OCIP – An OntoClean Evaluation System Based on a Constraint Prolog Extension Language

Cleyton Mário de Oliveira Rodrigues[1,2], Frederico Luiz Gonçalves de Freitas[1], Ryan Ribeiro de Azevedo[1,3]

[1]Center of Informatics – Federal University of Pernambuco, UFPE
50.732-970, Recife-PE, Brazil

[2]FACETEG – University of Pernambuco, UPE
55.294-902, Garanhuns-PE, Brazil

[3]UAG – Rural Federal University of Pernambuco, UFRPE
55.292-270, Garanhuns-PE, Brazil

{cmor,fred,rra2}@cin.ufpe.br

Abstract. *An ontological model must evolve, since several and different knowledge sources can contribute to the addition of new concepts, relations and properties. Hence, we expect a certain level of quality during the engineering of ontologies, as well as the ontological commitment, in order to produce clear, well-formulated and correct subsumption relations. OntoClean is a methodology that addresses the creation of clean ontologies, i.e. the creation of taxonomic hierarchies to model properly the concepts in the domain. Due the lack of stable implementations in the literature, this paper presents OCIP: an OntoClean implementation in Constraint Handling Rules (CHR), a Constraint Programming Prolog extension.*

1. Introduction

Ontologies, in a higher level of abstraction, establish a common and unambiguous terminology for the domain in question. The idea of ontology is often restricted to what is called "formal ontology" [Guarino 1998]. This means that the content of an ontology is described using mathematical logic, which can provide computer system's ability of logical inference. You can also support autonomous discovery from recorded data, as well as reuse and exchange of knowledge. Recently the use of ontologies has been popularized through various other sub-areas of computer science, such as Software Engineering, Database and Information System.

OntoClean [Guarino and Welty 2000] [Welty and Guarino 2001], on the other hand, is a methodology that addresses the creation of clean ontologies, i.e. the creation of taxonomic hierarchies to model properly the concepts in the domain of discourse. OntoClean comprises a set of meta properties, restrictions and assumptions which together defines a methodology for conceptual analysis of taxonomic subsumption (is-a) in any arbitrary ontology. OntoClean does not care about the semantics of the relationship itself, but with the ontological nature of concepts in the relationship. Due to the lack of stable implementations in the literature, this paper presents an OntoClean

implementation in Constraint Handling Rules (CHR^v), a Constraint Programming Prolog extension.

CHR^v [Frühwirth 2009] is a rule based language which was initially conceived to represent white box constraint solvers, but that has been shown to be able to implement many different reasoning services in a straightforward way. Through this language, we have defined a set of rules to check any restrictions violation imposed by OntoClean, known as OCIP (OntoClean Implementation in Prolog).

This paper is organized as follows. Section 2 explores the OntoClean methodology, highlighting the meta properties and restrictions used in this work. Then, Section 3 illustrates CHR^v language through an example for coloring maps. Syntax and Semantics are briefly discussed. Section 4 discusses the implementation of OntoClean in Prolog. Following, Section 5 explains the evaluation of a legal ontology through OCIP, highlighting some violations. Sections 6 explores some related work. Finally, last section presents a conclusion of what has been achieved so far in this research, as well as outlines up prospects for the continuation of this work.

2. OntoClean

An ontological model must evolve, since several and different knowledge sources can contribute to the addition of new concepts, relations and properties. Hence, we expect a certain level of quality during the engineering of ontologies, as well as the ontological commitment, in order to produce clear, well-formulated and correct subsumption relations. That is, decisions regarding the taxonomic structure must faithfully represent the real domain elements and their associations. In addition to leveraging the understanding with a cleaner ontology, the correct establishment of subsumption between these concepts (relying on the ontological nature of them) favors the reuse and integration of these models. Hence, it avoids rework to adjust/tune ontologies by adding new knowledge, allowing them to be widely shared across several information systems.

Therefore, it is suggested that a methodology for decision evaluation is widely required. Furthermore, this methodology must not be directed to a particular domain. It must be general enough to be used and reused in different fields, without adjustments. Thus, this work explores a domain-independent methodology for assessing decisions about the ontological nature of the elements in subsumption relations, namely the OntoClean [Guarino and Welty 2000], [Guarino and Welty 2004]. This methodology, relying on notions arising from philosophical ontology, was proposed by the Ontology Group at the Italian National Research Council (CNR).

