Meta-Level Properties for Reasoning on Dynamic Data

Yuting Zhao[1(✉)], Guido Vetere[1], Jeff Z. Pan[2], Alessandro Faraotti[1], Marco Monti[1], and Honghan Wu[2]

[1] IBM Italia S.p.A., Roma, Italy
yuting.zhao@it.ibm.com
[2] Department of Computer Science, University of Aberdeen, Aberdeen, UK

Abstract. Dynamic features are important for data processing when dealing with real applications. In this paper we introduce a methodology for validating the construction of ontological knowledge base and optimising the query answering with such ontologies. In this paper, we firstly introduce some meta-properties of dynamic for ontologies. These meta-properties impose several constraints on the taxonomic structure of an ontology. We then investigate how to build up a meta-ontology with the constrains on these meta-properties. The goal of our methodology is *not* to help on providing strict logical conditions for judging inconsistency, but rather to help as heuristics for validating ontologies. Furthermore, some results on how to improve the reasoning on dynamic data by using these properties are also introduced.

1 Introduction

Dynamic is an important nature of data. Some data are always changing, *e.g.*, *the outdoor temperature of Rome*; some do not change, *e.g.*, *Chinese people speaks Chinese in general*. Some data will be valid for some period, *e.g.*, *Mr. David Cameron is the "Prime Minister of the UK"*; some data will change from one category to another, *e.g.*, if he retired, Mr. David Cameron would become the *"Former Prime Minister of the UK"*. Specially in the Big Data time, understanding the dynamic feature of data becomes a key issue for improving the quality of data, and developing efficient services based on the data.

Dynamic properties or stream data processing [1,2,4–7,11,12,14] have been studied in research communities of Artificial Intelligence, Database, and Linked Data for many years. Specifically, the first line of relevant techniques is those for modelling dynamics in the data level. Roughly speaking, a data stream is an (infinite) ordered sequence of data items. One of the key questions is how to model the timestamps in the data. The simplest approach is no timestamp associated with the data items, which means that the data is simply an ordered list of data items. The second approach is to attach a timestamp to each data item, which annotates when the item occurs. The third one is one step further from the second to add one more timestamp that means when the item was removed (or invalidated). The second type of approaches is the extensions on

G. Qi et al. (Eds.): JIST 2015, LNCS 9544, pp. 271–279, 2016.
DOI: 10.1007/978-3-319-31676-5_19

query languages for processing data dynamics. For example, CQL [2] is a continuous query language model for querying data streams. The stream data, infinite unbounded bag of data, is converted into discrete snapshots each of which is a finite bag of relations. So, the key question to solve here is how to select the right snapshot(s). Inspired by CQL, researchers in Linked Data community also proposed similar ideas to processing RDF data streams [1,4,11].

Our approach is a light weight solution, and does not request the special timestamp on the data. Furthermore, we are targeting a meta level description of the dynamic properties of the data items. One of the most related interesting works is OntoClean [9,10]. Guarino and Welty defined formal and domain independent properties of concepts (*i.e.* meta-properties) that aid the knowledge engineer to make correct decisions when building ontologies and therefore to avoid common mistakes. OntoClean originally defines four meta-properties as follow:

- **Identity:** the criteria for defining ‘sortal’ classes; classes all of whose instances are identified in the same way.
- **Unity:** a property that only holds of individuals that are ‘wholes’.
- **Rigidity:** a property that only holds of individuals that have it and cannot change, is ‘essential’.
- **Dependence:** a property that only holds for a class if each instance of it implies the existence of another entity.

According the methodology those meta-properties are used to analyze concepts in order to check hierarchies correctness, for example a ‘rigid’ concept like *Person* cannot be subsumed by an ‘not-rigid’ concept like *Physician*. In this paper, we go one step further and focus on formal description of changes and dynamic characterizations of the entities.

In general, “changes” in a dynamic knowledge base would be categorised in terms of meta-properties such as *Temporary*, *Reversible*, *Non-Reversible*, *Static*, *Rigid*, and *Periodic*, etc. If we model a dynamic knowledge base as a stream of Knowledge Bases (KBs), *i.e.* an ordered set of KBs sharing the same TBox where $KB(t)$ means the stream KB at time t, we could come up with a complete analysis in terms of meta-properties such as rigidity, cyclicity, monotonicity, etc. Then, one of the tasks could be that of inferring such meta properties for the KB TBox, based on observations of the stream KB at different times. Once we have a complete dynamic meta-level characterization of the TBox, we can decide how to optimize the access of the dynamic KB in many use cases. When applied to unary concepts, *Static* and *Temporary* relate to Rigid and Anti-Rigid in Guarinos formal ontology [8].

