DECIDE - Detection of Contextual Identity
(Technical Report)

Joe Raad, Nathalie Pernelle, Fatiha Sais

July 30, 2017

Contents

1 Introduction

In this technical report we present a new approach for discovering contextual identity relationships in RDF knowledge bases. The approach aims at detecting identity links that are valid in contexts that can be defined as sub-ontologies of the domain ontology. In this section, we introduce the basic notions and the definitions that are needed to define a contextual identity link. These links are based on the notion of global context. We first present the considered data model in sub-section 1.1 and the problem statement in sub-section 1.2. We define the global contexts and introduce their order relation in sub-section 1.3, and the contextual identity relationship in sub-section 1.4. Then, we present our approach of detecting contextual identity links in section 2. Finally, in section

3 we present the output of $DECIDE$ on some examples, with each example presenting a specific case.

1.1 Knowledge Base

We consider a knowledge base where the ontology is represented in OWL[1] and the data represented in RDF[2]. A knowledge base $\mathcal{B}$ is defined by a couple $(\mathcal{O}, \mathcal{F})$ where:
– the ontology $\mathcal{O} = (\mathcal{C}, \mathcal{DP}, \mathcal{OP}, \mathcal{A})$ is defined by a set of classes $\mathcal{C}$, a set of *owl:DataTypeProperty* $\mathcal{DP}$, a set of *owl:ObjectProperty* $\mathcal{OP}$, and a set of axioms $\mathcal{A}$ (e.g property domains or ranges, subsumption).
– $\mathcal{F}$ is a collection of triples *(subject, property, object)*, that expresses that some relationship, indicated by the property, holds between the subject and object of the triple (between two resources or between a resource and a literal value) [3].

1.2 Problem statement

The problem of detecting contextual identity links can be defined as follows: given a knowledge base $\mathcal{B} = (\mathcal{O}, \mathcal{F})$ and a set $\mathcal{I}^{tc}$ of individuals of a target class tc of the ontology $\mathcal{O}$, find for the set of all individual pairs $(i_1, i_2) \in (\mathcal{I}^{tc} \times \mathcal{I}^{tc})$ the most specific contexts in which (i_1, i_2) are identical.

For instance, if we consider the example depicted in Figure 1, the two individuals $drug1$ and $drug2$ of the target class $Drug$ can be considered as identical in the context where we consider all the ontology's properties except of the property name for the drugs. On other hand, the two individuals $drug2$ and $drug3$ can be considered as identical in two most specific global contexts. The first context is the one in which we consider all the ingredients composing the drugs and for every ingredient we consider its weight. However, in this first context, the description of a weight is reduced to the measure unit without considering the quantity (property $hasValue$). A second context in which these individuals are identical is the context where we consider the weight of *Paracetamol* described by its value and its measure unit, and only the presence of *Lactose* in the drugs without considering their weight. These contexts also represent the most specific contexts in which $drug1$ and $drug3$ are identical.

We are interested in the search of the most specific global contexts involving a subset of classes, and we consider for every class a subset of properties. Some contexts can be obviously more relevant than others (e.g. a value of the weight without its measure unit does not have sense). Hence, we also aim to take into account some expert knowledge that can be represented as a set of constraints on the classes and/or properties that should or should not be involved in the considered contexts.

[1] `https://www.w3.org/OWL/`
[2] `https://www.w3.org/RDF/`
[3] We do not consider blank nodes in this work

1.3 Contexts

The contextual identity link detection is based on identifying the most specific global contexts where two instances are identical. We consider a global context as a connected sub-ontology of $\mathcal{O}$ which represents the vocabulary on which two instances are considered as identical. We first introduce the set of classes $DepC$ that can be involved in the contexts. Then, we formally define the global contexts and the contextual identity relation, named $identiConTo$, that expresses that two instances are identical in a given global context.

A global context is represented as a subset of classes and properties of the ontology, and a set of axioms which is limited to constraints on property domains and ranges. We automatically choose the abstraction level of the classes involved in a global context by selecting, from the instantiated classes (direct types), the most general ones.

Definition 1.1. Selected classes $DepC$. The set of selected classes $DepC$ that can be involved in the contextual identity links is the subset of instantiated classes c_i of $\mathcal{B}$ such that:

$$DepC = \{c_i \in \mathcal{C} \mid \nexists c_j \in \mathcal{C} \ s.t. \ \exists x, directType(x, c_j) \ and \ c_i \sqsubset c_j\}$$

Example 1. In Figure 1, $DepC$ will contain all the classes of the graph except of $Ingredients$ which is not instantiated. Therefore, $par1$ and $lac1$ will be uniquely considered as of type $Paracetamol$ and $Lactose$ respectively.

