AN ONTOLOGY FOR CERAMICS CATALOGUING AND RELATED REASONING TASKS

Domenico Cantone, Cristiano Longo, Marianna Nicolosi-Asmundo, Daniele Francesco Santamaria
*Department of Mathematics and Computer Science University of Catania**

Si ringrazia il "Progetto Pon Ricerca e Competitività 2007-2012 – PRISMA"

Abstract

We briefly outline the work developed in [4], namely the definition of the OWL 2 ontology Ontoceramic for cataloguing ceramics, the analysis of the classification of Ontoceramic with some of the most widespread OWL reasoners and of its expressivity with respect to the principal existing OWL 2 profiles. Ontoceramic has been defined in collaboration with archeological experts as a first step to overcome the problem of efficiently mechanize the task of correctly cataloguing ceramics for the purpose of making such knowledge easily retrievable by scientists and researchers in the field. Currently, in fact, classification of ceramics is performed by using traditional methods like hard-copy archives and standard digital techniques like relational data-bases. The task of classification of Ontoceramic brought to light the limits of each of the reasoners used for the purpose. For example, we found out that Hermit was able to reason about all of the constructs occurring in the ontology but it did not fully support reasoning on data-type operations. Pellet, on the other hand, was capable to completely reason with Ontoceramic as long as the ontology was deprived of some of its constructs. For the purpose of studying and addressing some of the weaknesses of the reasoners we used, our first step was to analyze the expressiveness of Ontoceramic. Thus, we defined a logical model of Ontoceramic. Since none of the existing OWL 2 profiles coincides with our model, that is, each existing OWL 2 profile resulted to be much expressive and to not support some of the constructs occurring in Ontoceramic, we defined a new OWL 2 profile called OWL 2 SS. Our new profile contains all the features of the logical model of Ontoceramic that include a wide subset of OWL 2 constructs [9]. To ascertain the computational complexity of the reasoning problems of the OWL 2 SS profile and to define an efficient reasoning algorithm for it, we expressed it in set theoretical terms as a fragment of the four level stratified syllogistic called 4LQSR (Restricted Four Level Quantified Syllogistic), proved to be decidable in [2]. Thus, we singled out a sub-fragment of 4LQSR, called SSOWL, capable to express the profile OWL 2 SS, and adapted to SSOWL the decision procedure described in [2]. It turns out that under certain conditions, for 4LQSR the satisfiability problem for SSOWL is NP-complete.

References

1. Cycorp, Inc. Cyc. http://www.cyc.com/.
2. D. Cantone, M. Nicolosi-Asmundo. On the Satisfuability Problem for a 4-level Quantified Syllogistic and Some Applications to Modal Logic. Fundam. Inform., 124(4):427-448, 2013.
3. HP Lab. Apache jena. http://jena.apache.org/.
4. D. F. Santamaria. A semantic web ontology for ceramics cataloguing and set-theoretical representation for owl 2 profiles. Tesi di laura magistrale, University of Catania, Department of Mathematics and Computer Science, 2014.
5. R. Shearer, B. Motik, and I. Horrocks. Hermit: A highly-efficient OWL reasoner. In Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions,collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008, 2008.
6. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. J. Web Sem., 5(2):51{53, 2007.
7. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings, pages 292{297, 2006.
8. World Wide Web Consortium (W3C). SWRL: A semantic web rule language. http://www.w3.org/Submission/SWRL/.
9. World Wide Web Consortium (W3C). OWL 2 web ontology language structural specification and functional-style syntax (second edition). http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

$\mathrm{SS}_{\mathrm{OWL}}$

$\mathtt{SS}_{\mathtt{OWL}}$ presents four collections of variables, denoted by $\mathcal{V}_0$, $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$, such that:

- $\mathcal{V}_0$ contains variables of sort 0 denoted by $x, y, z, ..., .$
- $\mathcal{V}_1$ contains variables of sort denoted by $X^1, Y^1, Z^1, ...,$
- $\mathcal{V}_2$ contains variables of sort 2 denoted by $X^2, Y^2, Z^2...,$
- $\mathcal{V}_3$ contains variables of sort 3 denoted by $X^3, Y^3, Z^3...$.