OntoClean is a methodology that allows the construction of clean ontologies. Firstly, establishing a set of general and well formalized meta properties so that the concepts can be properly characterized. Secondly, as a result, these meta properties impose a set of constraints between a super and a subclass. Regarding the ontology definition proposed by [Studer et al 1998]: "*A formal explicit specification of a shared conceptualization*", hopefully, through the methodology is possible to detect possible disagreements amongst different conceptualizations, so that some corrective action can be taken.

2.1. OntoClean Meta Properties

Table 1 summarizes the basic notions extracted from philosophical sphere, in which OntoClean is based, namely: Rigidity, Identity, Unity and Dependence. Herein, we will consider concepts and classes as equivalent. Therefore, they represent a collection of individuals, which have been grouped by having common characteristics. The concepts are related by subsumption association, where concept ρ subsumes concept σ, if $\sigma \rightarrow \rho$ which means that all individuals from σ, are also part of ρ, but the reverse is not necessarily true. For further information, the complete basic notions as well as a brief formal analysis can be found at [Welty and Guarino 2001].

Table 1. OntoClean Meta Properties

Meta Property	Symbol	Label	Definition
Rigidity	+R	Rigid	All the instance will always be instances of this concept in every possible world
	-R	Non-Rigid	There are instances that will stop being instances of the concept
	~R	Anti-Rigid	All instances will no longer be instance of that concept
Identity	+I	Carry Identity	Instances carry an unique identification (IC) criteria from any superclass
	-I	Non Carry Identity	There is no identification criteria (IC)
	+O	Supply Identity	Instances themselves provide an unique identification criteria (IC)
Unity	+U	Unity	Instances are “whole”, and have a single unit criteria (UC)
	-U	Non-Unity	Instances are “whole”, but they do not have a single unit criteria (UC)
	~U	Anti-Unity	Instances are not “wholes”
Dependence	+D	External Dependence	There is dependency on external concept
	-D	Non External Dependence	There is no dependency

2.2. OntoClean Constraints

From the methodology, emerges a set of restrictions on the subsumption relations present in the taxonomy. In total, there were defined five restrictions [Guarino and Welty 2000], which follow:

- Anti-rigid class cannot subsume a rigid subclass;
- A class with identity cannot subsume a non-identity subclass;
- A class with the unity meta property cannot subsume a subclass without unity criterion;
- Anti-Unit class cannot subsume unity class;
- Dependent class cannot subsume non-dependent class.

3. CHR^v

Constraint Handling Rules with Disjunction (CHR^v) [Abdennadher and Schütz 1998] is a general concurrent logic programming language, rule-based, which has been adapted to a wide set of applications as: constraint satisfaction [Wolf 2005], abduction [Gavanelli et al 2008], component-development engineering [Fages et al 2008], and so on. The language also emerges as an attempt to integrate an ontological language of the Semantic Web with some rule-based logic programming [Frühwirth 2007]. In essence, it is designed for creation of constraint solvers. CHR^v is a fully accepted logic programming language, since it subsumes the main types of reasoning systems [Frühwirth 2009]: the production system, the term rewriting system, besides Prolog rules. Additionally, the language is syntactically and semantically well-defined [Abdennadher and Schütz 1998].

Without loss of generality, a CHR^v program is a conjunction of simpagation rules, whose syntax is described as follows:

rule_name@ Hk \ Hr <=> G | B.

rule_name@ is the non-compulsory rule identification. The head is defined by the predicates represented by Hk and Hr, with which an engine tries to match with the constraints in the store. Further, G stands for the set of guard predicates, that is, a condition imposed to be verified to fire any rule. Finally, B is the disjunctive body, corresponding to a set of constraints added within the store, whenever the rule fires. The logical conjunction and disjunction of predicates are syntactically expressed by the symbols ',' and ';', respectively. Logically, the interpretation of the rule is as follows:

$$\forall V_{GH}\ (G \rightarrow ((Hk \wedge Hr) \leftrightarrow (\exists\ V_{B\setminus GH}\ B \wedge Hk))),$$

$$\text{where } V_{GH} = vars(G) \cup vars(Hk) \cup vars(Hr),$$

$$V_{B\setminus GH} = vars(B) \setminus V_{GH}$$

For the sake of space, we ask the reader to check the bibliography for further reference to the declarative semantics. Besides the simpagation rule, there are two other cases that are specializations of the former: the simplification (Hr <=> G | B.) which replaces constraints by others equivalent; and the propagation rules (Hk ==> G | B.) which add new constraints within the store, leading to further simplification.