Definition 1.2. Global Context. A global context is a sub-ontology GC_u=(C_u, DP_u, OP_u, A_u) of $\mathcal{O}$ such that $C_u \subseteq DepC$, $DP_u \subseteq DP$, $OP_u \subseteq OP$ and A_u is a set of domain and range constraints that are more specific than those described in A: $\forall p \in OP_u \cup DP_u$, $domain_u(p) \sqsubseteq domain_{\mathcal{O}}(p)$ and $\forall p \in OP_u, range_u(p) \sqsubseteq range_{\mathcal{O}}(p)$

Example 2. In Figure 1, there exists many possible global contexts. We present one:
GC_1=($C = \{Drug, Paracetamol, Lactose, Weight\}$,
$OP = \{isComposedOf, hasWeight\}$, $DP = \{hasValue\}$,
$A = \{domain(isComposedOf) = Drug$,
$range(isComposedOf) = Lactose \sqcup Paracetamol$,
$domain(hasWeight) = Lactose \sqcup Paracetamol$,
$range(hasWeight) = Weight\}$)

Definition 1.3. Order relation between global contexts. Let $GC_u = (C_u, OP_u, DP_u, A_u)$ and $GC_v = (C_v, OP_v, DP_v, A_v)$ be two global contexts. The context GC_u is more specific than GC_v, noted $GC_u \leq GC_v$, if $C_v \subseteq C_u$, $OP_v \subseteq OP_u$, $DP_v \subseteq DP_u$ and
$\forall p \in OP_v \cup DP_v$, $domain_u(p) \sqsubseteq domain_v(p)$, and $\forall p \in OP_v$, $range_v(p) \sqsubseteq range_u(p)$.

In order to filter out the irrelevant contexts to consider, we take in consideration the experts' knowledge when it is available. An expert can supply three

types of constraints:
– *Unwanted properties*(UP): this refers to properties that an expert wants to discard in the detection of contextual identity links. Such constraints can be used when property values correspond to unstructured (free) text, or are known to be particularly heterogeneous, or when the property subjects or objects are evolutive or insignificant to compare two instances for a given task. In such cases, an expert can declare that a property p is unwanted for a given domain c_i (or a particular range c_j) by adding a constraint $up = (c_i, p, *)$ (resp. $up = (*, p, c_j)$) in UP. When a property is unwanted in all domains and ranges, the constraint $(*, p, *)$ can be used. In such cases, $p \notin OP \cup DP$.
– *Necessary properties*(NP): a necessary property np is a constraint noted $(c_i, p, *)$ or $(*, p, c_j)$. When such constraints are added to NP, we will only consider global contexts where the property $p \in OP$ or $p \in DP$, and such that $c_i \in domain(p)$ (resp. $c_j \in range(p)$).
– *Co-occurring properties*(CP): a co-occurrence constraint $cp = \{(c_i, p_1, *), ..., (c_i, p_n, *)\}$ can be declared to guarantee that a certain class c_i will be either declared as the domain (or range) of *all* the properties indicated in the constraint, or *none* of them. For instance, to declare that the weight's value has no meaning without its measure unit, an expert can add the constraint $cp_1 = \{(Weight, hasValue, *), (Weight, hasUnit, *)\}$.

1.4 Contextual identity links

In our approach, two instances are considered identical in a given context, when all the information described in the context is known and represented in the instances' descriptions, and when these descriptions are equal. Therefore, we firstly define the contextual description that is considered for one instance in one context. Then we will define the conditions that must hold to consider that two RDF descriptions refer to identical instances in a given context.

Definition 1.4. Contextual instance description according to a global context. Given a RDF dataset $\mathcal{F}$, a global context $GC_u = (C_u, OP_u, DP_u, A_u)$ and an instance i, a contextual description G_i of i in GC_u is the maximal set of triples that describe i in $\mathcal{F}$ such that:
– G_i forms a connected graph that contains at least one triple where i is subject or object.
– $\forall$ j a class instance of G_i, and $\forall p_1 \in OP_u \cup DP_u$ such that $type(j) \sqsubseteq domain(p_1)$ then $\exists$ $t_a = < j, p_1, l > \in G_i$, and $\forall p_2 \in OP_u \cup DP_u$ such that $type(j) \sqsubseteq range(p_2)$ then $\exists$ $t_b = < l, p_2, j > \in G_i$
– $\forall$ $t = < s, p, o > \in G_i$ then $p \in OP_u \cup DP_u$ and $type(s) \sqsubseteq domain_u(p)$ and $type(o) \sqsubseteq range(p)$