Terms of sort 2 include variables of sort 2 and pair terms of the form $\langle x, y\rangle$ for $x, y \in \mathcal{V}_0$.
$\mathtt{SS}_{\mathtt{OWL}}$ quantifier-free atomic formulae are classified as:

- level 0: $x = y$, $x \in X^1$, for $x \in \mathcal{V}_0$, $y \in \mathcal{V}_0$, $X^1 \in \mathcal{V}_1$;
- level 2: $\langle x, y\rangle \in X^3$, $Z^2 \in X^3$, $\langle x, y\rangle = Z^2$, with $x \in \mathcal{V}_0$, $y \in \mathcal{V}_0$, $Z^2 \in \mathcal{V}_2$ and $X^3 \in \mathcal{V}_3$;

$\mathtt{SS}_{\mathtt{OWL}}$ purely universal formulae are classified as:

- level 1: $(\forall z_1)...(\forall z_n)\varphi_0$, where φ_0 is any propositional combination of quantifier-free atomic formulae and $z_1, ..., z_n$ are variables of sort 0.
- level 3: $(\forall Z^2)\varphi_1$, where $Z^2 \in \mathcal{V}_2$ and φ_1 is either any propositional combination of quantifier-free atomic formulae of levels 0 and 2 or it is of the form
  $(\forall Z^2)((Z^2 \in X^3) \leftrightarrow \neg(\forall z_1)(\forall z_2)\neg(\langle z_1, z_2\rangle = Z^2))$.

An $\mathtt{SS}_{\mathtt{OWL}}$ formula is any propositional combination of quantifier-free atomic formulae of levels 0 and 2 and of purely universal formulae of levels 1 and 3. Semantics of $\mathtt{SS}_{\mathtt{OWL}}$ is the same as the one presented in $4LQS$.

Example of Translation:

- $(\forall z)(z \in X^1_I \rightarrow (\neg\langle z, z\rangle \in X^3_{hasClass}))$
- $(\forall z, z_1)((z \in X^1_I \wedge z_1 \in X^1_I \wedge \langle z, z_1\rangle \in X^3_{hasHall}) \rightarrow \neg(\langle z_1, z\rangle \in X^3_{hasHall}))$

OWL 2-SS

ClassExpression := simpleClassExpression | supClassExpression | supClassExpression | equivalentClassExpression.

simpleClassExpression := ObjectIntersectionOf | DataIntersectionOf | ObjectUnionOf | DataUnionOf | ObjectComplementOf | ObjectOneOf | DataOneOf | EquivalentClasses | DisjointClasses | ObjectPropertyDomain | ObjectPropertyRange | DataPropertyDomain | DataPropertyRange.

supClassExpression := Class other than owl:Thing | ObjectAllValuesFrom | DataAllValuesFrom | ObjectMaxCardinality | DataMaxCardinality.

subClassExpression:= Class other than owl:Thing | ObjectSomeValuesFrom | ObjectHasValue | ObjectHasSelf | DataHasValue | ObjectMinCardinality | DataSomeValuesFrom | DataMinCardinality.

equivalentClassExpression := Class other than owl:Thing | simpleClassExpression.

subObjectPropertyExpression := ObjectPropertyExpression | propertyExpressionChain.

superObjectPropertyExpression := ObjectPropertyExpression.

ObjectIntersectionOf := simpleClassExpression | supClassExpression.

ObjectUnionOf := simpleClassExpression | supClassExpression.

DataUnionOf := simpleClassExpression | supClassExpression.

ObjectComplementOf := simpleClassExpression | supClassExpression.

DisjointClasses := simpleClassExpression | supClassExpression.

ObjectPropertyDomain := simpleClassExpression | supClassExpression.

DataPropertyDomain := simpleClassExpression | supClassExpression.

ObjectPropertyRange := simpleClassExpression | supClassExpression.

DataPropertyRange := simpleClassExpression | supClassExpression.

