Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

Generating Referring Expressions with OWL2

Yuan Ren, Kees van Deemter, and Jeff Z. Pan

Department of Computing Science
University of Aberdeen
Aberdeen, UK

Abstract. The task of generating referring expressions, an important subtask of Natural Language Generation is to generate phrases that uniquely identify domain entities. Until recently, many GRE algorithms were developed using only simple and essentially home-made formalisms. Following the fast development of ontology-based systems, reinterpretations of GRE in terms of description logic have been studied. However, the quantifiers generated are still limited, not exceeding the works covered by existing GRE approaches. In this paper, we propose an DL-based approach to GRE that exploits the full power of OWL2 to generate referring expressions that goes beyond the expressivity of previous GRE algorithms. The potential of DL reasoning in GRE is also discussed.

1 GRE and KR: the story so far

Generation of Referring Expressions (GRE) is the subtask of Natural Language Generation (NLG) that focuses on the identification of objects in natural language. For example, Fig.1 depicts the relations between several individuals of women, dogs and cats. In such a scenario, a GRE system might identify a given object as "Dog" or, if this fails to identify the referent uniquely, "the Dog that loves a Cat", which is literally unique for $d1$. Reference has long been a key issue in theoretical linguistics and psycholinguistics, and GRE is a crucial component of almost every practical NLG system [5, 4]. Accordingly, GRE has become one of the best developed areas of NLG, with links to many other areas of Cognitive Science.

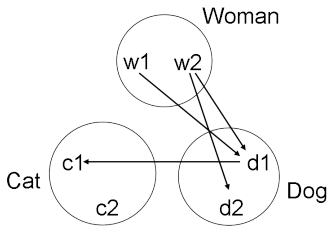

Fig. 1. An example of women, dogs and cats, in which edges from women to dogs denote $feed$ relations, from dogs to cats denote $love$ relations.

Traditional GRE algorithms are usually based on very simple, essentially home-made, forms of Knowledge Representation (KR); in many cases all properties are ex-

pressed in a simple $\langle Attribute : Value \rangle$ format, e.g $\langle Type : Dog \rangle$. This is perhaps justifiable as long as the properties expressed by these algorithms are simple one-place predicates (being a cat, being a Dog, etc.), but when logically more complex descriptions are involved, the potential advantages of "serious" KR become overwhelming. A few proposals in recent years have started to combine GRE with well-developed KR. Following on from earlier work based on labelled directed graphs [9, 14], for example, analyzed GRE as a *projection* problem in Conceptual Graphs. More recently, [1] reinterpreted GRE as a problem in Description Logic (DL), producing a formula such as $Dog \sqcap \exists love.Cat$ to refer to the (unique) Dog who loves at least one Cat. It is this last approach that forms the basis of the present paper, which aims to show that when a principled, logic based approach is chosen, it becomes possible to refer to objects which no existing approach to GRE has so far been able to refer to.

In doing so, we shall follow Areces et al (and many other researchers in GRE) in focussing on the semantic core of the GRE problem: we shall be content to generate descriptions of logical/semantic content, leaving the decision of what English (or other languages) words to use for expressing this content (e.g., 'the ancient dog', or 'the dog which is old') to later stages in the NLG pipeline. Secondly, we shall assume that all domain objects are equally salient [8]. Perhaps most importantly, we do not consider here the important matter of the naturalness or efficacy of the descriptions generated. We shall be content proposing an algorithm that produces uniquely referring expressions whenever such expressions are possible, leaving the choice of the *optimal* referring expression in each given situation for later.

In what follows we start by explaining how DL has been applied in GRE (Sec.2), pointing out the limitations of existing works. In Sec.3 we discuss which kinds of additional expressivity are required and how they can be achieved through modern DL. In Sec.4 we present an generic algorithm to compute these expressive REs. Sec.5 concludes the paper by comparing its aims and achievements with current practise in GRE.

