A Structural $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model for Change Modelling

Perrine Pittet, Christophe Cruz, and Christophe Nicolle

LE2I, UMR CNRS 630, University of Burgundy - Dijon, France
{perrine.pittet,christophe.cruz,cnicolle}@u-bourgogne.fr

Abstract. This paper presents a complete structural ontology model suited for change modelling on $\mathcal{SHOIN}(\mathcal{D})$ ontologies. The application of this model is illustrated along the paper through the description of an ontology example inspired by the UOBM ontology benchmark and its evolution.

Keywords: $\mathcal{SHOIN}(\mathcal{D})$ Description Logic; Change Modelling, OWL DL.

1 Introduction

Ontologies make possible to application, enterprise, and community boundaries of any domain to bridge the gap of semantic heterogeneity. Ontologies development, to be correctly achieved, requires a dynamic and incremental process (Djedidi & Aufaure, 2008.). It starts with a rigorous ontological analysis (Guarino, 1995) that provides a conceptualization of the domain to model agreed by the community. The ontology, specified in a formal language, approximates the intended models of the conceptualization: the closer it is the better it is. The ontology needs to be revised and refined until an ontological commitment is found. Ulterior updates of the ontology, addressed by ontology evolution, aim at responding to changes in the domain and/or the conceptualization (Flouris, 2008). Changes are consequently inherent in the ontology life cycle. Modelling changes then implies having an exhaustive and non-ambiguous definition of the ontology model according to its language, so that each element of the ontology impacted by changes can be formally described.

This paper focuses on the $\mathcal{SHOIN}(\mathcal{D})$ level of expressivity, on which the ontological language OWL DL is based (Horrocks I. , 2005). After presenting the structural constraints of a $\mathcal{SHOIN}(\mathcal{D})$ ontology, it describes a list of basic changes, constrained by this structural model to avoid performing structural inconsistent updates on the ontology.

2 $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model

To formalize our framework the Karlsruhe Ontology Model (Ehrig, 2004) is used and extended to cover the whole $\mathcal{SHOIN}(\mathcal{D})$ constructors. From a mathematical point of view, an ontology can be defined as a structure. Formally, a structure is a triple

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 442–446, 2013.

$A=(S, \sigma, F)$ consisting of an underlying set S, a signature σ, and an interpretation function F that indicates how the signature is to be interpreted on S.

Definition 1: $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model

A $\mathcal{SHOIN}(\mathcal{D})$ ontology is a structure $O=(SO ,\sigma O ,FO)$ consisting of:

- The underlying set SO containing:
  - Six disjoint sets sC, sT, sR, sA, sI, sV, sK_R and sK_A called concepts, datatypes, relations, attributes, instances, data values, relation characteristics (among Symmetric, Functional, Inverse Functional, Transitive) and attribute characteristics (Functional),
  - Four partial orders $\leq_C$, $\leq_T$, $\leq_R$ and $\leq_A$, respectively on sC called concept hierarchy or taxonomy, on sT called type hierarchy, on sR called relation hierarchy and on sA called attribute hierarchy,