Ontoceramic

OntoCeramic is an ontology for classification and cataloguing of ceramics. Currently, classification of ceramics is performed by using traditional methods like hard-copy archives and standard digital techniques like relational database. OntoCeramic has been developed to solve the problem of efficiently classification and cataloguing of ceramics and to overcome the exponential complexity of ontological languages (i.e., OWL 2 DL) without sacrifice to much expressiveness. The ontology aims to cover different aspects of the classification and cataloguing problem. First at all, it is needed to identify unambiguously the location of the finding. One can have many locations to consider for a specific finding that can be organized in some particular hierarchy. The entire allowed relations path together with the object-properties required is shown in Figure 1. The double-hoop entities are optional, the others are mandatory. The figure indicates how locations must be defined for an object. Circles indicate OWL classes, arrows object-properties; the complete hierarchy of these object-properties is shown in Figure 2 (using the OWL Editor Protege). No other relations (or combinations) are allowed. Then, OntoCeramic has to handle all the components belonging to a specific object; for example, a cap can be found broken in three parts, each one requires a distinct description but they need to be associated together. Unlike the hard-copy version of ceramic catalogue, which does not manage directly the components of the finding but includes them in a descriptive field of the archive, every fragment in OntoCeramic is an entity. Each entity is associated to the object of provenance with the object-property "hasFragment", and, in particular, to one of its sub-properties. These sub-properties are intended to link each fragment to the correct provenance position of the object. The "hasFragment" property has as domain the class "Object" and as range "Type_Of_Fragment", instead each sub-property has as range the type of the fragment. "hasFragment" has an inverse property, "isFragmentOf", that has four sub-properties, one for each sub-property of "hasFragment" and defines their respective inverse. Some of these sub-properties are related exclusively to the "Lamp" object and in order to provide this constraint, in their domains is expressed only the "Lamp" object. Thanks to this structure it is possible to describe precisely every part of an object. All the possible components are described in a taxonomy having root in "Type_Of_Fragment" and shown in Figure 3, taken using the Protege Plugin Ontograf. Notice that "Foot" and "Bottom" have to be intended as equivalent. Figure 4 shows how properties about fragment managing are related each other. The class of an object is structured in a taxonomy as shown in Figure 5. To assign a class to an object the object-property "hasClass" is provided, having "Object" as domain and "Class" as range. Sample and sector of a finding are represented by the respective classes "Sample" and "Sector". Shape of an object is represented by the "Shape" class; the instances of these classes are associated to an object with, respectively, the "hasSample", "hasSector" and "hasShape" object-property. Every instance of class "Shape" can be unequivocally identified with a triple of properties: "hasFirstShapeDescriptor", "hasSecondShapeDescriptor", "hasThirdShapeDescriptor", which are sub-properties of "hasShapeDescriptor" data-properties. For each data-properties can be specified a string value: these values will be the keys for the "Shape" instances . OntoCeramic can specify for a finding its color using the Munsell Color System through the data-property "hasColor". This property provides three sub-data-property relations ("hasChroma", "hasHue", "hasValue") for Munsell chroma, hue and value, respectively. In addition one can provide a finding date and a general additional description using, respectively, "hasSiteDate" and "hasGeneralDescription" data-properties. As I explained above, it is possible to indicate every fragment of an object using the "Type_of_Fragment" taxonomy and the relative object-property. For each fragment one can specify its measurements. In particular, two data-properties are defined to indicate the thickness of the fragments. Such data-properties are sub-properties of "hasThickness", which is used for a general fragment. The first one, "hasWallThickness", is used to indicate the wall thickness of wall and foot fragment, the second, "hasBottomThickness", is used to indicate the bottom thickness of a foot fragment. It is possible to indicate if a fragment can be physically associated with another fragment to compose an unique object. In this case the "isFittedWith" object properties is provided. One can specify box and sheet of the hard-copy archive of the object description, using a "nonNegativeInteger" value in "hasBox" and "hasSheet" data property for the object indicating the number of the box and the number of the sheet of the object form, respectively. To face the shape question in the archaeological findings cataloguing the "Shape" taxonomy is provided as shown in Figure 6. Currently, OntoCeramic supports two type of "shape" and three type of "shape type", but one can add an arbitrary number of these classes and assert equivalence or dis-equivalence among them. As described above, there is no world-wide accord in using a specific nomenclature for the shape and type of an object. The taxonomy tries to resolve the question providing a class for each type of shape and many classes for each shape in order to distinguish their types; where required, an equivalence (or dis-equivalence) relation can be provided among the shape classes (or their sub-classes) to identify (or distinguish) shapes which are identical with respect to the classification system but which have been called with different names. For example, Figure 7 shows how "Lamboglia 1A" and "Hayes 8" are represented and identified.