2 DL for GRE

2.1 Description Logics

Description Logic (DLs) come in different flavours, based on decidable fragments of first-order logic. A DL-based KB represents the domain with descriptions of concepts, relations, and their instances. DLs underpin the Web Ontology Language (OWL), whose latest version of OWL2 [11] is based on DL $\mathcal{SROIQ}$ [7].

An $\mathcal{SROIQ}$ ontology Σ usually consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. $\mathcal{T}$ contains a set of concept inclusion axioms such as $C \sqsubseteq D$, relation inclusion axioms such as $R \sqsubseteq S$, $R_1 \circ \ldots \circ R_n \sqsubseteq S$, and other axioms saying that a particular relation is functional, or reflexive, or symmetric, etc.; $\mathcal{A}$ contains axioms about individuals, e.g. $a : C$ (a is an instance of C), $(a, b) : R$ (a has an R relation with b).

Given a set of atomic concepts, the entire set of concepts expressible by $\mathcal{SROIQ}$ is defined recursively. Assuming that C and D are concepts, then so are $\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \ \leq nR.C \mid \ \geq nR.C \mid \exists R.Self \mid \{a_1, \ldots, a_n\}$, where $\top$ is the top concept, $\bot$ the bottom concept, A an atomic concept, n a non-negative

integer number, $\exists.Self$ the self-restriction, a_i individual names and R a relation which can either be an atomic relation r or the inverse of another relation (R^-).

An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^\mathcal{I}, .^\mathcal{I} \rangle$ where $\Delta^\mathcal{I}$ is a non-empty set and $.^\mathcal{I}$ is a function that maps atomic concept A to $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, atomic role r to $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ and individual a to $a^I \in \Delta^\mathcal{I}$. The interpretation of complex concepts and axioms can be defined inductively based on their semantics, e.g. $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, etc.

$\mathcal{I}$ is a *model* of Σ, written $\mathcal{I} \models \Sigma$, iff the semantics of all the axioms in Σ are satisfied. It should be noted that one Σ can have multiple models. For example when $\mathcal{T} = \emptyset, \mathcal{A} = \{a : A \sqcup B\}$, there can be a model $\mathcal{I}_1$ s.t. $a^{\mathcal{I}_1} \in A^{\mathcal{I}_1}, a^{\mathcal{I}_1} \notin B^{\mathcal{I}_1}$, and another model $\mathcal{I}_2$ s.t. $a^{\mathcal{I}_2} \notin A^{\mathcal{I}_2}, a^{\mathcal{I}_2} \in B^{\mathcal{I}_2}$. In other word, the world is open. For details, we refer the readers to [2].

When dealing with a closed world, people usually (partially) close the ontology with a DBox $\mathcal{D}$ [12], which is syntactically similar to the $\mathcal{A}$. However, $\mathcal{D}$ contains only $a : A$ ($(a, b) : r$) where A (r) is atomic. Furthermore, every concept or relation appearing in $\mathcal{D}$ is closed. Its extension is exactly defined by the contents of $\mathcal{D}$, i.e. if A (r) appearing in $\mathcal{D}$ and $a : A \notin \mathcal{D}$ ($(a, b) : r \notin \mathcal{D}$), then $a : \neg A$ ($(a, b) : \neg r$), thus is the same in all the models. The concepts and relations not appearing in $\mathcal{D}$ are still open.

DL reasoning can be exploited to infer implicit information from ontologies. For example, given $\mathcal{T} = \{Dog \sqsubseteq \exists feed^-.Woman\}$ and $\mathcal{A} = \{d1 : Dog, w1 : Woman\}$, we know that there must be some $Woman$ who feeds $d1$. When the domain is closed as $\mathcal{D} = \mathcal{A}$ we can further infer that this $Woman$ is $w1$ although there is no explicit relation between $w1$ and $d1$. The complexity of such reasoning services is normally 2NEXPTIME-complete.