From two contextual descriptions of two class instances defined in a given context, we can define if they can be considered as identical. In this work we will consider that properties are local complete: if a property p is instantiated for a given class instance i, we consider that all the property instances are known for i. Since a local completness is assumed, two instances can be considered

as identical when the contextual graphs, formed by the contextual descriptions, are isomorphic up to a renaming of the instance URI. Note that since some classes can be removed from the global context, this constraint can in fact be considered class by class.

Definition 1.5. Identity in a global context. Given a global context GC_u, a pair of instances (i_1, i_2) are identical in GC_u, noted $identiConTo_{<GCu>}(i_1, i_2)$, only if the two graphs G_{i_1} and G_{i_2} that represent the contextual description of i_1 and i_2 are isomorphic up to a rewriting of the URI of the class instances.

Example 3. $juice1$ and $juice2$ are considered as identical according to the global context GC_1 defined in Example 2.
(i.e. $identiConTo_{<GC_1>}(juice1, juice2)$).

The contextual identity relations will only be specified for the most specific global contexts, but can be inferred for the more general ones using the order relation between global contexts: given GC_u and GC_v two global contexts, with $GC_u \leqslant GC_v$, then $identiConTo_{<GC_u>}(i_1, i_2) \Rightarrow identiConTo_{<GC_v>}(i_1, i_2)$.

2 Algorithm

The goal of the algorithm $DECIDE$ (DEtection of Contextual IDEntity) is to determine for every pair of individuals $(i_1, i_2) \in I \times I$ of a target class tc given by the user, the set of the most specific global contexts in which the identity relation $identiConTo$ is true. $DECIDE$ requires to have a knowledge base $\mathcal{B}$ and a target class tc as inputs, and may consider different constraint lists given by experts. This algorithm, which is implemented in $Java$ and available on `https://github.com/raadjoe/DECIDE_v2`, is composed of four main steps:

2.1 Classes Selection

$DECIDE$ starts by collecting the set of classes $DepC$ that can be involved in the contexts, using the Algorithm 1.

2.2 Local Identity Generation

Now that the selected classes are collected, $DECIDE$ identifies for each pair of individuals in the target class, the most specific local context in which these pairs are identical. This set of pairs of individuals of the target class, with their most specific local context is noted TC, with each pair noted $(i_1, i_2)_j^{tc}$, and it corresponding most specific local context noted $\pi_j(tc)$, with $1 \leq j \leq \dfrac{n \times n - 1}{2}$, and n = number of instances of the target class.

Definition 2.1. (Local Context). Given a class c, a local context $\pi_i(c)$ is a set of triples $(s_\pi,\ p_\pi,\ o_\pi)$ where subject s_π is a class variable y that can

Algorithm 1: getDepClasses

Input: $\mathcal{F}$: the set of facts of the considered knowledge base
Output: *DepC*: the set of classes that can be involved in the contextual identity links

```
1  DepC ← ∅ ;
2  addClass ← false ;
3  foreach (s, rdf:type, o) ∈ F do
4      if DepC isEmpty then
5          DepC.add(o);
6      else
7          foreach c ∈ DepC do
8              if (rdfs:subClassOf(c, o) then
9                  DepC.remove(c);
10                 addClass ← true ;
11         if addClass == true then
12             DepC.add(o);
13 return DepC;
```

be mapped to a class instance, p_π is a property in which i^c, an instance of c appears as subject or object, and o_π is a class variable y that can be mapped to a class instance or to a literal variable $y*$. The class variable y is labelled by its corresponding class c^y, where $c^y \in DepC$.

For instance, one of the local contexts of the class $Weight$ in Figure 1 is: $\pi_1(Weight) = \{(Weight, hasValue, v*), (Weight, hasUnit, u*), (Paracetamol, hasWeight, Weight)\}$. This local context indicates that for each triple in $\pi_1(Weight)$ such as $(Weight, hasUnit, u*)$, $\exists$ a triple in $\mathcal{F}$, which has as subject an instance of type $Weight$, as predicate the property $hasUnit$, and as object a literal value.