OWL 2 -SS Comparison

Construct	EL	QL	RL	SS-OWL
ObjectSomeValuesFrom	Y	SUB to THING and SUP	SUB to THING	SUB
DataSomeValuesFrom	Y	Y	SUB	SUB
ObjectHasValue	Y	N	SUB	SUB
DataHasValue	Y	N	SUB	SUB
ObjectHasSelf	Y	N	SUB	SUB
ObjectAllValuesFrom	N	N	SUP	SUP
DataAllValuesFrom	N	N	SUP	SUP
Construct	**EL**	**QL**	**RL**	**SS-OWL**
ObjectMaxCardinality	N	N	SUP to 0/1	SUP
DataMaxCardinality	N	N	SUP to 0/1	SUP
ObjectMinCardinality	N	N	N	SUB
DataMinCardinality	N	N	N	SUB
ObjectExactCardinality	N	N	N	N
DataExactCardinality	N	N	N	N
ObjectOneOf	Y	N	SUB	Y
DataOneOf	Y	N	N	Y
ObjectInsersectionOf	Y	SUP	Y	Y
DataIntersectionOf	Y	Y	Y	Y
SubClassOf	Y	Y	Y	Y
EquivalentClasses	Y	Y	Class & Has-Value	Y
DisjointClasses	Y	Y	SUB	Y
ObjectUnionOf	N	N	SUB	Y
DataUnionOf	N	N	N	Y
DisjointUnion	N	N	N	Y
Construct	**EL**	**QL**	**RL**	**SS-OWL**
ObjectComplementOf	N	SUP	SUP	Y
SameIndividual	Y	N	N	Y
DifferentIndividuals	Y	Y	Y	Y
ClassAssertion	Y	Y	Y	Y
ObjectPropertyAssertion	Y	Y	Y	Y
DataPropertyAssertion	Y	Y	Y	Y
NegativeObjectPropertyAssertion	Y	Y	N	Y
NegativeDataPropertyAssertion	Y	Y	N	Y
SubObjectPropertyOf	Y	Y	Y	Y
SubDataPropertyOf	Y	Y	Y	Y
EquivalentObjectProperties	Y	Y	Y	Y
EquivalentDataProperties	Y	Y	Y	Y
TransitiveObjectProperty	Y	N	Y	Y
ReflexiveObjectProperty	Y	Y	Y	Y
FunctionalDataProperty	N	N	Y	Y
FunctionalObjectProperty	N	N	Y	Y
DisjointObjectProperties	N	Y	Y	Y
DisjointDataProperties	N	Y	Y	Y
IrreflexiveObjectProperties	N	Y	Y	Y
InverseObjectProperties	N	Y	Y	Y
InverseFunctionalObjectProperties	N	N	Y	Y
SymmetricObjectProperty	N	Y	Y	Y
AsymmetricObjectProperty	N	Y	Y	Y
ObjectPropertyDomain	Y	Y	SUP	Y
DataPropertyDomain	Y	Y	SUP	Y
ObjectPropertyRange	Y	Y	SUP	Y
DataPropertyRange	Y	Y	SUP	Y
HasKey	Y	N	SUP	SUP
ObjectPropertyChain	Y	N	N	Y