In this paper we analysis different kind of changes in a dynamic knowledge base, and categorise changes into some meta-properties. We have also investigated how to build up a meta-ontology with the constrains on these meta-properties. We stress the similar opinion as OntoClean, that our methodology is *not* to help on providing strict logical conditions for judging inconsistency, but rather to help as heuristics for validating the ontological knowledge bases, and optimising reasonings on them.

Currently the methodology of OntoClean does not aim at characterising ontology elements from the dynamic point of view and we believe that refining it by adding meta-properties about dynamics can be useful, when evolving domains (and datasets) are involved, both to design a well formed ontology and also to support reasoning. As an illustrative example we may have an individual which acquires a 'status' that cannot be changed any more, such kind of property is not rigid but become rigid somehow once acquired; we can imagine the case of person which once vaccinated against poliomyelitis remains immunized throughout his life. We called such meta-property 'Reversible' property. In a stream reasoning context, knowing that a property is not reversible makes it possible to leverage previous results so to avoid re-computation. So the second contribution in this paper is, we show some interesting results on how to improve the reasoning on dynamic knowledge base by using these properties.

We note in this paper we do not restrict our approach on any specific logic languages, but we use the convenience of normal expressive capabilities of *Description Logics* (DLs) [3] to present some logic properties in our discussion. DLs is a family of *Knowledge Representation* (KR) formalisms that represent the knowledge structure of an application domain, in terms of *concepts*, *roles*, *individuals*, and their relationships. We assume most of the readers are familiar with the DLs. At the same time, we also use the *First Order Logic* (FOL) to describe some semantics in our study, and we assume most of the readers are familiar with the FOL.

The paper is organised as follows. In Sect. 2 we introduce a formalization of dynamic knowledge base and various kind of "changes". In Sect. 3 we study the dynamic meta-properties and show how to build the meta-ontology. In Sect. 4, we show some interesting results on how to optimize Query Answering in a dynamic knowledge base with the meta-properties. A Conclusion is given in Sect. 5.

2 Knowledge Stream Bases and Changes

In this section we firstly define a generic formalization of dynamic knowledge base, then we classify several typical kind of changes in a dynamic knowledge base.

2.1 A Generic Knowledge Stream Bases

Here we formalize a *dynamic Knowledge Base* as a steam of knowledge bases, and then we use *change* to capture the difference between KBs.

Definition 1 ***(Knowledge Stream Bases (KSB)).*** *A* Knowledge Stream Bases (KSB) *is a steam of knowledge base* K_1, K_2, ..., K_n, *a* change $C(i)$ *is the difference between* K_{i-1} *and* K_i, i.e.,

$$C(i) = (K_i - K_{i-1}) \cup (K_{i-1} - K_i). \quad (1)$$

in where $(K_i - K_{i-1})$ *is the set of data ADDED in* K_i*, donated by* $C_{add}(i)$*; and* $(K_{i-1} - K_i)$ *is the set of data REMOVED from in* K_{i-1}*, donated by* $C_{del}(i)$*. So we have:*

$$C_{add}(i) = \{k | k \in K_i \text{ and } k \notin K_{i-1}\}; \tag{2}$$

$$C_{del}(i) = \{k | k \in K_{i-1} \text{ and } k \notin K_i\}. \tag{3}$$

Obviously above definition of "change" reflects the difference between the current KB (*e.g.* K_i) and the former KB (*e.g.* K_{i-1}). In general it contains two parts of information: the "new data" added in the current KB, and the "outdated data" removed from the previous KB.