2.2 Background Assumptions

When applying DL to GRE, people usually impose the following assumptions.

- Identifiers cannot be used in REs. For example, "the Woman who feeds $d1$" would be invalid, because $d1$ is an identifier. Such identifiers are typically outlawed in GRE because, in many applications, many objects do not have identifiers that readers/hearers would be familiar with: e.g. chairs, trees, or time periods seldom have commonly known identifiers.
- *Closed Domain Assumption (CWA)*: In existing works regarding DL and GRE, people assume that $\mathcal{D} = \mathcal{A}$. Furthermore, the domain is usually considered to be finite and containing individuals only visible in $\mathcal{D}$. As we will show, this "partially" close the interpretation of the atomic symbols mentioned in $\mathcal{A}$ but will still allow the rest open.
- *Unique Name Assumption (UNA)*: Different names denote different individuals. If, for example, $w1$ and $w2$ may potentially be the same woman, then we can not distinguish one from the other.

We follow these assumptions when discussing existing works and presenting our approach. We will also show how our approach can be extended to achieve more when these assumptions are not imposed. In addition, we consider the entire KB, including both $\mathcal{A}$ ($\mathcal{D}$) and $\mathcal{T}$.

It is worth mentioning that, in the syntax of $\mathcal{SROIQ}$, negation of relations are not allowed in concept expressions, e.g. you can not compose a concept $\exists\neg feed.Dog$. However, if we impose the CWA and close $feed$, then we can interpret $(\neg feed)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus feed^{\mathcal{I}}$. In the rest of the paper, we use this as a syntactic sugar.

2.3 Using DL for GRE

Every DL concept can be interpreted as a set. If the KB allows one to prove that this set is a singleton then the concept is a referring expression. It is this simple idea (earlier expounded by [6]) that [1] explored. In doing so, they say little about the TBox, appearing to consider only the ABox (DBox), which contains only axioms about instances of atomic concepts and relations. For example, the domain in Fig.1 can be described as

KB1: $\mathcal{T}_1 = \emptyset$, $\mathcal{A}_1 = \{w1 : Woman, w2 : Woman, d1 : Dog, d2 : Dog, c1 : Cat, c2 : Cat, (w1, d1) : feed, (w2, d1) : feed, (w2, d2) : feed, (d1, c1) : love\}$

Assuming that this represents a Closed World, the authors propose an algorithm that is able to generate descriptions by partitioning the domain.[1] More precisely, the algorithm first finds out which objects are describable through increasingly large conjunction of (possibly negative) atomic concepts, then tries to extend these conjunctions with complex concepts of the form $(\neg)\exists R.Concept$, then with concepts of the form $(\neg)\exists R1.(Concept \sqcap (\neg)\exists R2.Concept)$, and so on. At each stage, only those concepts that have been acceptable through earlier stages are used. Consider, for instance, KB1 above. Regardless of what the intended referent is, the algorithm starts partitioning the domain with atomic concept (suppose staring with Dog) in (1), then the ones in (2), then the ones in (3). Each stage makes use of all previous stages. During stage (3), for example, the object $w2$ could only be identified because $d2$ was identified in phase (2).

(1). $Dog = \{d1, d2\}$, $\neg Dog \sqcap Woman = \{w1, w2\}$,
$\neg Dog \sqcap \neg Woman = \{c1, c2\}$.
(2). $Dog \sqcap \exists love.(\neg Dog \sqcap \neg Woman) = \{d1\}$,
$Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman) = \{d2\}$.
(3). $(\neg Dog \sqcap Woman) \sqcap \exists feed.(Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman)) = \{w2\}$,
$(\neg Dog \sqcap Woman) \sqcap \neg\exists feed.(Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman)) = \{w1\}$.

As before, we disregard the important question of the quality of the descriptions generated, other than whether they do or do not identify a given referent uniquely. Other aspects of quality depend in part on details, such as the question in which order atomic properties are combined during phase (1), and analogously during later phases.

