Defining Key Semantics for the Semantic Web : a Theoretical View

Michel Chein, Madalina Croitoru, Michel Leclère, Nathalie Pernelle, Fatiha Sais, Danai Symeonidou

Technical Report: Defining Key Semantics for the Semantic Web (A Theoretical View)

Michel Chein [1], Madalina Croitoru [1], Michel Leclere [1], Nathalie Pernelle [2], Fatiha Sais [2], Danai Symeonidou [2]
[1] University of Montpellier 2, France
[2] University of Paris Sud, France

1 Introduction

This technical report presents the theoretical results in defining various key semantics in a Semantic Web context.

Keys are fundamental in relational databases. Keys also figure largely in the Semantic Web (SW), especially since the inclusion in OWL 2 of HasKey construct, which allows to include in an ontology an axiom stating that a collection of data or object properties is a key of a particular class.

Different notions of a key can certainly be useful for datasets cleansing and interlinking. In general, if a set of properties $K = \{p_1, \ldots, p_n\}$ is declared to be a key for a class C, then not respecting the K key within the same dataset may be due to mistakes (e.g. they may be duplicates), whereas not respecting K from separate datasets can be seen as candidates for interlinking. In the following we abusively call "exceptions" the incoherences obtained from not respecting a key K. According to OWL 2 RDF-based semantics of HasKey, an exception to K is a pair of *distinct* named instances x and y of C for which there exist (instances or literals) $o_1, \ldots, o_n$ such that both x and y are related to o_i through p_i $(i = 1, \ldots, n)$.

Note that, even if a property $p_i \in K$ is not functional, it is not required that x and y coincide on all values for p_i. This is in line with a natural assumption currently made by the Semantic Web: whether we consider a high information entropy in new knowledge about x and y or not, they will remain exceptions to K. Nevertheless, for datasets cleansing and interlinking it might also be useful to consider a notion of a key which complies more with the low entropy new knowledge assumptions made by databases. More specifically, for x and y to be exceptions of K we may prefer to impose that x and y coincide on all values for each property $p_i \in K$, i.e. if there exists o_i such that x is related to o_i through p_i then y must also be related to o_i

through p_i, and vice versa. With such a notion of a key, x and y may cease to be exceptions to K if new knowledge about x and y is learnt.

This paper *contributes to formalising different notions of a key* in the context of datasets cleansing and interlinking. This paper does not address the practical issues of (1) algorithmic approaches for key discovery given a particular semantics or (2) the relevance of a semantic over another in the context of applications such as data interlinking or cleansing. These aspects are currently investigated and will be presented in a follow up paper ("Part two").

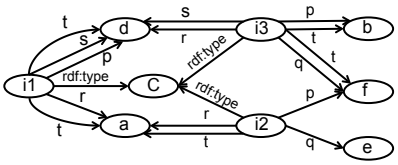
2 Intuition and Example

The key notion has been first introduced in the entity-relationship model. This model allows to describe entity sets and their relationships. Entity sets have attributes which associate with each entity in the set a value from a domain of values for that attribute. A *key* for an entity set is a set of attributes whose values uniquely identify each entity in that entity set.

In the relational data model, entity sets and their relationships are both represented by relations. For a given relation, we distinguish its schema which is the set of attribute names, from its instance which is the set of value tuples (each tuple representing an entity). Each relation must have a key. When no key exists for an entity set or when the keys involve numerous or large value attributes (e.g. an attribute text field) a *surrogate key* (a new attribute whose values are automatically generated and guaranteed pairwise distinct) is added to relation. In relational model, a key is required to be minimal, i.e. any non-empty subset of a key should not be a key.

A natural transformation from relational model to First Order Logic (FOL) consists in choosing an order for attributes of each relation and translating each tuple into an atom having the name of the relation as predicate and the values of the tuple in the chosen order as terms. The instance of the relation is then the conjunction of the obtained atoms. One has to be careful as the Semantic Web (SW) context comes with several fundamental differences from database model:

- First, while, in database model, an entity set is represented in intension by a relation schema and in extension by a set of tuples, in SW, an entity set is represented in intension by a class (or a class expression) and the asserted instances of this class are only a part of its extension.
- Secondly, SW data are not controlled by a schema; this means that nothing forbids to use any vocabulary to describe data. However, a current use consists in declaring the used vocabulary in one or several ontologies. Even under this assumption, an ontology is rarely equivalent to a relational schema: to have this equivalence, we should have



C	p	r	s	q	t
i1	d	a	d		a, d
i2	f	a		e	a
i3	b	d	d	f	b,f

Figure 1: An RDF knowledge base example (KB_1) and its associated multivalued table.

for each class and each property an exact cardinality constraint.