  such that $SO :=\{(sC, \leq_C),(sT, \leq_T),(sR, \leq_R),(sA, \leq_A), sI, sV, sK_R, sK_A,\}$,
- The signature σO containing two functions $\sigma_R{:}sR\rightarrow sC^2$ called relation signature and $\sigma_A{:}sA\rightarrow sC \times sT$ called attribute signature, such that $\sigma O :=\{\sigma_R, \sigma_A\}$,
- The interpretation function FO containing:
  - A function $\iota_C{:}sC\rightarrow 2^{sI}$ called concept instantiation,
  - A function $\iota_T{:}sA\rightarrow 2^{sV}$ called data type instantiation,
  - A function $\iota_R{:}sC\rightarrow 2^{sI\times sI}$ called relation instantiation,
  - A function $\iota_A{:}sC\rightarrow 2^{sI\times sV}$ called attribute instantiation,
  - A function $\kappa_R{:}sR\rightarrow 2^{sKR}$ called relation characterization,
  - A function $\kappa_A{:}sA\rightarrow 2^{sKA}$ called attribute characterization,
  - A function $\varepsilon_C{:}sC\rightarrow 2^{sC}$ called concept equivalence,
  - A function $\varepsilon_R{:}sR\rightarrow 2^{sR}$ called relation equivalence,
  - A function $\varepsilon_A{:}sA\rightarrow 2^{sA}$ called attribute equivalence,
  - A function $\varepsilon_I{:}sI\rightarrow 2^{sI}$ called instance equivalence,
  - A function $\delta_C{:}sC\rightarrow 2^{sC}$ called concept disjunction,
  - A function $\delta_I{:}sI\rightarrow 2^{sI}$ called instance differentiation,
  - A function $\text{-}_C{:}sC\rightarrow 2^{sC}$ called concept complement specification,
  - A function $\text{-}_R{:}sR\rightarrow 2^{sR}$ called relation inverse specification,
  - A function $maxCardR{:}sR\rightarrow N$ called relation maximal cardinality restriction,
  - A function $minCardR{:}sR\rightarrow N$ called relation minimal cardinality restriction,
  - A function $\sqcap_C{:}sC\rightarrow 2^{sC}$ called concept intersection,
  - A function $\sqcup_C{:}sC\rightarrow 2^{sC}$ called concept union,
  - A function $\sqcup_{iC}{:}sI\rightarrow 2^{sC}$ called concept union enumeration,
  - A function $\sqcup_V{:}sV\rightarrow 2^{sC}$ called data value union,
  - A function $\sqcap_{iC}{:}sC\rightarrow 2^{sI}$ called concept enumeration,
  - A function $\rho_{\exists R}{:}sR\rightarrow 2^{sC}$ called relation existential restriction,
  - A function $\rho_{\forall R}{:}sR\rightarrow 2^{sC}$ called relation universal restriction,
  - A function $\rho_{R:}{:}sR\rightarrow 2^{sI}$ called relation value restriction,
  - A function $\rho_{\exists A}{:}sA\rightarrow 2^{sT}$ called attribute existential restriction,
  - A function $\rho_{\forall A}{:}sA\rightarrow 2^{sT}$ called attribute universal restriction,
  - A function $\rho_{A:}{:}sA\rightarrow 2^{sV}$ called attribute value restriction,

such that $FO:=\{\iota_C, \iota_T, \iota_R,\iota_A, \kappa_R, \kappa_A, \varepsilon_C, \varepsilon_R, \varepsilon_A, \varepsilon_I, \delta_C, \delta_I, \text{-}_C, \text{-}_R, maxCard_R, minCard_R, \sqcap_C, \sqcup, \sqcup_{iC}, \sqcup_V, \sqcap_{iC}, \rho_{\exists R}\ \rho_{\forall R}, \rho_{R:}, \rho_{\exists A}, \rho_{\forall A}, \rho_{A:}\}$.

2.1 $\mathcal{SHOIN}(\mathcal{D})$ Change Modelling

To model changes, we give the five definitions below.

- **Definition 2: Change.** A change ω is the application of a modification on an ontology O, that potentially affects one or more elements of its structure as defined by the $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model.
- **Definition 3: Log of Changes.** Given an ontology O a log of changes, noted log_i, is defined by an ordered set of changes (simple and complex) $<\omega_1, \ldots, \omega_n>$ that applied to O results in O.
  Like in (Klein, 2004), 2 change types are distinguished: basic and complex.
- **Definition 4: Basic Change.** A basic change on an ontology O is a function $\omega_B: sK \rightarrow 2^O$ with $sK:=\{sC \cup sI \cup sR \cup sA\}$ that corresponds to an addition, a removal of a modification of one element $\in O$.
- **Definition 5: Complex Change.** A complex change on an ontology O is a disjoint union of basic changes. It is a function $\omega_C: nsK \rightarrow 2^O$ such that $\omega_C := \omega_{B1} + \ldots + \omega_{Bn}$.

The application of a change on an ontology, basic or complex, can be an addition or a deletion. It is traced as such in the log of changes.

- **Definition 6: Addition of a Change.** The addition of a change ω_i traced in the log of changes log_i, noted $log_i + \{\omega_i\}$, is defined by the disjoint union between the two disjoint sets log_i and $\{\omega_i\}$.
- **Definition 7: Deletion of a Change.** The deletion of a change ω_i traced in the log of changes log_i, noted $log_i - \{\omega_i\}$, is defined by the set-theoretic complement such that $log_i - \{\omega_i\} = \{x \in log_i \mid x \notin \{\omega_i\}\}$.