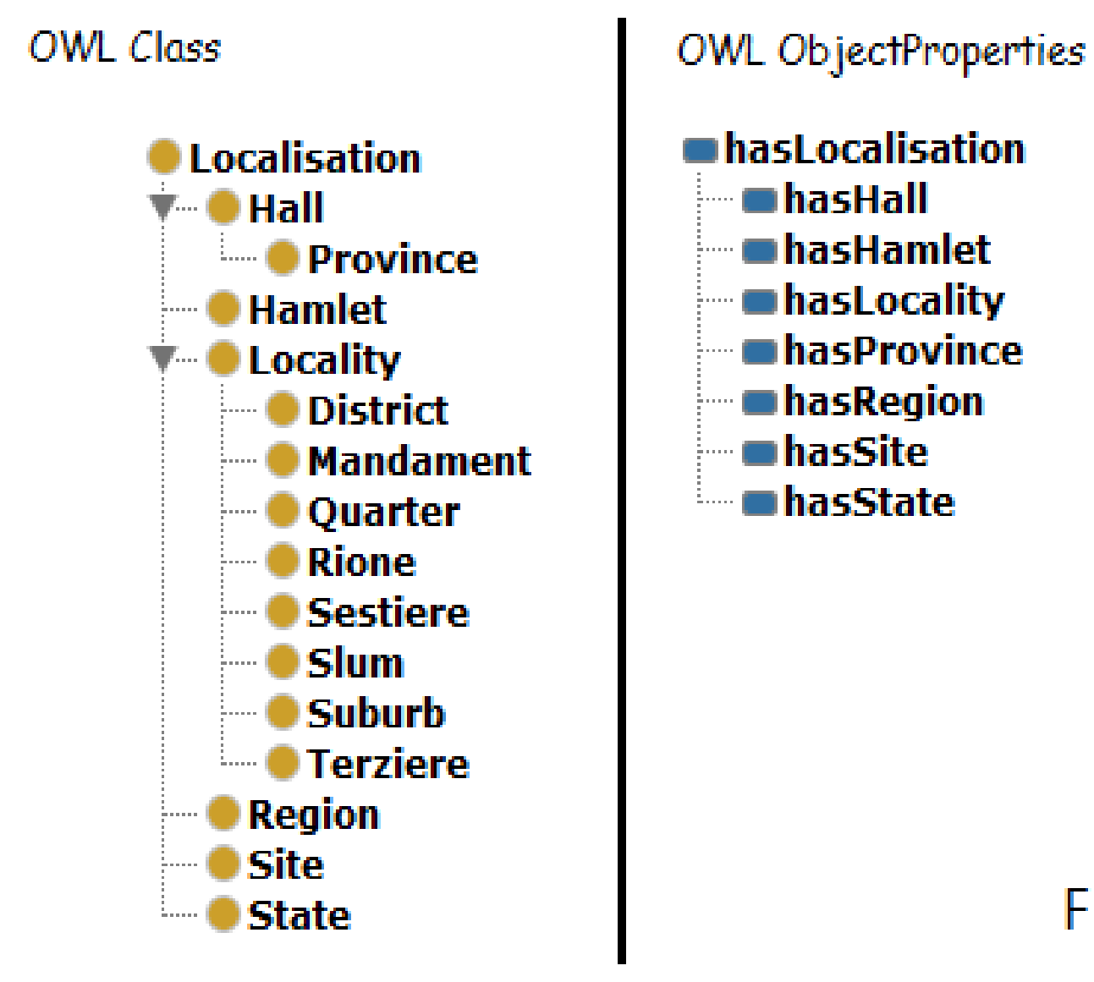

Fig.2

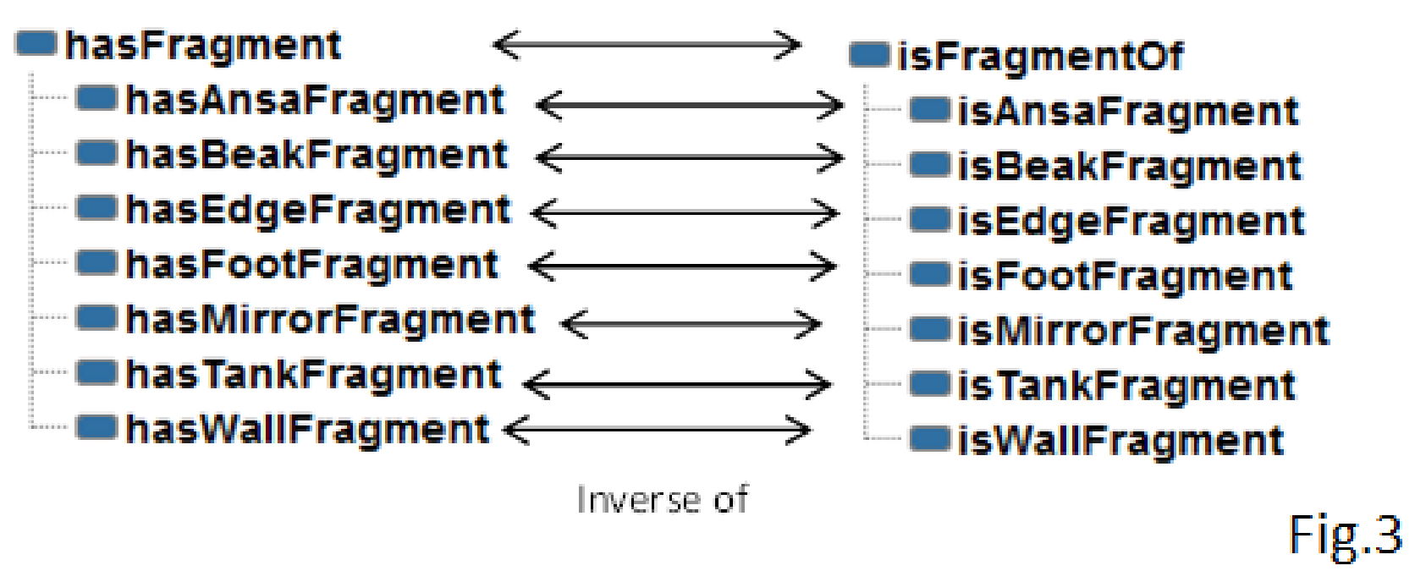