2.2 Category of Changes in Knowledge Stream Bases

Based on common sense knowledge, there could be various kinds of changes on the data in a Knowledge Stream Bases. Without losing generality, the behavioural properties of "changes" in Stream data could be categorised as:

1. **Come and Remain (Change.CM)**: A fact is detected and will never change, *e.g.* a person was born: *hasBirthday*(Marie Curie, 1867-11-07). Obviously Change.CM belongs to the added data, but does not belong to the removed data, i.e., $Change.CM \subseteq C_{add}$ and $Change.CM \nsubseteq C_{del}$.
2. **Removed Forever (Change.RF)**: A fact is removed from the KB forever, *e.g.* if a patient is dead, then all in treatment data will be removed from the current KB, and move to the other KB.
3. **Come and Leave (Change.CL)**: A fact appeared for a while and then disappeared, *e.g.* a patient is pregnant: *Pregnant*(Mary). Obviously Change.CL belongs to both the added data and the removed data, i.e., $Change.CL \subseteq C_{add}$ and $Change.CL \subseteq C_{del}$.
4. **Periodical Changes (Change.PC)**: A fact has a value which will change periodically with repeated values, *e.g.* the changes of 4 seasons.
5. **Monotonic Changing the Value (Change.MC)**: A fact has a value which will change monotonically, *e.g.* the ages of a person: $hasAge^{-}.\{Mary\}$.
6. **Non-monotonic Changes (Change.NC)**: A fact has a value which will change NON-monotonically, *e.g.* the body temperature of a patient: $hasTemperature^{-}.\{Mary\}$.

In the following Table 1 we give the semantics to different kinds of categories of changes.

3 Meta-Properties for Dynamic and Meta Ontology

In this section we introduce a method to describe static and dynamic properties in a dynamic knowledge base. In general, these "changes" in knowledge streams would be categorised in terms of meta-properties such as Periodic, Monotonicity, etc. At the same time, we also need meta-properties such as Static and Rigid, in order to handle the real practical applications.

Table 1. Semantics of changes

Change.CM = $\{c \mid \exists t$, so that $c \in C_{add}(t)$, in where $i < t$, $c \notin K_i$, and $j \geq t$, $c \in K_j$ $\}$
Change.RF = $\{c \mid \exists t$, so that $c \in C_{del}(t)$, in where $i < t$, $c \in K_i$, and $j \geq t$, $c \notin K_j$ $\}$
Change.CL = $\{c \mid \exists i < j$, so that $c \in C_{add}(i)$ and $c \in C_{del}(j)$, in where $i \leq t < j$, $c \in K_t$ $\}$
Change.PC = $\{c \mid c \in$ Change.CL, and $\exists t$, let $k = [1, 2, \cdots]$, so that if $c \in K_i$ then $c \in K_{i+k*t}$, and if $c \notin K_i$ then $c \notin K_{i+k*t}$ $\}$
Change.MC = $\{c \mid$ if $i < j < k$, then either $c_{K_i} \geq c_{K_j} \geq c_{K_k}$, or $c_{K_i} \leq c_{K_j} \leq c_{K_k}$ $\}$
Change.NC = $\{c \mid$ if $c \notin$ Change.MC $\}$

– **Temporary:** Temporary property holds for entities in which individuals are always changing. For example, concept *Student* is temporary, because in general most of the individuals of *Student* would not be *Students* if they graduated. According to the semantic of changes in Table 1, *Change.CL* (come and leave) is a typical dynamic property.
– **Static:** Static property only holds for entities in which all the individuals will never be withdrew. For example, somebody is a *Person*, and he/she is always a person. In some special cases an individual may not have the membership of an entity at the beginning, but when it gained the membership, the membership always holds. For example, if *Peter* gained the *PhD* degree, he will always be a *PhD*. We call this kind of entities "*semi-static*". Obviously based on the semantic of changes in Table 1, *Change.CM* (come and remain) and *Change.RF* (removed forever) are semi-static.
– **Reversible:** An entity is "*reversible*", if it is the case that an individual could lose the membership of this entity, but could regain the membership again. For example, *Alice* was *Preganant* several years ago, and she is *Preganant* again. According to Table 1, *Change.PC* (periodical changes) is a typically reversible.
– **Non-Reversible:** A temporary entity is "*non-reversible*", if for this kind of entity, an individual lose the membership, he/she will never gain the membership again. For example, if someone had finished his/her job as the President of the USA, then he/she would never take this job again. So *president-of-USA* is not reversible. Based on the semantic of changes in Table 1, we find *Change.CM* (come and remain) and *Change.RF* (removed forever) are non-reversible.
– **Periodic:** A temporary entity is "*Periodic*", if the individual regains the membership of this entity periodically. For example, if "*today* is *Sunday*", then after every seven days, "*today* is *Sunday*" again. Obviously from Table 1, *Change.PC* (periodical changes) is a typical *Periodic* property.
– **Rigid:** a property is "*rigid*" if it is essential to all its possible instance; that only holds for a concept that have it and cannot change.