Definition 2.2. (Order relation between local contexts). Let $\pi_i(c)$ and $\pi_j(c)$ be two local contexts for the class c. The context $\pi_i(c)$ is more specific than $\pi_j(c)$, noted $\pi_i(c) \leqslant \pi_j(c)$, only if $\forall$ $(s_\pi, p_\pi, o_\pi) \in \pi_j(c)$, $\exists$ $(s_\pi, p_\pi, o_\pi) \in \pi_i(c)$.

For instance in Figure 1, $\pi_1(Weight) \leqslant \pi_2(Weight)$, with $\pi_2(Weight) = \{(w, hasValue, v*), (w, hasUnit, u*)\}$.

Definition 2.3. (Identity in a local context). Given a local context $\pi_i(c)$ for a class c, a pair of instances (i_1, i_2) of c are identical in $\pi_i(c)$ noted $identiConTo_{<\pi_i(c)>}(i_1, i_2)$ only if $\forall$ $(s_\pi, p_\pi, o_\pi) \in \pi_i(c)$ there exists a mapping in $\mathcal{B}$ according to the variable's type in o_π:

- if o_π is a literal variable $y*$, then $\forall$ $(i_1, p_1, v_1) \in \mathcal{F}$, $\exists$ a unique triple (i_2, p_1, v_2) with $v_1 = v_2$. For simplicity, we require in this paper the

exact match between two literal values, nonetheless we can easily relax the constraints to similarity match.

- if o_π is a class variable y, then $\forall$ (i_1, p_1, y_1) $\in \mathcal{F}$, $\exists$ a unique triple (i_2, p_1, y_2). y_1 and y_2 are class instances, and do not necessarily need to be the same, just the fact that both instances i_1 and i_2 have the same relationship in common, with the same number of objects of type y is enough.

For instance in Figure 1, $w1$ and $w3$ are identical in $\pi_1(Weight)$ previously defined, noted $identiConTo_{<\pi_i(Weight)>}(w1, w3)$, since $\forall$ ($w1$, $hasValue$, v_1), $\exists$ a unique triple ($w3$, $hasValue$, v_2) with $v_1 = v_2$. The same condition applies for the property $hasUnit$. Also $\forall$ ($par1$, $hasWeight$, $w1$) $\exists$ a unique triple ($par2$, $hasWeight$, $w3$)

Algorithm 2: generateLocalIdentity

Input:
– tc: the target class
– $DepC$: the set of classes that can be involved in the contextual identity links
– $\mathcal{F}$: the set of facts of the considered knowledge base
Output: TC: the set of pairs of individuals of the target class with their most specific local context

```
1  TC ← ∅ ;
2  I^tc ← ∅ ;
3  j = 1 ;
4  foreach (s, rdf:type, tc) ∈ F do
5      I^tc.add(s);
6  foreach (i1, i2) ∈ I^tc × I^tc with i1 ≠ i2 do
7      tc(j) ← (i1, i2) ;
8      π_j(tc) ← getMostSpecificLocalContext((i1, i2), tc DepC, F) ;
9      TC.add(tc(j), π_j(tc)) ;
10     j++ ;
11 return TC;
```

2.3 Global Identity Detection

After generating the set of pairs of individuals of the target class TC with their most specific local context, we want to detect the most specific global contexts in which each pair $tc(i)$ is identical.

Definition 2.4. (Global Context). Given a local context $\pi_i(tc)$ for a target class tc with $tc \in DepC$, a global context $\Pi(tc)$ is a set of local contexts $\pi_i(c_n)$,