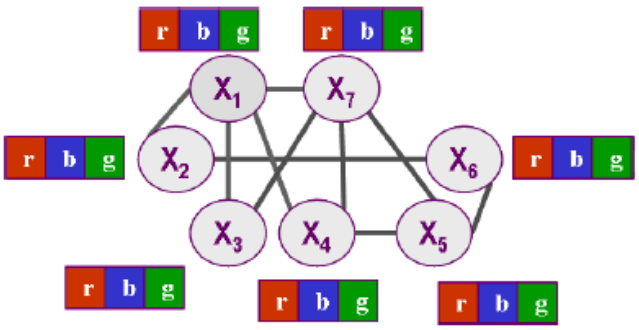

Figure 1: A Pedagogical Map Coloring Problem

Figure 1 illustrates a pedagogical map coloring problem. There are 7 places (X1, X2, X3, X4, X5, X6, X7) whose neighborhood is expressed by an arc connecting these

locations. Further, each one can assume one of the following domain values: D = {r,g,b}, referring to the colors, red, green, and blue, respectively. The only constraint imposed restricts the neighboring places (that is, each pair of nodes linked by an arc) to have different colors. As usual, this problem can be reformulated into a search tree problem, where the branches represent all the possible paths to a consistent solution. By definition, each branch not in accordance with the restriction must be pruned. The problem depicted in Figure 1 is represented by the logical conjunction of the following CHR^v rules:

```
f@ facts ==> m, d(x1,C1), d(x7,C7), d(x4,C4), d(x3,C3), d(x2,C2), d(x5,C5), d(x6,C6).
d1@ d(x1,C) ==> C=red; C=green; C=blue.
d7@ d(x7,C) ==> C=red; C=green; C=blue.
d4@ d(x4,C) ==> C=red; C=green; C=blue.
d3@ d(x3,C) ==> C=red; C=green; C=blue.
d2@ d(x2,C) ==> C=red; C=green; C=blue.
d5@ d(x5,C) ==> C=red; C=green; C=blue.
d6@ d(x6,C) ==> C=red; C=green; C=blue.
m@ m <=> n(x1,x2), n(x1,x3), n(x1,x4), n(x1,x7), n(x2,x6), n(x3,x7), n(x4,x7), n(x4,x5), n(x5,x7),
n(x5,x6).
n1@ n(Ri,Rj), d(Ri,Ci), d(Rj,Cj) <=> Ci=Cj | fail.
```

The first rule `f` introduces the constraints into the store, which is a set of predicates with functor `d` and two arguments: the location and a variable to store the possible color. The seven following rules relate the locations with the respective domain. Additionally, rule `m` adds all the conceptual constraints, in the following sense: `n(Ri,Rj)` means there is an arc linking `Ri` to `Rj`, thus, both places could not share the same color. Finally, the last rule is a sort of integrity constraint. It fires whenever the constraints imposed are violated. Logically, it says that if two linked locations `n(Ri,Rj)` share the same color (condition ensured by the guard), then the engine needs to backtrack to a new (consistent) valuation.

4. OCIP

Due to lack of implementations for OntoClean as already discussed briefly, this paper proposes a new, simple Prolog-based implementation, particularly using the CHR library provided by SWI-Prolog[1]. Being a logic, rule-based and constraint-oriented language, CHR^v has allowed a rapid prototyping of an ontology analyser: OCIP (*OntoClean Implementation in Prolog*). Through propagation rules, a forward chaining reasoner analyzes metaproperties and restrictions pointing out the inconsistencies.