- Thirdly, the SW does not make the Close World Assumption (CWA) that is a non entailed knowledge is false. So, for instance, if an annotation asserts that a resource r owns a value v by a property p, this does not entail that a value $v' \neq v$ is not a associated with r by p.

- Last but not least, SW does not make the Unique Name Assumption (UNA) that is two constants are semantically equal iff they are syntactically identical. So, in SW, two URIs can be asserted or entailed as equal or different.

2.1 Illustrating Example

Whatever the considered notion of key (and its translation in key rule), we are interested about identifying keys for a class in a satisfiable FOL knowledge base (a translation into FOL of an RDF/S + OWL 2 knowledge base). A key can be identified by (i) inference or by (ii) learning from a knowledge base. The former are called inferred keys, and the latter observed keys.

Consider Figure 1 which shows an example of an RDF graph with three instances i_1, i_2 and i_3 of a class C, and its associated table. The translation of this graph into FOL is the fact: $F = C(i_1) \wedge p(i_1, d) \wedge \wedge q(i_1, e) \wedge r(i_1, a) \wedge s(i_1, d) \wedge t(i_1, a) \wedge t(i_1, d) \wedge C(i_2) \wedge p(i_2, f) \wedge r(i_2, a) \wedge q(i_2, f) \wedge t(i_2, a) \wedge C(i_3) \wedge p(i_3, b) \wedge r(i_3, d) \wedge s(i_3, d) \wedge t(i_3, b) \wedge t(i_3, f)$

With a database point of view, we would conclude that $\{p\}$ is an observed key but that $\{r\}$ is not an observed key. And we do not know how consider the three others properties. An "observed key" would be a set of predicates that will ensure that the values of those predicates (the objects) are sufficient to uniquely identify each of the instance (the subjects) occurring in the base. It is however important to make here a remark. By concluding that $\{r\}$ is not an observed key, we make the implicit assumption that i_1 and i_2 are different. Also, by concluding that $\{p\}$ is an observed key, we make the implicit assumption that d and f are different. In fact, in order to observe a key on a fact it is needed to have a **complete** knowledge of the dual equality/difference relation between the set of terms appearing in the fact.

On one hand, we have to know what are the distinct pairs of instances of the class on which the key is observed. On the other hand, we have to know, for the range of each property, what are the identical individuals. If there is no information about these relations, a simple heuristic is to choose the syntactic equality relation (corresponding to the Unique Name Assumption (UNA) in database model). In certain applications it is perfectly reasonable to consider constraints on certain predicates. For instance we can impose that a person does not have more than two surnames. Or that an address can only have a zip code. When such constraints are available (corresponding to what it is called "closed predicates") the semantics of considered keys can be adapted to incorporate them. In the reminder of the article the intuition behind the so called observed keys correspond to what it is denoted as simple keys. The intuition behind the constrained based key notion is denoted as forall keys.

Let us note that in the OWL 2 definition of a key, the "sameAS" is logically interpreted as equality. This is not always relevant on the Linked Data as highlighted by [2]. Introducing Sim relations allows us to further extend our work in the context of real world usages of "sameAs".

If the Sim relations are the syntactic equality relations, with the proposed definition of an observed key, one can also conclude that $\{q\}$ is an observed key, $\{s\}$ is not one, and $\{r, s\}$ is an observed key. Please note that in an inferential setting keys are used in order to infer "sameAs" relations (whatever their intuitive meaning - that is either logical equality or simply object similiarity).

In the following we will formalise the above intuitions and explore theoretically the different possible notions of keys.