Fig.3

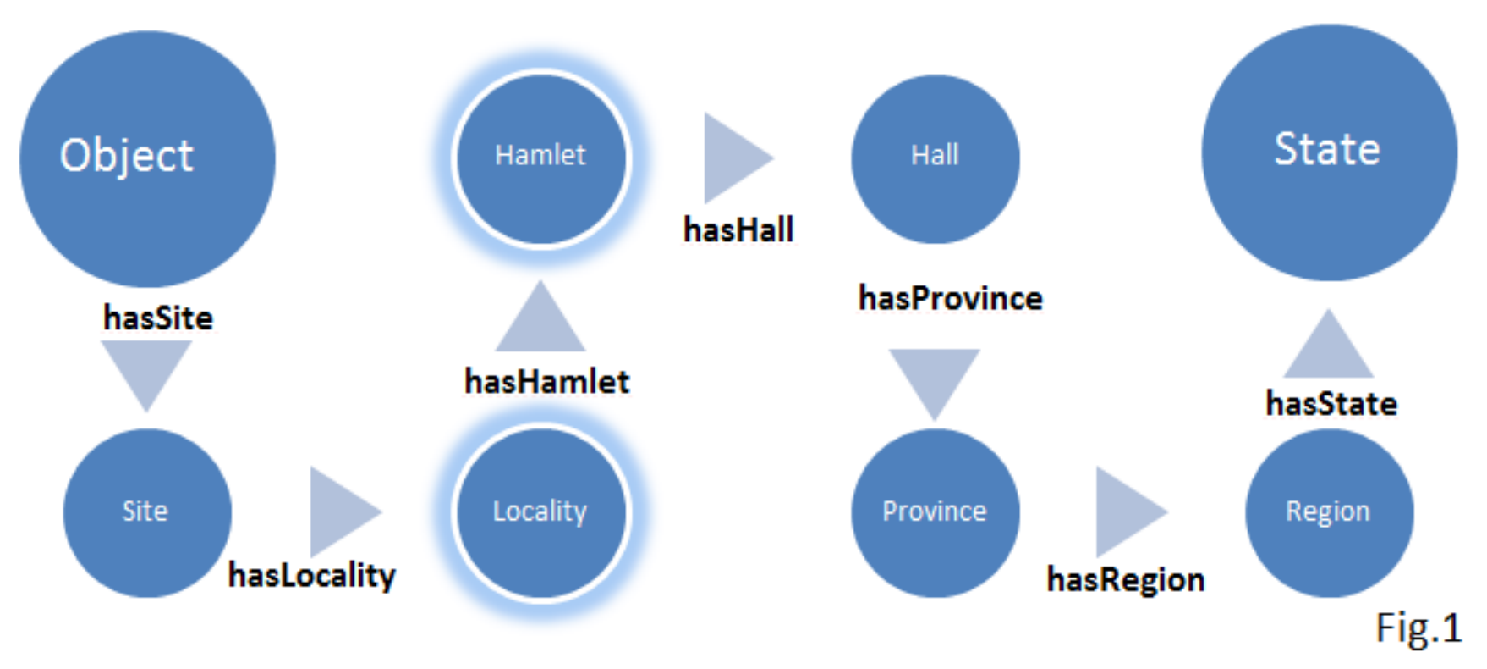

Fig.1