[1] Areces et al. [1] consider several DL fragments (e.g., $\mathcal{ALC}$ and $\mathcal{EL}$). Which referring expressions are expressible, in their framework, depends on which DL fragment is chosen. Their analysis of the differences between fragments is perhaps the most valuable aspect of their paper. Existential quantification, however, is the only quantifier that was used, and inverse relations are not considered either.

Although this demonstrates how DL can be used in GRE, it does not extend the expressive power of GRE. This is not because of some specific lapse on the part of the authors: it seems to have escaped the GRE community as a whole that relations can enter REs in a variety of alternative ways. Also, the above algorithm considers only the ABox therefore some background information will not be used. For example, suppose we extend Fig.1 with background knowledge saying that one should care about thoes beloved by whom he/she is feeding, and an additional love relation (Fig.2).

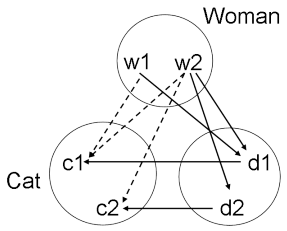

Fig. 2. An extended example of Fig.1. Edges from women to cats denote $care$ relations. Dashed-edge indicates implicit relations.

Together, the domain can be described as:

KB2: $\mathcal{T}_2 = \{feed \circ love \sqsubseteq care\}$, $\mathcal{A}_2 = \mathcal{A}_1 \cup \{(d2, c2) : love\}$

The TBox axiom should allow the inference of additional facts: the facts $(w1, c2) : care$, $(w2, c1) : care$, and $(w2, c2) : care$ can be inferred using ontology reasoning. Our own approach will allow this kind of reasoning to be brought to GRE. Continuing to focus on the materialised KB2, we note another limitation of existing works: if only existential quantifiers are used then some objects are unidentifiable (i.e., it is not possible to distinguish them uniquely). These objects would become identifiable if other quantifiers and inverse relations were allowed. For example,

The cat who is cared by *at least 2* women $= Cat \sqcap \geq 2feed^{-}.Woman = \{c1\}$,
The woman feeding *only* those fed by *at least 2* women $= Woman \sqcap \forall feed. \geq 2.feed^{-}.Woman = \{w1\}$,
The woman who feeds *all* the dogs $= \{w2\}$.

It thus raises the question: which quantifiers will be appreciated and how to use DL to realise them in GRE?

3 Beyond Existential Descriptions

In this section, we show how more expressive DLs can make objects referable that were previously unreferable. Far from being a minor modification of existing works, this will amount to a substantial reformulation which will allow the DL-based approach to move beyond other GRE algorithms in its expressive power and representational efficiency.

3.1 Expressing Generalized Quantifiers in OWL2

Because the proposal in [1] uses only existential quantification, it fails to identify any individual in Fig.2. Before attempting to fill this gap, we briefly pause to ask what level of expressivity might be achievable. In doing so, we shall make use of a conceptual apparatus developed in the formal study of natural language. The most general format for REs that involve a relation `R` is, informally, `the N1 who R Q N2's`, where `N1` and `N2` denote sets and `Q` is a quantifier. (Thus for example `the women who feed SOME dogs`.) An expression of this form is a uniquely identifying expression if it corresponds to exactly one element in the domain. Using a slightly more formal set-theoretic notation, this means that the following set has a cardinality of 1:

$$\{y \in N1 : Qx \in N2 \mid Ryx\}$$

where Q is a *generalized quantifier* (GQ [10]). For example, if Q is the existential quantifier, while $N1$ denotes the set of women, N2 the set of dogs, and R the relation of feeding, then this says that the number of women who feed SOME dog is just one. If Q is the quantifier *Exactly three*, however, then it says that the number of women who feed exactly THREE dogs is just one. It will be convenient to write the formula above in the standard GQ format where quantifiers are seen as relations between sets of domain objects A, B. For example, using the universal quantifier as an example, instead of writing $\forall x \in A \mid x \in B$, we write $\forall(AB)$. Thus, the formula above is written

$$\{y \in N1 : Q(N2\{z : Ryz\})\}.$$

Instantiating this as before, we get $\{y \in Woman : \exists(Dog\{z : Feed\ yz\})\}$, or "women who feed a dog", where Q is $\exists$, $A = Dog$ and $B = \{z : Feed\ yz\}$ for some y.