3 Logical and Functional Keys in Linked Open Data

3.1 Existing Definitions in the Semantic Web

Two notions of a key exist in the SW: `owl:InverseFunctionalProperty` and `owl:hasKey` axioms of OWL[1]. The `owl:InverseFunctionalProperty` axiom of OWL allows to declare a single property key for an RDF base. Applied to a property p, its logical semantics can be translated in the rule:

$$\forall x \forall y \forall z (p(x, z) \wedge p(y, z) \rightarrow x = y)$$

So, in the SW, the declaration of a set of properties $\{P_1, ...P_n\}$ as a key for a (target) expression class C could be a rule of the form:

[1] `owl:InverseFunctionalProperty` existed in the first version of OWL, `owl:hasKey` was introduced in OWL 2.

$$\mathbf{R1} : \forall x \forall y \forall z_1 ... z_n (C(x) \wedge C(y) \wedge \bigwedge_{i=1}^{n} (P_i(x, z_i) \wedge P_i(y, z_i)) \rightarrow x = y)$$

This definition is coherent with the OWL 2 RDF-Based semantics of `owl:hasKey` (cf. section 5.14 of [7]). Nevertheless, it is not coherent with neither the informal presentation of `owl:hasKey` axiom made in [8] (cf. section 9.5) , nor the OWL 2 direct semantics of `owl:hasKey` (cf. section 2.3.5 of [6]). In these notions an additional condition enforces the considered individuals to be named (*i.e.* they have to be URIs or literals, but not blank nodes). If it is assumed that there is a built-in unary predicate $Const$ which is $true$ for a constant and $false$ otherwise, this second semantics of `owl:hasKey` can be captured with a rule as follows:

$$\mathbf{R2} : \forall x \forall y \forall z_1 ... z_n (C(x) \wedge C(y) \wedge Const(x) \wedge Const(y) \wedge \bigwedge_{i=1}^{n} (P_i(x, z_i) \wedge P_i(y, z_i) \wedge Const(z_i)) \rightarrow x = y)$$

Generalizing this notion, while remaining coherent with OWL, one obtains:

$$\mathbf{R3} : \forall x \forall y \forall z_1 ... z_n (C[x] \wedge C[y] \wedge \bigwedge_{i=1}^{n} (P_i(x, z_i) \wedge P_i(y, z_i)) \rightarrow x = y)$$

where $C[.]$ is a formula with exactly one free variable (also called unary lambda formula). $C[.]$ is named the target expression class of the key.

3.2 Two Generalized Rule Definitions: Some-Rules and Forall-Rules

The use of the similarity predicates here generalises the equality notion. It corresponds to a "real world" usage of the Web where similarities are used rather than "pure logical" equality.

Let $Sim_1, ..., Sim_n$ be similarity predicates used to compare property values. Introducing similarity predicates leads to the following simple extension of the previous key rule $R3$:

Definition 1 (S-Rule) *The **"Some" similarity rule**, or **S-rule** for a class C with respect to $Sim_{S_1}, ..., Sim_{S_n}$ is the rule defined as follows:*

$$\forall x \forall y (C[x] \wedge C[y] \wedge$$

$$\bigwedge_{i=1}^{n} \exists z_i \exists w_i (p_i(x, z_i) \wedge p_i(y, w_i) \wedge Sim_{S_i}(z_i, w_i)) \rightarrow Sim_C(x, y))$$

Sim_{S_i} denotes the similarity predicate used in the rule S for comparing z_i and w_i, i.e. the values of the property p_i (for instance string metrics such as Levenshtein, Jaro-Winkler etc.). The S-rule for a class C with respect to $Sim_1, ..., Sim_n$ is noted S$[(C, Sim_C), (p_1, Sim_{S_1}), ..., (p_n, Sim_{S_n})]$ or, whenever there is no potential confusion, (C, Sim_C), (p_1, Sim_{S_1}), ..., (p_n, Sim_{S_n}).

Example 1 *Let KB_2 be the knowledge base obtained by adding to KB_1 the instance descriptions given in 1. Let assume that $(C, Sim_C), (p, SimSp)$ is a S-Rule and that we know $Sim_p(h, f)$. From this we infer the similarities: $\{Sim_C(i1, i4), Sim_C(i3, i5), Sim_C(i2, i6)\}$*

C	p	r	s	q	t
i4	d, g				
i5	b				
i6	h				
i7					

Table 1: Instance descriptions to be added to KB_1

An S-rule S is an extension of an OWL key in the following sense: if all similarity predicates in K, Sim_{S_i} and Sim_C, are the equality then K and R_3 are equivalent formulas.

