t_m\} \subseteq \mathbf{R_T}$, $\{\phi_1, \ldots, \phi_m\} \subseteq \mathbf{D_T}$ *and* $\{\mu_1, \ldots, \mu_n, \psi_1, \ldots, \psi_m\} \subseteq \Omega$*:*




$$(C \sqsubseteq E \sqcap \prod_{i=1}^{n} \ell(\mu_i) r_i.D_i \sqcap \prod_{j=1}^{m} \ell(\psi_j) t_j.\phi_j)^{\mathrm{U}}$$

Our empirical analysis has been performed on a machine equipped with an Intel Core at 3.30GHz and Ubuntu 15.04. We ran the Java-based reasoner Pellet 2.4.0 with Sun Java 1.8, and we set the maximum heap space to 7.5 GB. We populated the Tbox stemming from the UML interpretation of the Figure 1. We performed all the populations with $n$ and $m$ equal and with $\tau$ = 0.5 meaning in other words that the half of the assertions were concept assertions. For each pair $(n, m)$, we performed three populations: i- with $\rho$ = 0, ii- with $\rho$ = 0.5 and iii- with $\rho$ = 1.

Figure 2 shows the required CPU times of the populations and the consistency tasks. We used a range of total number of potential individuals and assertions going up to around one million.

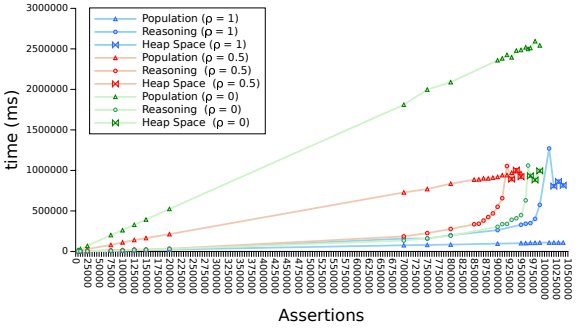



Figure 2: Some metrics about **JPoT**'s populations

Pellet output systematically that the ABOXes were consistent with the TBOX when the consistency checks were possible. After a certain amount of assertions, the reasoning tasks were impossible to conclude due to a heap space error thrown whenever the JVM reached the heap size limit. For $\rho = 0.5$ at $n, m \approx 930.000$, the first limit was detected. The same limit appeared for $\rho = 0$ at $n, m \approx 970.000$ and for $\rho = 1$ at $n, m \approx 1.020.000$. We can observe on the Figure2, the linear evolution of CPU times for the populator **JPoT** and the heap space limits (marked with bow ties) concerning the consistency checks of Pellet.

## 5 Conclusion

In this paper, we just presented another populator of TBOX: **JPoT**[5]. To the best of our knowledge, this is the first domain independent SDG guaranteeing consistency of the knowledge base founded on TBOXes expressed in $\mathcal{ALCQ}^{(\mathcal{D})}$. The issues of the data assertions were tackled and some performances presented. For future work, we intend that **JPoT** deals with an higher expressiveness to cover the whole fragment $\mathcal{SROIQ}^{(\mathcal{D})}$.

## Acknowledgments

This work has been made possible by "la Regione Autonoma della Sardegna e Autorità Portuale di Cagliari con L.R. 7/2007, Tender 16 2011, CRP-49656 con il projeto: Metodi innovativi per il supporto alle decisioni riguardanti l'ottimizzazione delle attività in un terminal container" and by "o EDITAL FAPES/CAPES N° 009/2014 (Bolsa de fixacão de doutores) com a proposta: Melhor integração de tecnologias de representação de conhecimento e raciocínio nas utilizações local e Web".

[5] **JPoT** is available at http://bit.ly/2vh5YE4

## References

Franz Baader and Werner Nutt. 2003. *Basic Description Logics*, Cambridge University Press, pages 43–95.

Sotiris Batsakis, Euripides Petrakis, Ilias Tachmazidis, and Grigoris Antoniou. 2009. Temporal representation and reasoning in OWL2. *Semantic Web* (Preprint):1–20.

Ivan Bedini and Benjamin Nguyen. 2007. Automatic ontology generation: State of the art. *PRiSM Laboratory Technical Report. University of Versailles* .

Martin Boeker, Janna Hastings, Daniel Schober, and Stefan Schulz. 2011. A T-Box generator for testing scalability of owl mereotopological patterns. In Michel Dumontier and Mélanie Courtot, editors, *Proceedings of the 8th International Workshop on OWL: Experiences and Directions*. volume 796 of *CEUR Workshop Proceedings*.

Jean-Rémi Bourguet and Luca Pulina. 2013. FRaQuE: A framework for rapid query processing evaluation. In Samantha Bail, Birte Glimm, Rafael S. Gonçalves, Ernesto Jiménez-Ruiz, Yevgeny Kazakov, Nicolas Matentzoglu, and Bijan Parsia, editors, *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation*. volume 1015 of *CEUR Workshop Proceedings*, pages 53–60.

Nafisa Chowdhury. 2012. Ontoevaluator – SKTI Synthetic Data Generator Synthetic Data Generator. http://aimlab-server.cs.uoregon.edu/services/skti-datagen/.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* 3(2):158–182.

Yingjie Li, Yang Yu, and Jeff Heflin. 2012. Evaluating reasoners under realistic semantic web conditions. In Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz, editors, *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation*. CEUR-WS.org, volume 858 of *CEUR Workshop Proceedings*.

Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, and Shengping Liu. 2006. Towards a complete OWL ontology benchmark. In *European Semantic Web Conference*. Springer, pages 125–139.

Femke Ongenae, Stijn Verstichel, Filip De Turck, Tom Dhaene, Bart Dhoedt, and Piet Demeester. 2008. OTAGen: A tunable ontology generator for benchmarking ontology-based agent collaboration. In *32nd Annual IEEE International on Computer Software and Applications*. IEEE, pages 529–530.

Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. 2006. The semantic web revisited. *IEEE intelligent systems* 21(3):96–101.