Algorithm 3: getMostSpecificLocalContext

Input:
– (i_1, i_2): pair of individuals of the class c
– c: a class $\in DepC$
– $DepC$: the set of classes that can be involved in the contextual identity links
– $\mathcal{F}$: the set of facts of the considered knowledge base
Output: $\pi_j(c)$: the most specific local context in which (i_1, i_2) are identical

```
1  π_j(c) ← ∅ ;
2  foreach p_n such as ∃ (i_1, p_n, O_{i_1}) and (i_2, p_n, O_{i_2}) ∈ F do
3      if O_{i_1} is a set of literal values then
4          if O_{i_1}.size() == O_{i_2}.size() then
5              π_j(c).add(y^c, p_n, v*) ;
6      else
7          C ← ∅ ;
8          foreach (o_{i_1}, rdf:type, c_i) ∈ F with o_{i_1} ∈ O_{i_1} do
9              if c_i ∈ DepC and c_i ∉ C then
10                 C.add(c_i) ;
11                 n_1 ← number of instances ∈ O_{i_1} of rdf:type c_i ;
12                 n_2 ← number of instances ∈ O_{i_2} of rdf:type c_i ;
13                 if n_1 == n_2 then
14                     π_j(c).add(y^c, p_n, y^{c_i}) ;
15 foreach p_n such as ∃ (O_{i_1}, p_n, i_1) and (O_{i_2}, p_n, i_2) ∈ F do
16     C ← ∅ ;
17     foreach (o_{i_1}, rdf:type, c_i) ∈ F with o_{i_1} ∈ O_{i_1} do
18         if c_i ∈ DepC and c_i ∉ C then
19             C.add(c_i) ;
20             n_1 ← number of instances ∈ O_{i_1} of rdf:type c_i ;
21             n_2 ← number of instances ∈ O_{i_2} of rdf:type c_i ;
22             if n_1 == n_2 then
23                 π_j(c).add(y^{c_i}, p_n, y^c) ;
24 return π_j(c);
```

where $c_n \in DepC$, and $\exists\ \pi_i(c_k) \in \Pi(tc)$ where $\pi_i(c_k)$ contains a triple of form: (s_π, p_π, o_π). Where subject s_π is a variable that can be mapped to an instance i^{c_k} of c_k, p_π is a property in which i^{c_k} appears as subject, or the inverse of a property in which an i^{c_k} appears as object, and o_π is an entity variable that can be mapped to a class instance of c_n.

In the example presented in Figure 1, there exists many possible global contexts for the target class *Drug*. We present one:

$\Pi_1(Drug) = \{\pi_1(Drug),\ \pi_1(Paracetamol),\ \pi_1(Lactose),\ \pi_1(Weight)\}$ where $\boldsymbol{\pi_1(Drug)} = \{(Drug, isComposedOf, Paracetamol),$
$(Drug, isComposedOf, Lactose)\}$, $\boldsymbol{\pi_1(Paracetamol)} =$
$\{(Paracetamol, isComposedOf^{-1}, Drug),$
$(Paracetamol, hasWeight, Weight)\}$, $\boldsymbol{\pi_1(Lactose)} =$
$\{(Lactose, isComposedOf^{-1}, Drug),\ (Lactose, hasWeight, Weight)\}$,
$\boldsymbol{\pi_1(Weight)} = \{(Weight, hasWeight^{-1}, Paracetamol),$
$(Weight, hasWeight^{-1}, Lactose),\ (Weight, hasValue, v*),$
$(Weight, hasValue, u*)\}$.

Definition 2.5. (Order relation between global contexts). Let $\Pi_i(tc)$ and $\Pi_j(tc)$ be two global contexts for the target class tc. The context $\Pi_i(tc)$ is more specific than $\Pi_j(tc)$, noted $\Pi_i(tc) \leqslant \Pi_j(tc)$, only if $\forall\ \pi_j(c) \in \Pi_j(tc)$, $\exists$ $\pi_i(c) \in \Pi_i(tc)$, where $\pi_i(c) \leqslant \pi_j(c)$.

Thanks to this order relation between local contexts, the set of all local contexts of a class c can be represented as a lattice of local contexts noted $T(c)$. In order to filter out the irrelevant local contexts to consider (and eventually the global contexts), we take in consideration the experts' knowledge when it is available. This knowledge that can be expressed as positive or negative inputs concerns the uselessness or the necessity of some properties, or information on the importance of the co-occurrence of some properties. More precisely, for some classes, an expert can supply three types of constraints:
– *Unwanted Patterns*: given a class c, an unwanted pattern for c is a triple $up^c = (s_\pi,\ p_\pi,\ o_\pi)$, in which $\nexists\ \pi_i(c) \in T(c)$ such as $up^c \notin \pi_i(c)$. With these patterns, we can discard in the detection of identity links the unstructured (free text), heterogeneous, and insignificant properties. Adding one unwanted property such as $up^{Weight} = (Lactose, hasWeight, Weight)$ can reduce the number of possible local contexts in $T(Weight)$ to half.
– *Necessary Patterns*: given a class c, a necessary pattern for c is a triple $np^c = (s_\pi,\ p_\pi,\ o_\pi)$, where $\forall \pi_i(c) \in T(c)$, $np^c \in \pi_i(c)$. With these patterns, we can guarantee the presence of some important properties in the identity context, by eliminating all the contexts in $T(c)$ that do not include such patterns.
– *Co-occurring Patterns*: given a class c, a co-occurring pattern CP^c for c is a set of triples $cp_i^c = (s_\pi,\ p_\pi,\ o_\pi)$, in which $\nexists \pi_i(c) \in T(c)$ where $cp_i^c \in \pi_i(c)$ and $cp_j^c \notin \pi_i(c)$. With these patterns, we can guarantee that some properties will not occur in a context without the presence of other properties. For instance,

the weight's value has no sense to occur in a context without its unit of measure. In that case, $CP^{Weight} = \{(Weight, hasValue, v*), (Weight, hasUnit, u*)\}$.