Proposition 1 *The S-Rule $S[(C, =); (p_1, =), ..., (p_n, =)]$ is logically equivalent to R_3, i.e. $R_3 \leftrightarrow K$ is valid.*

Note that to prove the above property one needs the substitution property for the logical equality which is not always relevant to uses of `owl:sameAs` and is not assumed in this paper for similarity predicates. In this paper it is proven that the logical semantics of computed keys depends on the algorithm used: it can be either an S-rule or an F-rule (or slight variants) defined as follows.

Definition 2 (F-Rule) *The **"Forall" similarity rule**, or**F-Rule** for a class C with respect to $Sim_{F_1}, ..., Sim_{F_n}$ is the rule defined as follows:*
$\forall x \forall y (C[x] \wedge C[y] \wedge \bigwedge_{i=1}^{n} (\forall z_i \exists w_i (p_i(y, z_i) \rightarrow p_i(x, w_i) \wedge Sim_{F_i}(z_i, w_i)) \wedge$
$(\forall u_i \exists v_i (p_i(x, u_i) \rightarrow p_i(y, v_i) \wedge Sim_{F_i}(u_i, v_i))) \rightarrow Sim_C(x, y))$

Example 2 *Let consider KB_2 defined in Example 1. Let assume now that $(C, Sim_C), (p, SimSp)$ is a F-Rule and that we know that $Sim_p(h, f)$. From this we infer the similarities: $\{Sim_C(i3, i5), Sim_C(i2, i6)\}$*

The F-rule for a class C with respect to $Sim_{F_1}, ..., Sim_{F_n}$ is noted $F[(C, Sim_C), (p_1, Sim_{F_1}), ..., (p_n, Sim_{F_n})]$ or simply (C, Sim_C), (p_1, Sim_{F_1}), ..., (p_n, Sim_{F_n}) whenever there is no potential confusion. The next proposition asserts that both S-rules and F-rules are monotonic w.r.t. the inclusion of their attributes. More precisely,

Proposition 2 *Let R be the S-rule (F-rule)* $[(C, Sim_C), (p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n})]$ *and R' be any S-rule (resp. K-rule)* $[(p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n}), (p_{n+1}, Sim_{R_{n+1}}), ...]$. *Then:* $R \models R'$, *i.e. R logically entails R'.*

The relationships between S-rules and F-rules are studied in the next section.

3.3 S-Rules and F-Rules Models

The rule discovery algorithms presented in Section **??** and assessed in Section **??** are described using set-based notions and properties corresponding to the interpretations satisfying (i.e. to models of) S-rules and F-rules. In this section these notions are defined and it is shown that the logical definitions of S-rules and F-rules are FOL semantics of the rules computed by the algorithms. For simplicity reasons, we consider C a unary predicate.

Definition 3 *Let* $\mathcal{C}$ *be a set,* P *be a binary relation over* $\mathcal{C}$, $\mathcal{S}$ *be a reflexive and symmetric similarity relation defined on the range of* P, *and* $c, c' \in \mathcal{C}$.

- $P(c)$ *denotes the set of elements in* $\mathcal{C}$ *related to* c *by* P, *i.e.* $P(c) = \{u \mid (c, u) \in P\}$;
- $P(c) \subseteq_s P(c')$ *means that for any element* u *in* $P(c)$ *there is an element* v *in* $P(c')$ *such that* u *and* v *are similar w.r.t.* $\mathcal{S}$, *i.e.* $(u, v) \in \mathcal{S}$;
- $P(c) =_s P(c')$ *means that* $P(c) \subseteq_s P(c')$ *and* $P(c') \subseteq_s P(c)$
- $P(c) \cap_s P(c')$ *is equal to the set* $\{u \in P(c) | \exists v \in P(c')$ *such that* $(u, v) \in \mathcal{S}\}$

Example 3 *In* KB_2 *we have, for example,* $p(i1) = \{d\}$, $p(i1) \subseteq_s p(14), p(i5) =_s p(i3)$. *We have also, for example,* $p(i2) \cap_s p(i6) = \{f\}$ *and* $p(i6) \cap_s p(i2) = \{h\}$.