In essence, the analyzer focuses on two logical predicates: `sub/2` and `oc/5`. The former establishes a subsumption relation between two classes of the ontology, i.e. `sub(ClassA, ClassB)` means `ClassA` subsumes `ClassB`. Only direct subsumption relations (between an arbitrary parent class and its immediately children classes) have to be

[1] http://www.swi-prolog.org/

directly defined by the user. The other relations (involving ancestor classes) are trivially propagated through the following transitivity rule:

```
transitivityRule@ sub(CA,CB), sub(CB,CC) ==> sub(CA,CC).
```

Following, the logical predicate oc/5 lists the metaproperties of any class, which is the first argument, and the four other defining respectively the Rigidity, Identity, Unity and Dependence criteria. oc(agent, r, ni, nu, nd) states that the Agent Class is rigid, besides it has non identity, non unity, and non external dependence. Other available labeling criteria are: Anti Rigid (ar), Non Rigid (nr), Identity (i), Owner Identity(o), Unity (u), Anti Unit (au) and finally, Dependence (d).

In order to avoid trivial non-termination (when forward chaining reasoners match indefinitely the same predicates with the same rules), besides the CHR^{v} operational semantics [Duck et al 2004] fully adopted by the SWI-Prolog, some simpagation rules delete equivalent predicates.

```
sympaOcRule@ oc(Class,R,I,U,D) \ oc(Class,R,I,U,D) <=> true.

sympaSubRule@ sub(CA,CB) \ sub(CA,CB) <=> true.
```

In essence, OCIP is based on three blocks rules: rules of natural propagation, horizontal constraints and vertical constraints, plus some auxiliary predicates for explanation of violating constraints. With regard to the rules of natural propagation, new facts contemplating the same class are propagated into the knowledge base. It is known, for example, a class that provide their own identification criterion (+O), is logically a rigid class (+R), which carries an identification criterion (+I). Also, anti-unit classes (~U) are also non-unit classes (-U). The same logical consequence is valid for metaproperties Anti-Ridig (~R) and Non-Rigid (-R).

```
supplyPropagRule@ oc(Class, ,o,X,Y) ==> oc(Class,r,i,X,Y).

unityPropagRule@ oc(Class,X,Y,au,Z) ==> oc(Class,X,Y,nu,Z).

rigidPropagRule@ oc(Class,ar,X,Y,Z) ==> oc(Class, nr,X,Y,Z).
```

Horizontal constraints have this name because they do not analyze superclass/subclass relationships. Unlike, these only evaluate whenever a class has been erroneously characterized with inconsistent metaproperties, like (+R and -R). Therefore, this block has four rules, one for each metaproperty. It is worth noting that anti-properties (~U and ~R) have not been codified, since firing the propagation rules (of the last block), classes should also be classified as -U and -R, respectively.

```
rigidRule@ oc(Class,r, , , ), oc(Class,nr, , , ) ==> rigidViolation(Class).

identityRule@ oc(Class, ,i, , ), oc(Class, ,ni, , ) ==> identityViolation(Class).

unityRule@ oc(Class, , ,u, ), oc(Class, , ,nu, ) ==> unityViolation(Class).

depedenceRule@ oc(Class, , , ,d), oc(Class, , , ,nd) ==> dependentViolation(Class).
```

The 1-ary prolog predicates (rigidViolation, identityViolation, unityViolation, dependentViolation) use other built-in predicates to generate explanations to the user about inconsistencies detected by class. The last rule block corresponds the vertical constraints, that is, those which evaluate the relations of subsumption. For each

OntoClean constraint (mentioned before), a CHR^{v} rule will identify whether there is any violation.

antiRigidRule@ oc(ClassSuper,ar, , ,), oc(ClassSub,r, , ,), sub(ClassSuper,ClassSub) ==> antiRigidViolation(ClassSuper,ClassSub).

noIdentityRule@ oc(ClassSuper, ,i, ,), oc(ClassSub, ,ni, ,), sub(ClassSuper,ClassSub) ==> noIdentityViolation(ClassSuper,ClassSub).

nonUnityRule@ oc(ClassSuper, , ,u,), oc(ClassSub, , ,nu,), sub(ClassSuper,ClassSub) ==> noUnityViolation(ClassSuper,ClassSub).

antiUnityRule@ oc(ClassSuper, , ,au,), oc(ClassSub, , ,u,), sub(ClassSuper,ClassSub) ==> antiUnityViolation(ClassSuper,ClassSub).

nonDependentRule@ oc(ClassSuper, , , ,d), oc(ClassSub, , , ,nd), sub(ClassSuper,ClassSub) ==> noDependentViolation(ClassSuper,ClassSub).