Mathematically characterizing the class of *all* quantifiers that can be expressed in referring expressions is a complex research programme to which we do not intend to contribute directly, partly because this class includes quantifiers that are computationally problematic; for example, quantifiers such as *most* (i.e., more than 50%) and *many* (which is vague) are not first-order expressible, as is well known.

To make transparent which quantifiers are expressible in the logic that we are using, let us think of quantifiers in terms of simple quantitative constraints on the sizes of the sets $A \cap B$, $A - B$, and $B - A$, as is often done in GQ theory, asking what types of constraints can be expressed in referring expressions based on $\mathcal{SROIQ}$. The findings are illustrated in Tab.1. The table shows that OWL2 can express any of the following, plus disjunctions and conjunctions of anything it can express.

Let us call the class of quantifiers defined by the table NatGQ. To see how large and general NatGQ is, a few examples will be useful. When $n = 1$, for example, type 1 becomes $\exists R.N2$, i.e. the *existential* quantifier. When $n = 0$ type 7 becomes $\forall R.N2$, i.e. the *only* quantifier. When $n = 0$ type 6 becomes $\forall \neg R.\neg N2$, i.e. the *all* quantifier. In types 2, 4, 6 and 8, negation of relation is used in a concept expression. This is not directly supported in $\mathcal{SROIQ}$ but, as we indicated earlier, given a closed KB Σ, when relation R is closed, $\neg R$ is valid in concepts.

Together, this allows the expression of a description such as "women who feed at least one but at most 7 dogs", by conjoining a quantifier of type 1 (with $n = 1$) with one of type 5 (with $n = 7$). It even allows expression of "women who do not feed

Table 1. Expressing GQ in DL

	QAB	DL
1	$\geq nN2\{z : Ryz\}$	$y :\geq nR.N2$
2	$\geq nN2\neg\{z : Ryz\}$	$y :\geq n\neg R.N2$
3	$\geq n\neg N2\{z : Ryz\}$	$y :\geq nR.\neg N2$
4	$\geq n\neg N2\neg\{z : Ryz\}$	$y :\geq n\neg R.\neg N2$
5	$\leq nN2\{z : Ryz\}$	$y :\leq nR.N2$
6	$\leq nN2\neg\{z : Ryz\}$	$y :\leq n\neg R.N2$
7	$\leq n\neg N2\{z : Ryz\}$	$y :\leq nR.\neg N2$
8	$\leq n\neg N2\neg\{z : Ryz\}$	$y :\leq n\neg R.\neg N2$

all dogs and who feed at least one non-dog", which can be expressed as $Woman \sqcap \neg\forall\neg Feed.\neg Dog \sqcap \exists Feed.\neg Dog$. In addition to Tab.1, $\mathcal{SROIQ}$ can even represent reflexive relation such as "the dog who loves itself" by $Dog \sqcap \exists love.Self$, which used to be regarded infeasible [6].

Comparing the quantifiers that become expressible through OWL2's apparatus with classes of quantifiers studied in the theory of GQ, it is clear that OWL2 is highly expressive: it does not only include quantifiers expressible in Van Benthem's binary tree of numbers [13] – which is often regarded as sufficient – but much else besides. Wider classes of referring expressions can certainly be conceived – for example by moving into intensionally or higher-order logic – but these are not likely to have overwhelming practical utility in todays's NLG applications.