Note that $\cap_s$ is not commutative, nevertheless $P(c) \cap_s P(c') \neq \emptyset$ iff $P(c') \cap_s P(c) \neq \emptyset$. Remark also that if Sim_P is the equality relation then $\subseteq_s$ is the set inclusion, $\cap_s$ is the set intersection and $=_s$ is the set equality.

Definition 4 *A FOL interpretation I with domain* Δ^I *of the predicates occurring in an S-rule, or an F-rule, is given by:*

- $C^I \subseteq \Delta^I$ *the interpretation of the class* C*;*
- $p_i^I \subseteq \Delta^I \times \Delta^I$ *the interpretation of the predicates* p_i*;*
- $Sim_i^I \subseteq \Delta^I \times \Delta^I$ *the interpretation of the predicates* Sim_i*;*
- $Sim_C^I \subseteq \Delta^I \times \Delta^I$ *the interpretation of the predicate* Sim_C*.*

Proposition 3 *An interpretation* I *is a model of the S-rule* (C, Sim_C)*,* (p_1, Sim_{S_1})*, ...,* (p_n, Sim_{S_n}) *iff for any* c *and* c' *in* C^I *such that for any* $i = 1, \ldots, k$*,* $p_i^I(c) \cap_S p_i^I(c') \neq \emptyset$*, one has* $(c, c') \in Sim_C^I$*.*

Note that whenever any property p_i^I is a total function and any similarity relation is the equality relation then a minimal (wrt inclusion of properties) S-rule is a key in the relational database framework.

Proposition 4 *An interpretation* I *is a model of the F-rule* (C, Sim_C)*,* (p_1, Sim_{F_1})*, ...,* (p_n, Sim_{F_n}) *iff for any* c *and* c' *in* C^I *such that for any* $i = 1, \ldots, k$*,* $p_i^I(c) =_S p_i^I(c')$*, one has* $(c, c') \in Sim_C^I$*.*

Relationships between S-rules and F-rules are given by the following proposition. The first property states that the S-rule notion is more restrictive than the F-rule notion when one considers interpretations in which for any p_i^I there is at most one element of C^I with no value. In relational DB it corresponds to the case where in each column there is at most one null value.

The second property states the opposite when one considers interpretations in which any p_i^I is functional. In relational DB it corresponds to the case where there is no multiple values.

Finally, the last property states that these notions are equivalent when the interpretation of any p_i, i.e. p_i^I, is a total function. In the relational DB it corresponds to the case where there is neither column with two null values nor multiple values.

Proposition 5 *Let* S*-*R *and* F*-*R *be respectively the S-rule and the F-rule associated with* (C, Sim_C)*,* (p_1, Sim_{S_1})*, ...,* (p_n, Sim_{S_n}) *.*

- *For any interpretation* I *such as for any* $i = 1, \ldots, n$ *there is at most one element* $c \in C^I$ *with* $p_i^I(c) = \emptyset$*, one has: if* I *is a model of* S*-*R *then* I *is a model of* F*-*R *(i.e. if* $I \models S$*-*R *then* $I \models F$*-*R*).*
- *For any interpretation* I *such as for any element* $c \in C^I$ *and any property* p_i *one has* $card(p_i^I(c)) \leq 1$*, then if* I *is a model of* F*-*R *then* I *is a model of* S*-*R *(i.e. if* $I \models F$*-*R *then* $I \models S$*-*R*).*
- *For any interpretation* I *such as for any element* $c \in C^I$ *and any property* p_i *one has* $card(p_i^I(c)) = 1$*, then* I *is a model of* S*-*R *iff* I *is a model of* F*-*R *(i.e.* $I \models F$*-*R *iff* $I \models S$*-*R*).*

3.4 Using Rules to Infer Similarities in Datasets

In this section, similarities deduced from an S-rule or an F-rule are defined. This inferential use of similarity rules can be used for data linking since conclusion of such a rule is precisely what it is looking for in data linking, i.e. the similarity, or the equality, between two objects.