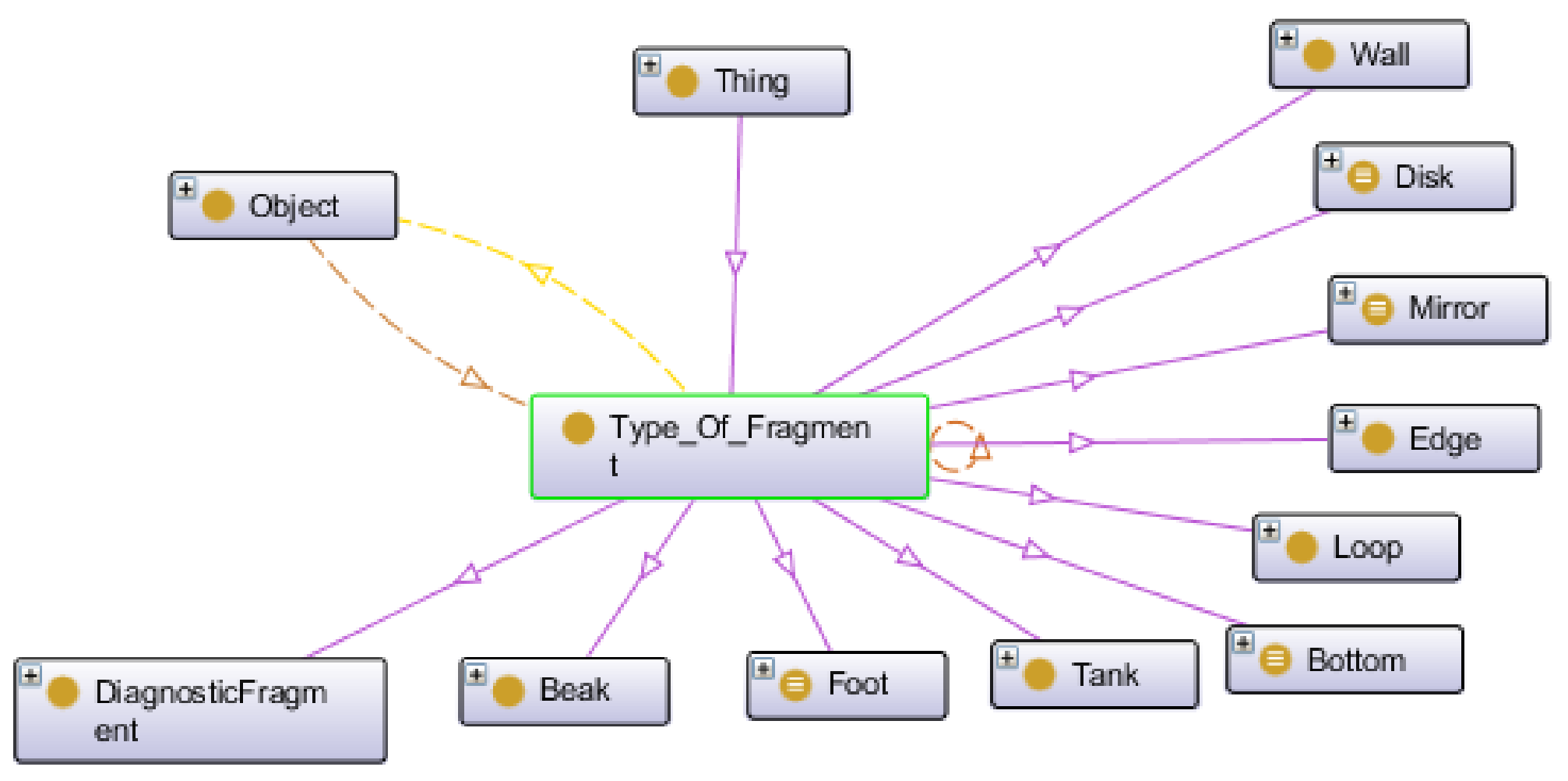

Fig.4

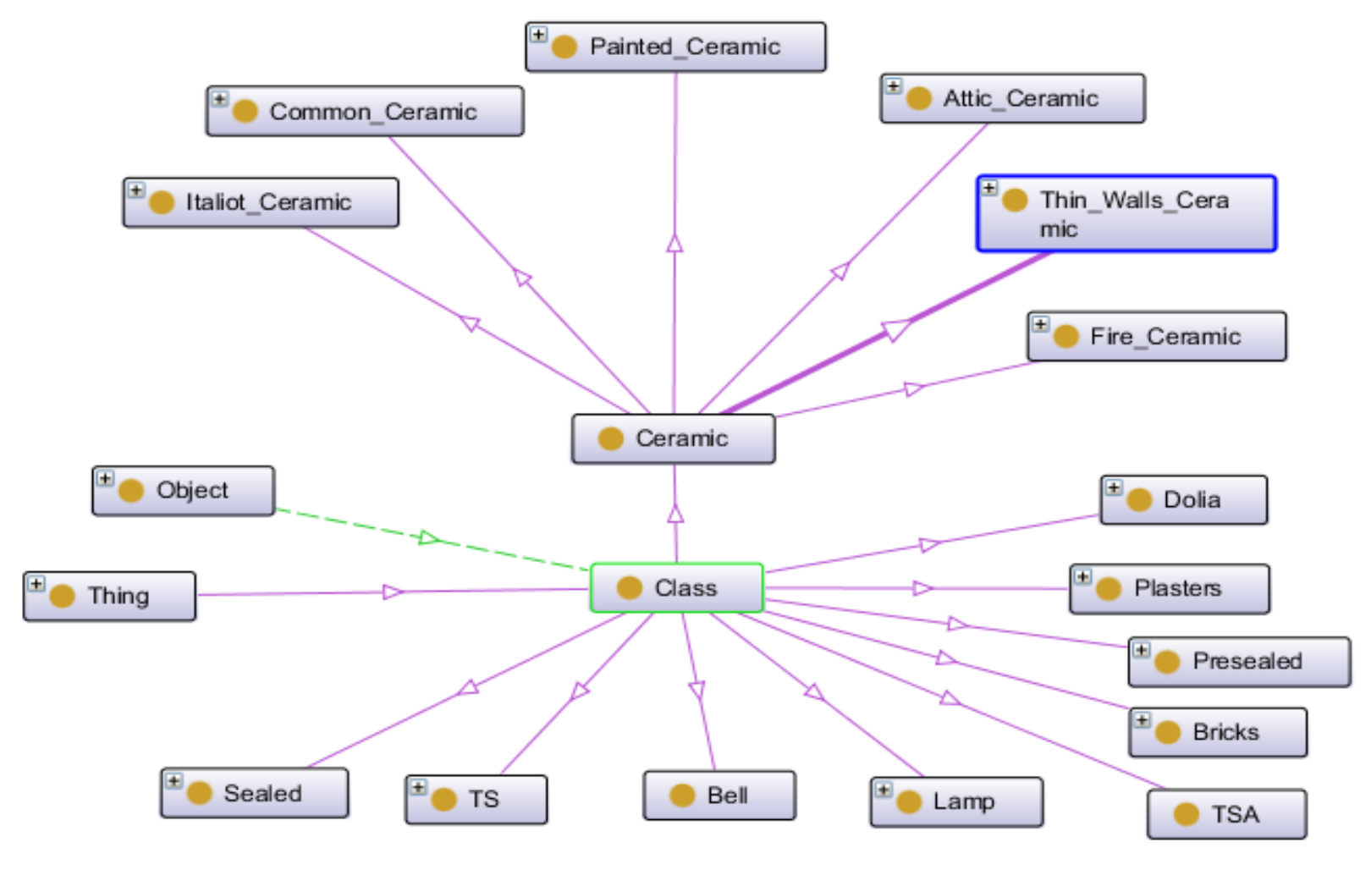