Definition 2.6. (Identity in a global context). Given a global context $\Pi_i(tc)$ for a target class tc, a pair of instances of tc (i_1, i_2) are identical in $\Pi_i(tc)$ noted $identiConTo_{<\Pi_i(tc)>}(i_1, i_2)$ only if $\forall\ \pi_k(c_n) \in \Pi_i(tc)$, $\exists$ a pair of individuals (n_1, n_2) of c_n, where $identiConTo_{<\pi_k(c_n)>}(n_1, n_2)$.

Algorithm **??** detects for each pair in the set of pairs of the target class, the most specific global context (s) in which this pair is identical. This algorithm takes as input the target class, the output of the Algorithm 2 which is the set of pairs of individuals of the target class with their most specific local context, the set of classes that can be involved in the contextual identity links, and the set of facts of the considered knowledge base.

Algorithm 4: IdentityGraphConstruction

Input:
– tc: the target class
– (i_1, i_2): pair of individuals of the class tc
– $depC$: the set of classes collected in the preprocessing
– $\mathcal{F}$: the set of facts of the considered knowledge base
Output: $IGset$: set of the possible identity graphs for (i_1, i_2)

```
1 IG_1 ← ∅; LCset ← ∅; IGset ← ∅ ;
2 n(i_1, i_2) ← emptyGraphNodeForIndivPair(i_1, i_2) ;
3 IGset.add(IG_1) ;
4 foreach IG ∈ IGset do
5     IG ←
        ExpandIdentityGraph(n(i_1, i_2), LCset, IG, IGset, DB, depC, F);
6 return IGset;
```

3 Example

In this section, we present the output of $DECIDE$ with different examples, with each one presenting a specific case such as:

- Multi-valued properties, with the properties' objects having different types
- Multi-valued properties, with the properties' objects having the same type
- Different properties with the same object

Algorithm 5: ExpandIdentityGraph

Input:
– $n(i_1, i_2)$: a graph node of a pair of individuals
– IG: the identity graph
– $IGset$: set of the possible identity graphs for (i_1, i_2)
– $depC$: the selected classes
– $\mathcal{F}$: the set of facts of the considered knowledge base

Output: IG: the expanded identity graph

```
1  c ← getDepClass((i1, i2), depC) ;
2  lc_max(c) ← getMostSpecificLocalContext(i1, i2, F);
3  if ∃ lc(c) ∈ LCset, with lc(c) ⋠ lc_max(c) then
4      lc_max ← getHighestCommonLocalContext(lc_max(c), lc(c)) ;
5  LCset.add(lc_max) ;
6  n(i1, i2).add(lc_max) ;
7  IG.add(n(i1, i2)) ;
8  foreach (p ∈ lc_max) do
9      if rdf:type(p, owl : ObjectProperty) then
10         vals_i1 ← getValuesWithSameDepC(p, i1, F) ;
11         vals_i2 ← getValuesWithSameDepC(p, i2, F) ;
12         CMB ← getAllNodesCombinations(vals_i1, vals_i2) ;
13         IGset' ← ∅ ;
14         foreach (cmb ∈ CMB) do
15             n(cmb) ← emptyGraphNodeForAllPairs(j1, j2) ∈ cmb ;
16             foreach (IG ∈ IGset) do
17                 IG' ← IG ;
18                 IG'.add(n(cmb)) ;
19                 IGset'.add(IG') ;
20             IGset' ← ExpandIdentityGraph(n, IG', IGset', depC, F) ;
21 return IGset;
```

3.1 Multi-valued properties, with the properties' objects having different types

In this section, we present the outcome of $DECICE$ in the case of multi-valued properties, with the properties' objects having different types. Figure 1 presents three drugs, with each drug is composed of one instance of type $Paracetamol$ and one instance of type $Lactose$.