In what follows, a *vocabulary* V is a FOL vocabulary without symbol functions and containing, at least, a unary predicate C, a binary predicate Sim_C, for any $i = 1, \ldots, k$ a binary predicate p_i, a set of reflexive and symmetrical similarity predicate Sim_1,..., Sim_n, a set of constants and a set of variables.

Definition 5 (Fact and Simple Fact)

- *A* fact relative to a class C *over a vocabulary V is the existential closure of a conjunction of positive atoms built with V such that terms occurring in a Sim_C atom also occurs in C atoms and terms occurring in a Sim_{p_i} atom also occurs in second position of a p_i atom.*
- *A* simple fact *is a fact without Sim_C atom.*

Example 4 *Let F' be the following fact relative to the class C over the vocabulary V_{KB_2}:*

$F' = C(i_1) \wedge p(i_1, d) \wedge q(i_1, e) \wedge r(i_1, a) \wedge s(i_1, d) \wedge t(i_1, a) \wedge t(i_1, d) \wedge C(i_2) \wedge p(i_2, f) \wedge r(i_2, a) \wedge q(i_2, f) \wedge t(i_2, a) \wedge C(i_3) \wedge p(i_3, b) \wedge r(i_3, d) \wedge s(i_3, d) \wedge t(i_3, b) \wedge t(i_3, f) \wedge C(i_4) \wedge p(i_4, d) \wedge p(i_4, g) \wedge C(i_5) \wedge p(i_5, b) \wedge C(i_6) \wedge p(i_6, h) \wedge \exists x\ p(i_7, x) \wedge Sim_S p(f, h)$

Definition 6 (Saturation by S-rule) *Let $\mathcal{F}$ be a fact and S be an S-rule, $R(\mathcal{F})$ is the maximal fact such that: $\mathcal{F}, S \models R(\mathcal{F})$.*

Example 5 *The saturation of F' using the S-Rule $(C, Sim_C), (p, SimSp)$ is: $R(F') = F' \wedge Sim_C(i_1, i_4) \wedge Sim_C(i3, i_5) \wedge Sim_C(i_2, i_6)$.*

Using F-rules in an inferential way needs to complete $\mathcal{F}$ with formulas expressing the closure of some predicates (cf. [11]).

Definition 7 (Closure) *Let t be an instance of a class C and p be a binary predicate. $Closed(t, p)$ is the following formula expressing the closure of $\mathcal{F}$ on p relatively to t:*

$Closed(t, p) = \forall x(p(t, x) \rightarrow (x = t_1) \vee \ldots \vee (x = t_n) \vee \bot)$ where $t_1, \ldots, t_n$ are the terms in the second position of an atom $p(t, -)$ in $\mathcal{F}$.

$Closure(\mathcal{F}, \{p_1, \ldots, p_n\})$ is the existential closure of the conjunction of atoms in $\mathcal{F}$ and the $Closed(t, p_i)$ formulas for each term t occurring in a C atom and each p_i.

Example 6 $Closure(F', \{p\}) = F' \wedge ((x = d) \vee (x = f) \vee (x = b) \vee (x = h) \vee \bot$

Definition 8 (Saturation by F-rule) *Let $\mathcal{F}$ be a fact and F be the F-rule (C, Sim_C), (p_1, Sim_{F_1}), ..., (p_n, Sim_{F_n}) , $F(\mathcal{F})$ is the maximal fact such that:*

$$Closure(\mathcal{F}, \{p_1, ..., p_n\}), F \models F(\mathcal{F})$$

Example 7 *Let $rule_1$ be the F-Rule $(C, Sim_C), (p, SimSp)$. $rule_1(F') = F' \wedge Sim_C(i_3, i_5) \wedge Sim_C(i_2, i_6) \wedge (Sim_C(i_7, i_1) \vee Sim_C(i_7, i_2) \vee Sim_C(i_7, i_3) \vee Sim_C(i_7, i_5) \vee Sim_C(i_7, i_6))$*

Definition 9 (Deduced Similarities by a rule) *Let $\mathcal{F}$ be a fact and R be an S or F-rule. The* deduced similarities *from $\mathcal{F}$ by R, denoted $Sim(R, \mathcal{F})$, is the subset of atoms in $R(\mathcal{F})$ which do not belong to $\mathcal{F}$, i.e. $Sim(R, \mathcal{F}) = R(\mathcal{F}) \setminus \mathcal{F}$.*