4 Generating $\mathcal{SROIQ}$-enabled REs

In this section, we present an algorithms that is able to compute the descriptions we presented in sect.3. A GRE algorithm should have the following behaviour: if an entity is distinguishable from all the others, the algorithm should find a unique description; otherwise, the algorithm should say there exists no unique description. There are two major tasks in a GRE program:

1. Finding possible descriptions: Generating syntactically valid descriptions.
2. Validating a description: Checking whether a description can be satisfied by a particular object.

These two tasks can be done simultaneously for all the domain elements, or for a target referent. In the former case, candidate descriptions are generated and then tested for each domain element. If a description holds for only one element, then it is the RE for that element. This generate & test cycle is repeated until all the REs can be found. In the later case, candidate descriptions are generated and tested for a particular target referent until a valid RE is found or such a target referent can not be distinguished from some other element. In this paper, we follow the strategy of Areces et.al work to generate REs in a simultaneous way.

Since we consider more constructs than any previous treatment of relational descriptions, the combination of them can result in an enormously large search space. To measure the complexity of these descriptions, we define their depth:

Definition 1. ***(Depth)*** *Given a description d, its depth $|d|$ is calculated as follows:*

1. $|d| = 1$ *for* $d := \top|\bot|A|\neg A$.
2. $|d \sqcap d'| = |d \sqcup d'| = max(|d|, |d'|) + 1$.
3. $|\exists r.d| = |\forall r.d| = | \leq nr.d| = | \geq nr.d| = | = nr.d| = |d| + 1$.

Syntactically different descriptions can have same semantics, e.g. $\neg\forall R.A \equiv \exists R.\neg A$. We leave aside the question which syntactic variant should be used and focus on generating one form, assuming all the concepts are in their unique *negation normal form* (NNF). A NNF has $\neg$ in front of only atomic concepts (include $\top$ and $\bot$) or nominals. The NNF of $\neg C$ is denoted by $\sim C$.

To ensure we can generate proper descriptions w.r.t. particular requirements, we present the following abstract algorithm A-1. Given an ontology Σ, we initialise the algorithm with the following sets:

1. The concept name set CN is the minimal set satisfying:
   - $\top \in CN$;
   - if A is an atomic concept in Σ, then $A \in CN$;
   - if R is an atomic role in Σ, then $\exists r.Self \in CN$;
   - if $A \in CN$, then $\sim A \in CN$;
2. The relation name set RN is the minimal set satisfying:
   - if R is an atomic relation in Σ, then $R \in RN$;
   - if $R \in RN$, then $\sim R \in RN$;
   - if $R \in RN$, then $R^- \in RN$;
3. The number set $N = \{1, 2, \ldots, n\}$ where n is the number of individuals in Σ.
4. The construct set S contains all the constructs that supported by a particular language. For $\mathcal{SROIQ}$, $S = \{\neg, \sqcap, \sqcup, \exists, \forall, \leq, \geq, =\}$. Usage of names is disallowed (cf sect.2).

Obviously, $\sim(\sim A) = A$, $\sim(\sim R) = R$, $(R^-)^- = R$, and $(\sim R)^- = \sim R^-$. Then the algorithm takes an ontology Σ as its input and output a queue D of descriptions.

Algorithm A-1: Construct-description(Σ, CN, RN, N, S)
INPUT: Σ, CN, RN, N, S
OUTPUT: Description Queue D

```
1:  D := CN
2:  for d = fetch(D) do
3:    for each s ∈ S do
4:      if s = ⊓ or s = ⊔ then
5:        for each d' ∈ D do
6:          D := Add(D, d ⊓ d'(d ⊔ d'))
7:      if s = ∃ or s = ∀ then
8:        for each r ∈ RN do
9:          D := Add(D, ∃r.d(∀r.d))
10:     if s =≤ or s =≥ or s is = then
11:       for each r ∈ RN, each k ∈ N do
12:         D := Add(D, ≤ kr.d(≥ kr.d, = kr.d))
```