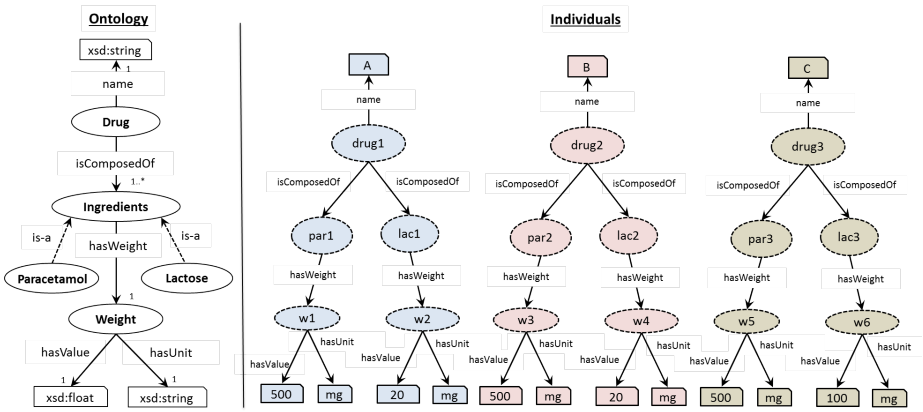

Figure 1: Example 1

Input: $Example1.rdf$, $Drug$
Output: $identiConTo_{<GC_1(Drug)>}(drug1, drug2)$,
$identiConTo_{<GC_2(Drug)>}(drug1, drug3)$, $identiConTo_{<GC_3(Drug)>}(drug1, drug3)$,
$identiConTo_{<GC_2(Drug)>}(drug2, drug3)$, $identiConTo_{<GC_3(Drug)>}(drug2, drug3)$,

with GC_1=($C = \{Drug, Paracetamol, Lactose, Weight\}$,
$OP = \{isComposedOf,\ hasWeight\}$, $DP = \{hasValue, hasUnit\}$,
$A = \{domain(isComposedOf) = Drug$,
$range(isComposedOf) = Lactose \sqcup Paracetamol$,
$domain(hasWeight) = Lactose \sqcup Paracetamol$,
$range(hasWeight) = Weight$,
$domain(hasValue) = Weight$,
$range(hasValue) = Literal$,
$domain(hasUnit) = Weight$,
$range(hasUnit) = Literal\})$

and GC_2=($C = \{Drug, Paracetamol, Lactose, Weight\}$,
$OP = \{isComposedOf,\ hasWeight\}$, $DP = \{hasValue, hasUnit\}$,
$A = \{domain(isComposedOf) = Drug$,
$range(isComposedOf) = Lactose \sqcup Paracetamol$,
$domain(hasWeight) = Paracetamol$,
$range(hasWeight) = Weight$,

$domain(hasValue) = Weight$,
$range(hasValue) = Literal$,
$domain(hasUnit) = Weight$,
$range(hasUnit) = Literal\})$

and GC_3=$(C = \{Drug, Paracetamol, Lactose, Weight\}$,
$OP = \{isComposedOf,\ hasWeight\}$, $DP = \{hasUnit\}$,
$A = \{domain(isComposedOf) = Drug$,
$range(isComposedOf) = Lactose \sqcup Paracetamol$,
$domain(hasWeight) = Lactose \sqcup Paracetamol$,
$range(hasWeight) = Weight$,
$domain(hasUnit) = Weight$,
$range(hasUnit) = Literal\})$

3.2 Multi-valued properties, with the properties' objects having the same type

In this section, we present the outcome of $DECICE$ in the case of multi-valued properties, with the properties' objects having the same type. Figure 2 presents three drugs with the same ontology of Figure 1. However in this example, $drug1$ and $drug2$ are composed of two instances of type $Paracetamol$ and one instance of type $Lactose$, while $drug3$ is composed of two instances of type $Paracetamol$ and two instances of type $Lactose$.