5. Evaluating OCIP through Legal Ontologies

5.1. Legal Ontologies

OntoCrime and OntoLegalTask [Rodrigues 2015] are ontological representations, through which it is possible to formalize the Brazilian Penal Code[2], making it possible to check the violation of norms, the resolution of legal conflicts (known as antinomies) and the automation of legal reasoning. These formalities have arisen due to semantic deficiencies found in legal texts, either linguistic or conceptual order. OntoCrime emerges as a Domain ontology, defining key concepts and relationships arising from the Penal Code, such as Crime, Punishment, Rules and Articles. OntoLegalTask extends these concepts by defining the tasks mentioned above.

The formalization expected for sound representation and complete reasoning relies on the Descriptions Logic (DLs) [Sirin et al 2007], a decidable subset of First Order Logic (FOL). DLs are the core of OntoCrime and OntoLegalTask representation, for structuring the knowledge bases and for providing reasoning services. Below, we list some DL expressions, which indicate: (i) a person who cannot be criminally punished is the one with a mental illness or a child, teenager or elderly person, (ii) an attributable person is one who does not fit the above profile, (iii) a prohibitive norm prohibits some conduct, (iv) a conduct is prohibited by some article, (v) Crime is a prohibited conduct (vi) with arrest or detention as punishment. For the sake of space, we ask the reader to check the reference for further information.

i. UnimputablePerson ≡ NaturalPerson ⊓ (∃hasDisorder.MentalDisorder ⊔ ∃hasAge.(Child ⊔ Teenager ⊔ Elderly)

ii. AttributablePerson ⊑ ¬UnimputablePerson

iii. ProhibitiveArticle ≡ ∃prohibits.ConductProhibited ⊓ ∀prohibits.ConductProhibited

iv. ConductProhibited ⊑ isProhibitedBy.ProhibitiveArticle

[2] http://www.planalto.gov.br/ccivil 03/decreto-lei/del2848.htm

v. Crime ⊑ ConductProhibited

vi. Crime ≡ ∃hasPunishment.(Arrest ⊔ Detention)

5.2. Legal Ontologies Labelling

Figures 2 and 3 illustrate a partial view of the the *is-a* relationships extracted from the legal ontologies for analysis. In the legal field conceptualization, Agent is a rigid class whose instances are "whole", but with different unit criteria since the class specializes in Person and Organization. Similarly, instances do not have the same criterion of identity. An instance of an Organization will always be necessarily an organization. Whenever an Organization faces bankruptcy, the entity ceases to be an organization, but also ceases to exist. If an Organization is bought by other, and change the CNPJ[3], the organization also is not the same, it ceased to exist and became a new one. Clearly, the criterion of identity of the instances is the CNPJ itself. A Person, in turn, has as a criterion of identity their own fingerprint.