13: **return** D

Algorithm A-2:Add(D, e)
INPUT: D, e
OUTPUT: (Extended)Description Queue D

1: **for** $d \in D$ **do**
2: **if** $|d| < |e|$ and $d \sqsubseteq_\Sigma e$ **then**
3: **return** D
4: **else if** $|d| = |e|$ and $d \sqsubset_\Sigma e$ **then**
5: **return** D
6: **if** $|[[e]]|^\Sigma > 0$ **then**
7: $D := D \cup \{e\}$
8: **return** D

In Step 1, D is initialised by CN. From Step 2, we recursively process elements of D one by one. We use $fetch(D)$ to retrieve the first unprocessed element of D and new elements are added to the end of D. Thus D is a first-come-first-server queue (note that processed elements are not removed from D). For each element d of D, Step 3 to 12 extend it with a construct s:

1. If s is $\sqcap$ or $\sqcup$, in Step 5 and 6, we extend d with all the elements of D and add new descriptions to D.
2. If s is $\exists$ or $\forall$, in Step 8 and 9, we extend d with all relations of RN and add new descriptions to D. In Areces et el.'s work, $\forall$ is also available when using $\neg$ and $\exists$ together, however due to their algorithm they can never generates descriptions like $\forall r.A$.
3. If s is $\leq, \geq$ or $=$, in Step 11 and 12, we extend d with all relations of RN and all numbers of N, and add new descriptions to D.
   In this step, $= kr.d \equiv \geq kr.d \sqcap \leq kr.d$, which means $=$ construct can be equivalently substituted by the combination of $\leq, \geq$ and $\sqcap$ constructs.
   Therefore, it is an modelling choice to use either $\leq, \geq$, or only $=$, or all of them. In this algorithm we present all of them from a syntactic point of view.

Because we compute only the NNF and we disallow the usage of individual identifiers, negation $\neg$ appears only in front of atomic concept names, which have all been included in CN. Thus in extension, we do not consider $s = \neg$. The ordering of choosing constructs, relations, integers and conjuncts/disjuncts is not the topic of this paper.

Obviously, at any time, D, RN, N, S are all finite, thus Step 3 to 12 terminates for a particular $d \in D$. Because Step 3 to 12 generates descriptions with incremental depth, for a particular n, there are finite $d \in D$ such that $|d| = n$. Thus, the termination of Algorithm A-1 depends on the increment of D. This is controlled by the Add procedure, which determines whether a new generated description is added into D or not.

The mechanism of Add depends on the requirements of the application. In this paper we control the addition by following a simple heuristic: more complex descriptions should have smaller extension.

In Algorithm A-2, Step 2 ensures that, when adding a new description e into D, its extension should be smaller than any existing description $d \in D$ with a smaller depth than e. Step 4 ensures that when adding a new description e into D, its extension should be no larger than any existing description $d \in D$ with same depth as e. Step 6 to 7 adds a new description when its extension is non-empty. The subsumption checking in Step 2 and 4, the instance retrieval in Step 6, must be realised by DL reasoning.

A-2 guarantees that when the complexity of descriptions increases, their extensions are getting smaller and smaller (but still non-empty). Because descriptions of a particular depth is always finite, when the domain is finite, Algorithm A-1 always terminates.

It can be shown that, our approach is an extension of the algorithm presented in Areces et al.'s work. The example in Fig.2 shows that some referring expressions generated by our algorithm cannot be generated by our predecessors; more importantly even, some objects that are not referable for them are referable for us.

It is worth stressing here that our algorithm focusses on finding uniquely referring expressions, leaving aside which of all the possible ways in which an object can be referred to is "best". For this reason, *empirical* validation of our algorithm – a very sizable enterprise in itself, which should probably be based on descriptions elicited by human speakers – is not yet in order.