Example 8 $Sim(rule_1, F') = Sim_C(i_3, i_5) \wedge Sim_C(i_2, i_6)$

Note that $Sim(R, \mathcal{F})$ produced by a rule R is reflexive and symmetrical (*i.e.* Sim_C is reflexive and symmetrical on the set of terms in $Sim(R, \mathcal{F})$). Moreover, this relation is transitive for F-rules since $=_s$ is transitive. As a final remark, Proposition 2 leads to:

Proposition 6 *Let F be a fact, $R = (C, Sim_C), (p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n})$ and $R' = (C, Sim_C), (p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n}), (p_{n+1}, Sim_{R_{n+1}})$ be S-rules (or F-rules). Then, $Sim(R', \mathcal{F}) \subseteq Sim(R, \mathcal{F})$.*

3.5 Key Discovery

Definition 10 (Key for a fact) *Let $\mathcal{F}$ be a fact and let R be an S or F-rule:*

$$(C, Sim_C), (p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n})$$

R is an S or F-key for C in $\mathcal{F}$ *if $Sim(R, \mathcal{F}) \subseteq \mathcal{F}$, i.e. any deduced similarity with R is in the fact $\mathcal{F}$.*

We will make here a further distinction. Depending if the fact $\mathcal{F}$ contains or not un-instantiated properties one could choose to follow two different interpretations of the missing values. Either one assumes a minimal entropy model (that is, no new information would be added by the missing value if stated), or a non minimal entropy model (potentially much information can be added by the values if stated). Abusively we denote by $p_i(c) \notin \mathcal{F}$ if the instance c has no value for property p_i in the fact $\mathcal{F}$. We then get:

Definition 11 (Minimal Entropy Key for a fact) *Let $\mathcal{F}$ be a fact and let R be an S or F-rule:*

$$(C, Sim_C), (p_1, Sim_{R_1}), ..., (p_n, Sim_{R_n})$$

We denote by $\mathcal{F}'_R$ the set obtained by removing all fully non instantiated instances: $\mathcal{F}_r = \mathcal{F} \setminus \{c | \exists p_i$ such that $p_i(c) \notin \mathcal{F}\}$. R is an minimal entropy S or F-key for C in $\mathcal{F}$ *if it is a* S or F-key for C in $\mathcal{F}'_R$.

Notation: In the following minimal entropy F-keys will be denoted ADS-keys (since they were first introduced, but from a functional view point, by [1]). The notion of ADS keys has also been investigated from a practical view point (to be reported on a follow up paper on implementation aspects).

Let us assume that $\mathcal{F}$ is complete with respect to Sim_C, i.e. that whenever $C(t)$ and $C(t')$ are in $\mathcal{F}$ and $Sim_C(t, t')$ is not in $\mathcal{F}$ is interpreted as the fact that t and t' are not similar. Then, a key should not allow the deduction of such an atom $Sim_C(t, t')$ not in $\mathcal{F}$. This can be stated in logical terms as follows.

Definition 12 (Extended Fact) *An* extended fact *is composed of a fact plus possibly negated Sim_C atoms.*

Definition 13 (Completion of a fact) *An extended fact $\mathcal{F}$ relative to a class C is* complete *if for any couple of terms (t, t'), not necessarily distinct, instance of C, it contains either $Sim_C(t, t')$ or $\neg Sim_C(t, t')$. Let $\mathcal{F}$ a fact, we denote $\mathcal{F}^C$ the* complete extended fact *obtained by adding $\neg Sim_C(t, t')$ atoms for each couple of terms (t, t') such that $Sim_C(t, t')$ is not occurring in $\mathcal{F}$.*

Example 9 $F'^C = F' \wedge \neg Sim_C(i_1, i_2) \wedge \neg Sim_C(i_1, i_3) \wedge ... \wedge \neg Sim_C(i_6, i_7)$

It is straightforward to see that: *An extended fact $\mathcal{F}$ is consistent iff there is no terms t, t' such that atoms $Sim_C(t, t')$ and $\neg Sim_C(t, t')$ both occurs in $\mathcal{F}$.* Note that a complete extended fact is consistent.