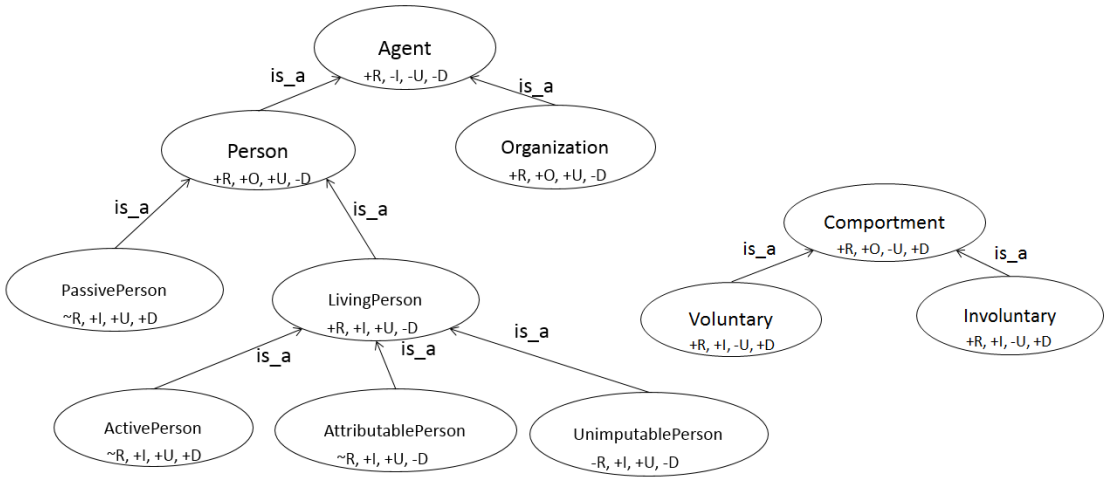

Figure 2: Agent and Comportment subclasses

In criminal law, the passive person is one who suffered the criminal action (or their dependents in the case of murder), while the Active person is the one who practices the act. Thus, a Person may cease to be passive, but it will be the same person. Suppose that this person is an active agent in another crime, notably, it cannot be also passive by the restrictions of the penal code. Another point is that a passive person could be passive in different crimes, but there is not a global criterion of identification; a passive person may be the victim of a thief, or may be the daughter of someone who was murdered, for example. Instances of this class must be related to a criminal conduct, there is an external dependency. The ActivePerson class fits in the same labels.

Class AttributablePerson is anti-rigid: people become old naturally. On the other hand, when a teenager goes into adulthood, he is no longer unimputable. Nevertheless, whether a person has been diagnosed with a mental disorder, even after the legal age, he

[3] CNPJ is the National Register of Legal Entities: a unique number that identifies a legal person

cannot be criminally penalized. Therefore, some instances of UnimputablePerson[4] remain unimputable while live.

With respect to class Comportment (and its subclasses), this depends on the agent who performs the comportment. An instance is bound to be a behavior in all possible worlds. Comportment instances can be identified by a set of variables such as: the action, the agent, place and time. However, there is no single unit criterion: some actions are performed with (criminals) objects, others do not.

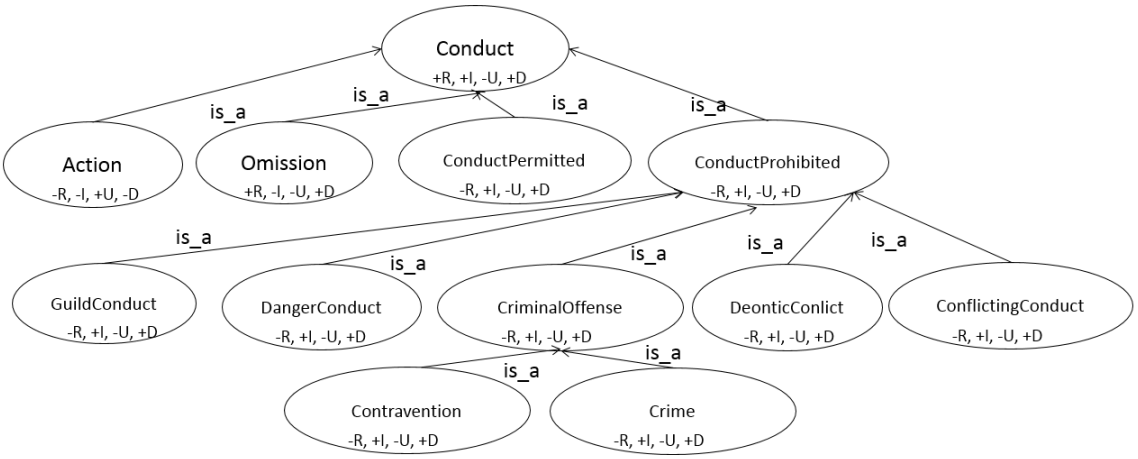

Figure 3: Conduct sublcasses

Similar to Comportment class, Conduct has the same metaproperties, except that it does not provide its own criterion of identity. According to the Criminal Code, Conduct is a voluntary Comportment (thus implicitly Conduct becomes a Voluntary subclass). Among its subclasses, Action class brings together some peculiar characteristics. While a Conduct depends on other external factors, an Action is something more specific, self-contained. Action instances do not share a common IC. Furthermore, their instances have a morphological unit in common (linguistically speaking): are verbs. Finally, the class is labeled with non-rigid metaproperty. Depending on the context, some instances can no longer be part of that concept. When one says, "He walked down the street when he was hit by a car" clearly there is an action. However, when someone says, "He walked nervously", we just have an agent state/condition[5].