From now on we restrict the Knowledge Base to be an *ontology*, which $K = (T, A)$ contains a pair: T (*TBox*) is a set of terminology of concepts and roles, and A (*ABox*) is a set of assertions based on T and a set of individuals.

Now we can specify a dynamic ontology stream which is capable to embody dynamic characters of its knowledge entities in a meta level, as shown in Fig. 1.

Definition 2 ***(Dynamic Ontology Stream (DOS)).*** *A* Dynamic Ontology Stream (DOS) *contains a meta layer M, and a knowledge stream base (Definition 1) K_1, K_2, ..., K_n, in which $K_i = (T_i, A_i)$ is a normal ontology. The meta layer M contains:*

(1) following meta-concepts: Static*,* Rigid*,* Temporary*,* Reversible*,* Non-Reversible*, and* Periodic*, and a meta-TBox as:*

$$Rigid \sqsubseteq Static;$$
$$Periodic \sqsubseteq Reversible;$$
$$Reversible \sqsubseteq Temporary;$$
$$Non\text{-}Reversible \sqsubseteq Temporary.$$

(2) and assertions about the meta-concepts and with concepts and roles in T as individuals.

Here we have the following constraints for these meta-properties. Obviously this proposition always hold. For example, $Person \sqsubseteq Teenager$ is unsatisfiable, because typically if someone is a person, he/she is always a person; but being a teenager is just a short period of his/her life. Here we say "typically" because we prefer to use these constrains as heuristic for validating the ontological knowledge bases, instead of strict logical conditions for judging inconsistency.

Proposition 1 ***(Constrain Between Static & Temporary).*** *Given an ontology and two entities (concepts or roles) C and D, if C is* static *and D is* temporary*, then $C \sqsubseteq D$ is typically unsatisfiable.*

4 Optimised Reasoning and Query Answering

In the following we will show how to optimise query answering with dynamic mete-properties.

Query Answering with Dynamic Profiles. In the dynamic ontology stream defined above, we could leverage query answering with meta-level dynamic characterizations. For example, given a boolean query Q_1*( )= Person(Peter)*, and at time t we know *Perter* is a person. So the answer of this boolean query is *True* at time t. Since concept *Person* is static, the answer should be fixed. So in any future time $k > t$, we know the answer of the boolean query should always be *True*, even we do not need to check the knowledge base. Similarly if we have boolean query Q_2*( )=BlueEyes(Peter), Student(Peter)*, since *BlueEyes* is static but *Student* is temporary, the whole query Q_2 is temporary. We have the following result for conjunctions in Table 2. In this table, P and Q are knowledge

Table 2. Dynamic properties of conjunctions

P	Q	P∧Q
Static	Static	Static
Static	Temporary	Temporary
Static	Periodic	Periodic
Temporary	Temporary	Temporary$_{min(f(P),f(Q))}$
Temporary	Static	Temporary$_{min(f(P),f(Q))}$
Periodic	Periodic	Periodic$_{min(f(P),f(Q))}$

Table 3. Dynamic profile of conjunctive queries

Query profile	Execution	Result Validity
Static	single	unlimited
Temporary$_f$	periodic	at most up to f
Periodic$_f$	periodic	at least up to f

entities (concepts or roles). We also use a function $f(P)$ to assigning a frequency value [13] to entity P.

Furthermore, for any ontology characterized by dynamic meta-properties, we can calculate the dynamic profile of conjunctive queries (*e.g.* SPARQL). This would allow optimizing query execution over knowledge streams (Table 3).

Query Answering with Semi-Static Properties. In traditional way it is OK to check "What is in the KB", but difficult to find "What was in the KB". For example in the stream (upper part of Fig. 1), given query $Q(X) = President(X)$,