2.2 Basic Changes Modelling

To produce the list of basic change operations on $\mathcal{SHOIN}(\mathcal{D})$ ontologies, the $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model is exploited as described in Table 1. The third column lists the 47 operators representing basic changes, which, if applied on the ontology, affect the corresponding $\mathcal{SHOIN}(\mathcal{D})$ model element. According to our model, every basic change can be declined as an addition or a deletion of an element of the underlying set, the signature or the interpretation function.

Table 1. Correspondence between $\mathcal{SHOIN}(\mathcal{D})$ Description Logic Descriptions, Axioms and Facts, the $\mathcal{SHOIN}(\mathcal{D})$ Ontology Model and the $\mathcal{SHOIN}(\mathcal{D})$-based List of Basic Changes

	DL Syntax	Model	$\mathcal{SHOIN}(\mathcal{D})$-based Changes Abstract Syntax
Descriptions	C	sC	Class(Class)
	$C_1 \sqcap \ldots \sqcap C_n$	$\sqcap_C$	IntersectionOf(Class$_1$,…,Class$_n$)
	$C_1 \sqcup \ldots \sqcup C_n$	$\sqcup_C$	**UnionOf(Class$_1$,…,Class$_n$)**
	$\neg C$	$-_C$	**ComplementOf(Class)**
	$\{I_1\} \sqcup \ldots \sqcup \{I_n\}$	$\sqcup_{IC}$	**OneOf(Class, Instance$_1$,…,Instance$_n$)**
	$\exists R.C$	$\rho_{\exists R}$	SomeValuesFrom(ObjectProperty, Class)
	$\forall R.C$	$\rho_{\forall R}$	AllValuesFrom(ObjectProperty, Class)

Table 1. (*continued*)

	$R : I$	$\rho_{R:}$	HasValue(ObjectProperty, Instance)
	$\geq n\ R$	$maxCard_R$	MinCardinalityProperty(ObjectProperty, n)
	$\leq n\ R$	$minCard_R$	MaxCardinalityProperty(ObjectProperty, n)
	$\exists A.T$	$\rho_{\exists A}$	SomeValuesFrom(DatatypeProperty, Datatype)
	$\forall A.T$	$\rho_{\forall A}$	AllValuesFrom(DatatypeProperty, Datatype)
	$A : V$	$\rho_{A:}$	HasValue(DatatypeProperty, Datavalue)
	T	sT	Datatype(Datatype)
	$\{V_1\} \sqcup \ldots \sqcup \{V_n\}$	$\sqcup_V$	**OneOf(Datavalue$_1$,…,Datavalue$_n$)**
	R	sR	ObjectProperty(ObjectProperty)
	A	sA	DatatypeProperty(DatatypeProperty)
	I	sI	Instance(Instance)
	V	sV	Datavalue(Datavalue)
	$C \equiv C_1 \sqcap \ldots \sqcap C_n$	$\sqcap_C$	**IntersectionClass(Class, (Class$_1$,…Class$_n$))**
	$C \equiv \{I_1\} \sqcap \ldots \sqcap \{I_n\}$	$\sqcap_{iC}$	**EnumeratedClass(Class, (Instance$_1$,…Instance$_n$))**
	$C_1 \sqsubseteq C_2$	$\leq_C$	SubClassOf(Class$_1$, Class$_2$)
	$C_1 \equiv \ldots \equiv C_n$	ε_C	**EquivalentClass(Class$_1$,…Class$_n$)**
	$\bot \equiv C_1 \sqcap C_2$	δ_C	DisjointClass(Class$_1$, Class$_2$)
	$T_1 \sqsubseteq T_2$	$\leq_T$	SubDatatypeOf(Datatype$_1$, Datatype$_2$)
	$V \in T_i$	ι_T	InstancesOfDatatype(Datavalue, Datatype)
	$\geq 1R \sqsubseteq C_i$	σ_R	DomainProperty(ObjectProperty, Class)
	$\top \sqsubseteq \forall R.C_i$		RangeProperty(ObjectProperty, Class)
	$R \equiv R_0^-$	$-_R$	InverseOf(ObjectProperty$_1$ ObjectProperty$_2$)
	$R \equiv R^-$	κ_R	SymmetricProperty(ObjectProperty)
	$\top \sqsubseteq \leq 1R$		FunctionalProperty(ObjectProperty)
	$\top \sqsubseteq \leq 1R^-$		InverseFunctionalProperty(ObjectProperty)
	$Tr(R)$		TransitiveProperty(ObjectProperty)
	$R_1 \sqsubseteq R_2$	$\leq_R$	InheritanceObjectPropertyLink(ObjectProperty$_1$, ObjectProperty$_2$)
	$R_1 \equiv \ldots \equiv R_n$	ε_R	**EquivalentProperty(ObjectProperty$_1$,…,ObjectProperty$_n$)**
	$A \sqsubseteq A_i$	A	DatatypeProperty(DatatypeProperty)
	$\geq 1A \sqsubseteq C_i$	σ_A	DomainProperty(DatatypeProperty Class)
	$\top \sqsubseteq A.T_i$		RangeProperty(DatatypeProperty, Datatype)
	$\top \sqsubseteq \leq 1A$	κ_A	FunctionalProperty(DatatypeProperty)
	$A_1 \sqsubseteq A_2$	$\leq_A$	InheritanceDatatypePropertyLink(DatatypeProperty$_1$, DatatypeProperty$_2$)
	$A_1 \equiv \ldots \equiv A_n$	ε_A	**EquivalentProperty(DatatypeProperty$_1$,…,DatatypeProperty$_n$)**
	$I \in C_i$	ι_C	InstancesOf(Instance, Class)
	$\{I, I_i\} \in R_i$	ι_R	InstancesOfObjectProperty(Instance, Instance$_1$, ObjectProperty)
	$\{I, V_i\} \in A_i$	ι_A	InstanceOfDatatypeProperty(Instance, Datavalue, DatatypeProperty)
	$\{I_1\} \equiv \ldots \equiv \{I_n\}$	ε_I	**SameAs(Instance$_1$,…,Instance$_n$)**
	$\{I_i\} \sqsubseteq \neg\{I_j\},\ i \neq j$	δ_I	**DifferentFrom(Instance$_1$,…,Instance$_n$)**