ConductProhibited is non-rigid because there may be decriminalization. Instances have an IC in common: the Article prohibitive, with ongoing dependence. Nevertheless, with decriminalization, this prohibitive document will no longer be valid. In principle, the conduct boundaries are known (wholes instances), but there is no single UC for all of them. The ConductProhibited subclasses have the same metaproperties. In GuiltConduct, for example, there are dependencies, as it needs to know the agent

[4] Dead people, animals and legal people cannot be criminally penalized.

[5] In the Portuguese Grammar, in this context, *walk* is a linking verb, which does not indicate action, but a state.

liability. Similarly, for Contravention and Crime, the dependence relates to the kind of penalty imposed by the article of the law. In contrast, due to criminalization, ConductPermitted is labeled as non rigid. The class also has no common IC criteria.

5.3. OCIP Evaluation

In order to carry out the ontological evaluation through OCIP implementation, it was necessary to add into a knowledge base the facts concerning the subsumption relation and the meta properties of the ontology concepts, as previously shown. Some violations were identified as the identity and dependence criteria, between Comportment, Voluntary, Action and Omission classes:

- Identity Class Comportment can not subsume Non Identity Class Action;
- Dependent Class Comportment can not subsume Non Dependent Class Action;
- Identity Class Voluntary can not subsume Non Identity Class Action;
- Identity Class Comportment can not subsume Non Identity Class Omission;

Revisiting the General Theory of Crime within the Brazilian Penal Code, it is said that: "*Conduct **is** any human action or omission, conscious and voluntary, focused on one purpose*". Then, at a first glance, a misinterpretation leads us to believe that a conduct is accomplished through an action or omission, besides being a purely voluntary comportment. Poor written specifications has caused inconsistencies such as these detected by OCIP. Unfortunately, these flaws and ambiguities (besides other linguist and conceptual problems) are present in other legal documents. To fix the inconsistency, it is enough to say that: "*Conduct **has** a human action or omission, conscious and voluntary [...]*". From this perspective, the classes Action and Omission were disconnected as of Conduct specializations. In fact, a Conduct has an action/omission.

6. Related Work

OntOWLClean is a proposal for cleaning ontologies using OWL. The approach has been implemented both in SWOOP tool, as through Protégé [Welty 2006]. In the former case, the tool is no longer available, and in the latter case the plug-in was discontinued. This research had defined two ontologies: one with the metaproperties and restrictions and the other with the semantic definitions to map the classes from a domain ontology in the metaproperties (classes of OntOWLClean). Each new domain ontology then needed to be mapped into OntOWLClean. In addition, the tools have had problems to clearly display or explain the inconsistency.

WebODE[6] was a framework for editing ontologies, which had allowed Ontological analysis, but the environment was discontinued in 2006. Also, the plug-ins available for Ontology edition/evaluation in NeOn project[7] does not support analysis by OntoClean. Being a framework for defining, creating and analyzing, WebODE presented portability issues with other platforms. So an ontology created in WebODE would hardly be analyzed in another tool.

---

[6] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/old-technologies/60-webode

[7] http://www.neon-project.org/

UFO [Guizzardi and Wagner 2004] is a foundational ontology for evaluate business modeling methods, based on other foundational ontologies as OntoClean/DOLCE. OCIP is designed to be a simple tool to assess domain ontologies, even without a foundational ontology. OCIP will be transformed into a plug-in for Protégé that people have a more simple and intuitive interface for analysis. It will be possible for the user, for example, to choose specific metaproperties for analysis, or even graphically shows the backbone of ontology (rigid classes). In fact, the CHR^{v}/Prolog language enables rapid rule prototyping for forward and backward reasoning to manipulate the metaproperties and restrictions. Although, as a future research, we plan to deepen the analysis amongst UFO and OCIP.

6. Conclusion and Future Steps

For a correct and ambiguity-free formalization of knowledge, an important and necessary step is the ontological validation to determine whether there is a close correlation between the domain knowledge and that modeled. OntoClean has emerged as a simple and precise methodology by grouping a set of metaproperties, constraints and assumptions to produce clean and consistent ontological structures. Surprisingly, as far as we know, it has not been identified any valid implementation of OntoClean methodology.