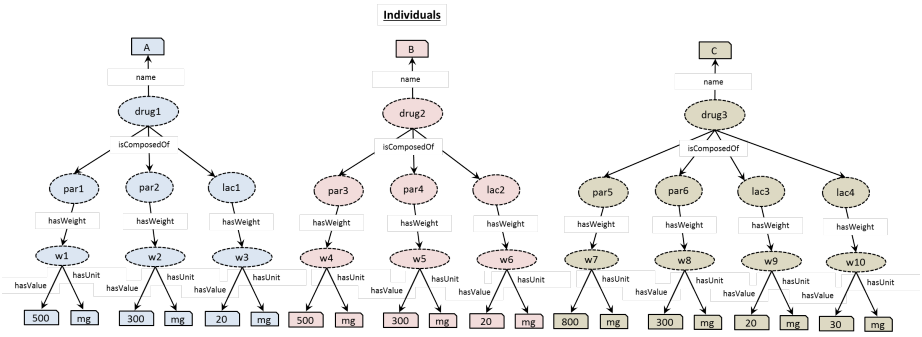

Figure 2: Example 2

Input: $Example2.rdf$, $Drug$
Output: $identiConTo_{<GC_4(Drug)>}(drug1, drug2)$,
$identiConTo_{<GC_5(Drug)>}(drug1, drug3)$,
$identiConTo_{<GC_5(Drug)>}(drug2, drug3)$

with GC_4=$(C = \{Drug, Paracetamol, Lactose, Weight\}$,
$OP = \{isComposedOf,\ hasWeight\}$, $DP = \{hasValue, hasUnit\}$,

$A = \{domain(isComposedOf) = Drug,$
$range(isComposedOf) = Lactose \sqcup Paracetamol,$
$domain(hasWeight) = Lactose \sqcup Paracetamol,$
$range(hasWeight) = Weight,$
$domain(hasValue) = Weight,$
$range(hasValue) = Literal,$
$domain(hasUnit) = Weight,$
$range(hasUnit) = Literal\})$

and GC_5=$(C = \{Drug, Paracetamol, Weight\},$
$OP = \{isComposedOf,\ hasWeight\},\ DP = \{hasUnit\},$
$A = \{domain(isComposedOf) = Drug,$
$range(isComposedOf) = Paracetamol,$
$domain(hasWeight) = Paracetamol,$
$range(hasWeight) = Weight,$
$domain(hasUnit) = Weight,$
$range(hasUnit) = Literal\})$

3.3 Different properties with the same object

In this section, we present the outcome of $DECICE$ in the case of different properties having the same object. Figure 3 presents three drugs with their corresponding companies.

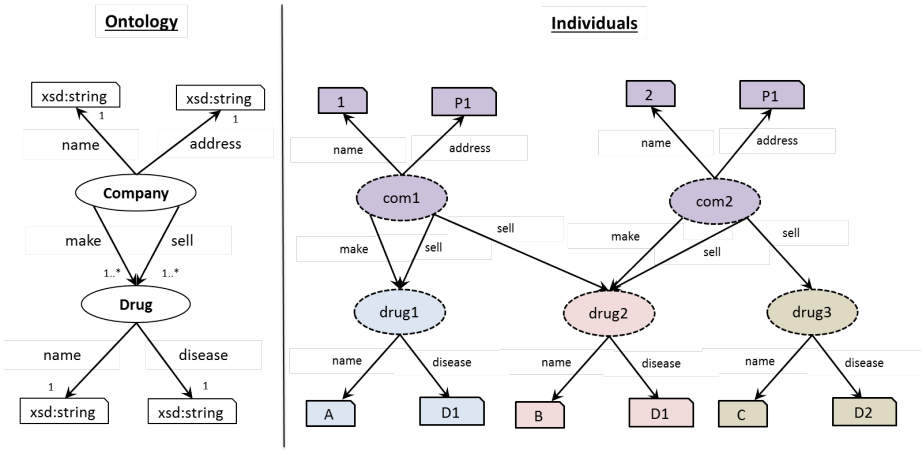

Figure 3: Example 3

Input: $Example3.rdf$, $Drug$
Output: $identiConTo_{<GC_6(Drug)>}(drug1, drug2)$,
$identiConTo_{<GC_7(Drug)>}(drug1, drug3)$,

$identiConTo_{<GC_8(Drug)>}(drug2, drug3)$

with GC_6=($C = \{Drug, Company\}$,
$OP = \{make\}$, $DP = \{address, disease\}$,
$A = \{domain(make) = Company$,
$range(make) = Drug$,
$domain(address) = Company$,
$range(address) = Literal$,
$domain(disease) = Drug$,
$range(disease) = Literal\}$)

and GC_7=($C = \{Drug, Company\}$,
$OP = \{sell\}$, $DP = \{address\}$,
$A = \{domain(sell) = Company$,
$range(sell) = Drug$,
$domain(address) = Company$,
$range(address) = Literal\}$)

and GC_8=($C = \{Drug\}$, $OP = \{\emptyset\}$, $DP = \{\emptyset\}$, $A = \{\emptyset\}$)