Fig.5

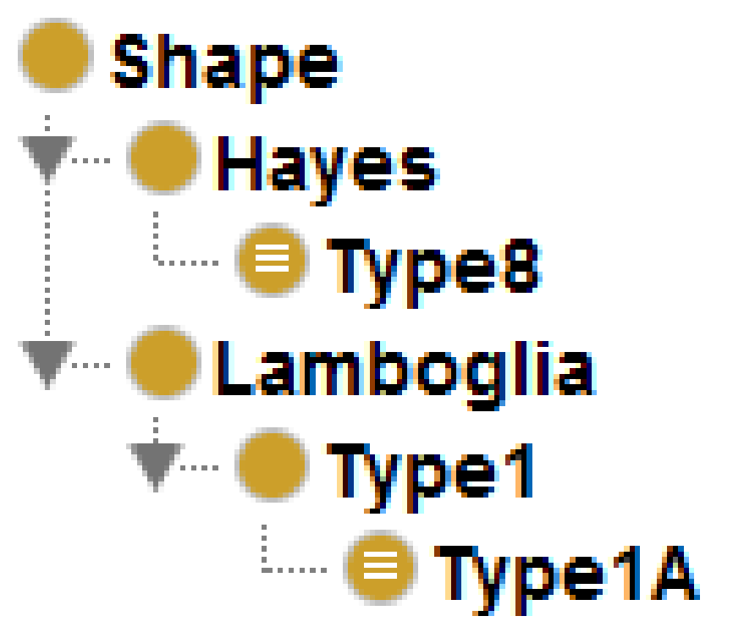

Fig.6