On the other hand, CHR^{v} has become a general and powerful logic language capable of creating constraint-oriented systems through rewriting, propagation, and Prolog-based system. Then this project has followed the idea of creating the OCIP: a Prolog-based implementation (in particular through CHR library), where the OntoClean metaproperties could be attributed, and consequently, the restrictions could be verified. Furthermore, the general purpose CHR^{v} language led to the creation of a simple validator, but that fully covers the methodology proposed by OntoClean.

Our next step will be to make the OCIP implementation plug-in available, so that more and more researchers can use and share their experiences, difficulties and improvements. A key step will be to build a parser able to read the RDF/OWL ontologies code and automatically create the necessary facts (metaproperties and subsumption relationships), leaving the user only labeling directly on the facts of the knowledge base.

Finally, it's worthy of remembering that due to the large amount and the heterogeneity of documents, the ontological evaluation is essential. As soon as more and more models are being built, justify the need for the plug-in to evaluate whether the concepts, their relationships and properties, really express what you see.

References

Abdennadher S. and Schütz, H. (1998) "CHR^{v}, a flexible query language," pp. 1–14.

Baader. F., Calvatese. D., McGuinness. D., Nardi, D. and Patel-Schneider, P. F. (2007) "The description logic handbook, theory, implementation, and applications" (2nd edition). CAMBRIDGE: Cambridge University Press.

Duck, J. D., Stuckey, P. J., de la Banda, M. J. G. and Holzbaur, C. (2004) “The refined operational semantics of constraint handling rules.” in ICLP, ser. Lecture Notes in Computer Science, B. Demoen and V. Lifschitz, Eds. Springer, pp. 90–104.

Fages, F., Rodrigues, C. M. O. and Martinez, T. (1998) “Modular CHR with ask and tell,” In: CHR ’08: Proc. 5th Workshop on Constraint Handling Rules, (Linz, Austria) pp. 95–110.

Frühwirth, T. (2007) “Description logic and rules the CHR way,” pp. 49–61, extended Abstract.

Frühwirth, T. (2009) Constraint Handling Rules, 1st ed. New York, NY, USA: Cambridge University Press.

Gavanelli, M., Alberti, M. and Lamma, E. (2008) “Integrating abduction and constraint optimization in constraint handling rules,” in Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence. Amsterdam, The Netherlands, The Netherlands: IOS Press, pp. 903–904. [Online].

Guarino, N. (1998) “Formal Ontology in Information Systems”. Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy, 1st ed. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Guarino, N. and Welty, C. (2000) “Ontological analysis of taxonomic relationships,” in Conceptual Modeling ER 2000, ser. Lecture Notes in Computer Science, A. Laender, S. Liddle, and V. Storey, Eds., vol. 1920.

Guarino, N. and Welty, C. (2004) “An Overview of OntoClean”, in Handbook on Ontologies, ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds.

Guizzardi, G. and Wagner G (2004) A Unified Foundational Ontology and some Applications of it in Business Modelling. Proc on Ws on Enterprise Modelling and Ontologies for interoperability (EMOI-INTEROP).

Rodrigues, C. M. O., Freitas, F. L. G., Silva, E. P., Azevedo, R. R. and Vieira, P. (2015) “An ontological approach for simulating legal action in the brazilian penal code,” in Proceedings of The 2015 ACM Symposium on Applied Computing, University of Salamanca, Salamanca, Spain, pp. 376–381.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. and Katz, Y. (2007) “Pellet: A practical owl-dl reasoner,” Web Semant., vol. 5, no. 2, pp. 51–53.

Studer, R., Benjamins, V.R. and Fensel, D. (1998) “Knowledge engineering: Principles and methods,” Data Knowl. Eng., vol. 25, no. 1-2, pp. 161–197.

Welty, C. and Guarino, N. (2001) “Supporting ontological analysis of taxonomic relationships,” Data Knowledge Engineering, vol. 39, no. 1, pp. 51–74. [Online].

Welty, C. (2006) “OntOWLClean: Cleaning OWL ontologies with OWL,” in Proceeding of the 2006 conference on Formal Ontology in Information Systems. Amsterdam, The Netherlands, The Netherlands: IOS Press, pp. 347–359.

Wolf, A. (2005) “Intelligent search strategies based on adaptive constraint handling rules.,” Theory and Pratice of Logic Programming 5(4-5), 567-594.