Proposition 7 *Let $\mathcal{F}$ be a fact relative to a class C and let S be an S-rule*

$$(C, Sim_C), (p_1, Sim_{S_1}), ..., (p_n, Sim_{S_n})$$

S is an S-key for C in $\mathcal{F}$ iff $(\mathcal{F}^C, S)$ is satisfiable.

Proposition 8 *Let $\mathcal{F}$ be a fact relative to a class C and let F be a F-rule*

$$(C, Sim_C), (p_1, Sim_{F_1}), ..., (p_n, Sim_{F_n})$$

F is an F-key for C in $\mathcal{F}$ iff closure$(\mathcal{F}^C, F), F$ is satisfiable.

4 Existing Work and Discussion

The problem of key discovery from RDF datasets is similar to the key discovery problem in relational databases. In both cases, the key discovery problem is a sub-problem of Functional Dependencies (FDs) discovery. A FD states that the value of one attribute is uniquely determined by the values of some other attributes. Discovering FDs in RDF data differs from the relational case as null values and multi-valued properties are to be considered. Keys or FDs can be used for different purposes. Some approaches focus on finding approximate keys or FDs. Blocking methods aim at using approximate keys to reduce the number of instance pairs that have to be compared by a data linking tool ([4],[9]). In [10], the authors discover (inverse) functional properties from data sources where the unique name assumption (UNA) hypothesis is fulfilled (i.e. non composite keys).

In the context of Open Linked Data, [5] have proposed a supervised approach to learn (inverse) functional properties on a set of reconciled data. In [3] make use of axioms declared in the ontology: functional dependencies, inverse functional dependencies, same as and maximum cardinality to infer equivalence rules for RDF resources.

To conclude, in this paper we have investigated different key semantics as well as their slight variants. We have highlighted their usage in the context of Linked Open Data and shown that key discovery can be useful both for interlinking and dataset cleansing. An interesting and immediate future work direction is to extend this evaluation to more datasets and to study the different key semantics relevance on a large scale.

Current work on the implementation of the above key notions and their practical applicability (relevance, qualitative and quantitative results) is currently under went. These results will be reported on a follow up paper to this one "Defining Key Semantics for the Semantic Web. Part II: An Experimental View".

Acknowledgment

This work has been supported by the Agence Nationale de la Recherche (grant ANR-12-CORD-0012)

References

[1] M. Atencia, J. David, and F. Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *EKAW*, pages 144–153, 2012.

[2] H. Halpin, P. Hayes, and H. S. Thompson. When owl: sameas isn't the same redux: A preliminary theory of identity and inference on the

semantic web. In *Proc of Workshop on Discovering Meaning On the Go in Large Heterogeneous Data*, pages 25–30, 2011.

[3] A. Hogan, A. Zimmermann, J. Umbrich, A. Polleres, and S. Decker. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *J. Web Sem.*, 10:76–110, 2012.

[4] M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, pages 440–445, 2006.

[5] A. Nikolov and E. Motta. Data linking: Capturing and utilising implicit schema-level relations. In *Proceedings of Linked Data on the Web workshop at 19th International World Wide Web Conference(WWW)'2010*, 2010.

[6] W. Recommendation. Owl 2 web ontology language: Direct semantics. In C. G. B. Motik B., Patel-Schneider P. F., editor, *http://www.w3.org/TR/owl2-direct-semantics/*. W3C, 27 October 2009.

[7] W. Recommendation. Owl 2 web ontology language: Rdf-based semantics. In S. M., editor, *http://www.w3.org/TR/owl2-rdf-based-semantics/*. W3C, 27 October 2009.

[8] W. Recommendation. Owl 2 web ontology language: Structural specification and functional-style syntax. In P. B. Motik B., Patel-Schneider P. F., editor, *http://www.w3.org/TR/owl2-syntax/*. W3C, 27 October 2009.

[9] D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *International Semantic Web Conference (1)*, pages 649–664, 2011.

[10] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *The Proceedings of the VLDB Endowment(PVLDB)*, 5(3):157–168, 2011.

[11] G. Wagner. Web rules need two kinds of negation. In *PPSWR*, pages 33–50, 2003.