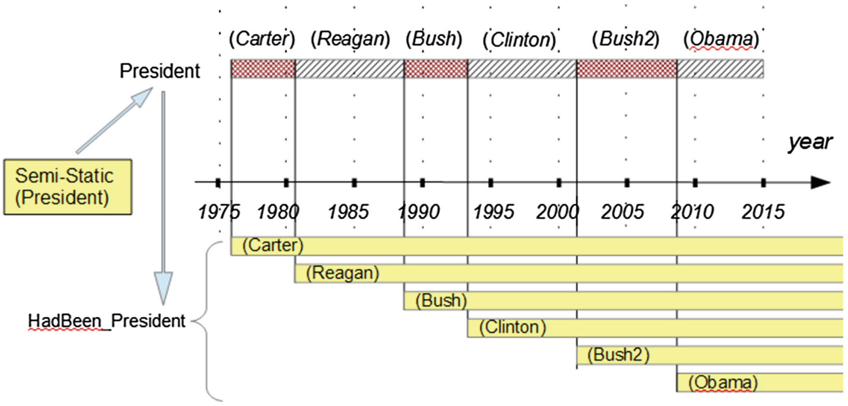

Fig. 1. Semi-static helps answering "What was in the KB".

the answer at time 1995 is *Clinton*, but at time 2010 is *Obama*. However it is difficult to answer "Who were the Presidents of the USA".

We note the dynamic nature of concept "Presidents of the USA" is *semi-static* and *non-reversible*. If someone was elected as the "Presidents of the USA", they will gain a property "had been the Presidents of the USA", and this property will never change in the future. For any interesting (being queried) semi-static concept P, an extra TBox axiom can be introduced (as shown in Fig. 1) in order to solve this kind of problem:

$$President \sqsubseteq HadBeing\_President \tag{4}$$

5 Conclusion

Inspirited by OntoClean [9] from Guarino and Welty, in this paper we have investigated a methodology to analysis the dynamic properties in ontology streams. Guarino and Welty defined generic and domain independent properties of concepts (*i.e.* meta-properties) that aid the knowledge engineer to make correct decisions when building ontologies and therefore to avoid common mistakes. But in our approach, instead of attribute properties we mainly focus on the dynamic properties: *Static*, *Rigid*, *Temporary*, *Reversible*, *Non-Reversible*, and *Periodic*. We also investigate how to build up a meta-ontology with the constrains on these meta-properties. We have also shown some primary results on how to improve the reasoning on dynamic data by using these properties. Further enrich our methodology is main future work. Other extensions we are going to add are related to the cyclicality, trend (*e.g.* monotonic or non-monotonic) and variability (*i.e.* frequency of changes) of properties which allow to distinguish entities according their dynamics.

Acknowledgments. This work is partially supported by the FP7 K-Drive project (No. 286348) and the EPSRC WhatIf project (No. EP/J014354/1).

References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web, pp. 635–644. ACM (2011)
2. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. VLDB J. **15**(2), 121–142 (2006)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
4. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-SPARQL: a continuous query language for RDF data streams. Int. J. Semant. Comput. **4**(1), 3–25 (2010)
5. Botan, I., Derakhshan, R., Dindar, N., Haas, L., Miller, R.J., Tatbul, N.: Secret: a model for analysis of the execution semantics of stream processing systems. Proc. VLDB Endow. **3**(1–2), 232–243 (2010)

6. Calbimonte, J.-P., Jeung, H.Y., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. Int. J. Semant. Web Inf. Syst. **8**(1), 43–63 (2012)
7. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proceedings of AAAI 2006, pp. 1295–1300 (2006)
8. Guarino, N.: Concepts, attributes, and arbitrary relations - some linguistic and ontological criteria for structuring knowledge bases. Data Knowl. Eng. **8**, 249–261 (1992)
9. Guarino, N., Welty, C.: An overview of ontoclean. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 151–159. Springer, Heidelberg (2004)
10. Guarino, N., Welty, C.A.: A formal ontology of properties. In: Dieng, R., Corby, O. (eds.) EKAW 2000. LNCS (LNAI), vol. 1937, pp. 97–112. Springer, Heidelberg (2000)
11. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
12. Ren, Y., Pan, J.Z.: Optimising ontology stream reasoning with truth maintenance system. In: Proceedings of the ACM Conference on Information and Knowledge Management (CIKM) (2011)
13. Tamma, V.A.M., Capon, T.J.M.B.: Attribute meta-properties for formal ontological analysis. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 301–316. Springer, Heidelberg (2002)
14. Wang, S., Pan, J.Z., Zhao, Y., Li, W., Han, S., Han, D.: Belief base revision for datalog+/- ontologies. In: Kim, W., Ding, Y., Kim, H.-G. (eds.) JIST 2013. LNCS, vol. 8388, pp. 175–186. Springer, Heidelberg (2014)