Discussion Now we revisit the basic assumptions to see what can be achieved without them.

1. Using names in REs, e.g. "the husband of Marie Curie". Here "Marie Curie" servies as both the identifier of the individual and the name of its interpretation. In this case, we extend our Algorithm A-1 by including $\{Maria\_Curie\}$ in CN.
2. An open world: when the domain is not restricted to be closed, traditional GRE approaches may fail because they have always been assuming a single model with complete knowledge. In this case, interesting REs can still be found by our approach. For example, if someone is known to be the only Chinese or Japanese, we can refer to him/her as $Chinese \sqcup Japanese$ although the exact nationality is unknown.
3. Individual with multiple names. DL imposes the UNA by explicit asserting the inequality of each two individuals. Without UNA, reasoning can still infer some results, e.g. $\{Woman \sqcap Man \sqsubseteq \bot, David : Man, May : Woman\} \models David \neq May$. Thus we can refer to David as "the man" if the domain is closed.

5 Conclusion: widening the remit of GRE

This paper has shown some of the benefits that arise when the power of KR is brought to bear on an important problem in NLG, namely the generation of referring expressions (GRE). We have done this by using DL as a representation and reasoning formalism, extending previous work in GRE in two ways. In order to explain what class of referring expressions is covered by our proposal, we have related our algorithm to the theory of Generalized Quantifiers, which allowed us to formally characterize the set of quantifiers that are used by our algorithm, thereby making exact how much expressive power we have gained. Secondly, we have demonstrated the benefits of *implicit* knowledge

through inferences that exploit TBox-information, thereby allowing facts to be represented more efficiently and elegantly, and allowing GRE to tap into kinds of generic (as opposed to atomic) knowledge that it had so far left aside, except for hints in [6] and in [3].

Current work on reference is overwhelmingly characterized by an emphasis on empirical accuracy, often focussing on very simple referring expressions, which are constituted by conjunctions of 1-place relations (as in "the grey poodle", "the Swedish woman"), and asking which of these conjunctions are most likely to be used by human speakers (or sometimes, which of these would be most useful to a human hearer or reader). The present work stresses entire different concerns: we have focussed on questions of expressive power, focussing on relatively complex descriptions, asking what referring expressions are possible when relations (such as "love" or "feed") between domain objects are used. We believe that, at the present stage of work in GRE, it is of crucial importance to gain insight into questions of this kind, since this will tell us what types of reference are possible in principle. Once these questions are answered, we shall explore how the newly gained expressive power can be put to practical use.

References

1. Carlos Areces, Alexander Koller, and Kristina Striegnitz. Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference*, Salt Fork, Ohio, 2008.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. Madalina Croitoru and Kees van Deemter. A conceptual graph approach to the generation of referring expressions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.
4. Robert Dale. Cooking up referring expressions. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 68–75, 1989.
5. Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *CoRR*, cmp-lg/9504020, 1995.
6. Claire Gardent and Kristina Striegnitz. Generating bridging definite descriptions. *Computing Meaning*, 3:369–396, 2007.
7. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In *KR 2006*, 2006.
8. Emiel Krahmer and Mari?t Theune. Efficient context-sensitive generation of descriptions in context. *Information Sharing: Givenness and Newness in Language*, pages 223–264, 2002.
9. Emiel Krahmer, Sebastiaan van Erk, and Andr Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003.
10. A Mostowski. On a generalization of quantifiers. *Fund. Math.*, 44:235–273, 1957.
11. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. Owl 2 web ontology language: Profiles. W3c working draft, W3C, October 2008.
12. Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over dboxes. In *IJCAI 2009*, 2009.
13. Johan van Benthem. Essays in Logical Semantics. Reidel, 1986.
14. Kees van Deemter and Emiel Krahmer. Graphs and booleans: On the generation of referring expressions. *Computing Meaning*, 3:397–422, 2007.