2.3 Complex Changes Modelling

More generally, an infinite set of complex changes can be generated from the aggregation of basic changes (Plessers, 2005). Their pertinence depends on the need of particular changes implied by particular uses. For example, the renaming of a concept is often used in collaborative development of an ontology to reach a consensus but can be unused in other contexts. For this reason, our model natively provides the limited set of 47 basic changes but, depending on change modelling needs, gives the opportunity to build complex changes from these basic changes.

3 Discussion and Conclusion

Our model aims at facilitating the modelling of basic and complex changes. It aims at contributing to the maintenance of the ontology structural consistency by clearly defining each change impact on the structure of the ontology. This model is the structural basis of a change management methodology called OntoVersionGraph (Pittet, 2012). To ensure a complete consistent evolution of the ontology, it is used in conjunction with a logical inconsistency identification methodology called CLOCk (Gueffaz, 2012), based on ontology design patterns and model-checking.

References

Djedidi, R., Aufaure, M.A.: Change Management Patternsfor Ontology Evolution Process –. IWOD at ISWC 2008, Karlsruhe (2008)

Ehrig, M.H.: Similarity for ontologies-a comprehensive framework. Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM (2004)

Flouris, G.M.: Ontology Change: Classification & Survey. The Knowledge Engineering Review 23(2), 117–152 (2008)

Guarino, N.: Formal ontology, conceptual analysis and knowledge representation. International Journal of Human Computer Studies 43(5), 625–640 (1995)

Gueffaz, M.P.: Inconsistency Identification In Dynamic Ontologies Based On Model Checking, pp. 418–421. INSTICC, ACM SIGMIS (2012)

Horrocks, I.: Owl: A description logic based ontology language. Logic Programming, 1–4 (2005)

Klein, M.C.: Change management for distributed ontologies (2004)

Pittet, P.N.: Guidelines for a Dynamic Ontology-Integrating Tools of Evolution and Versioning in Ontology. arXiv (2012)

Plessers, P., De Troyer, O.: Ontology Change Detection Using a Version Log. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 578–592. Springer, Heidelberg (2005)