Referability
Kees van Deemter
University of Aberdeen

1 Introduction

A key task of almost any Natural Language Generation (NLG) system is to refer to an entity. Linguists and philosophers have a long tradition of theorising about reference. In the words of the philosopher John Searle, “Any expression which serves to identify any thing, process, event, action, or any other kind of individual or particular I shall call a referring expression. Referring expressions point to particular things; they answer the questions Who?, What?, Which?” [Searle, 1969]. In this chapter, we take Generation of Referring Expressions to be the task of producing a description of the referent that allows the reader to identify it. In doing so, the generator can make use of any information that it can safely assume the user to possess, based on a model of the hearer’s information (henceforth, the Knowledge Base, or KB for short). Given an algorithm for producing referring expressions (REs), the following questions can be asked:

1. How many entities is the algorithm able to identify? In this connection, we shall speak of the expressive power of an algorithm. Loosely speaking, the more entities the algorithm is able to single out, the greater its expressive power.

2. How empirically adequate are the REs generated by the algorithm? For example, to what extent do they resemble the human-produced REs in a corpus, and how effective are the REs generated by the algorithm, in terms of allowing a human recipient to identify the referent easily, quickly and reliably?

Most work on the generation of referring expressions (variously abbreviated as REG or GRE) is devoted to questions of the second type, with studies such as [Passonneau, 1996], [Jordan and Walker, 2000], [van Deemter et al., 2012], [Gatt and Belz, 2010], [Viethen and Dale, 2006], [Viethen and Dale, 2008], [Guhe and Bard, 2008], [Goudbeek and Krahmer, 2012] focussing on the “human-likeness” of the descriptions generated and a far smaller number of studies starting to focus on the effectivity of the descriptions ([Paraboni et al., 2007], [Campana et al., 2011], [Khan et al., 2012]). We shall argue that, although this empirical work is crucial, the first, logically oriented question deserves the attention of the research community as well, because it can open our eyes to limitations of the algorithms involved, and enable us to diminish these limitations by extending the expressive power of REG algorithms. In discussing question 1, we shall see how limited the expressive power of most REG algorithms still is. In studying these issues, we will hit upon another, trickier question:

3. Given a KB, which entities can be referred to (i.e. identified)?

We shall ask how current REG algorithms shape up against the yardstick suggested by this question, and explore novel ways of extending their expressive power. Since this is a large area to be writing about, we focus on the “logical” task of individuating a target referent, bypassing the question which of all the different expressions that individuate a given target is best (e.g. in the sense of question (2) above). Each extension in expressive power gives rise to new empirical questions because, for every novel type of referring expression, one can ask how and when human speakers
use it, and how effective it is for a human recipient. These empirical considerations, however, will not be the theme of this chapter, except for a brief discussion in section 6.

Moreover, we shall focus on ‘first mention’ descriptions only, (unlike [Krahmer and Theune, 2002], [Siddharthan and Copestake, 2004], [Belz et al., 2010]). This is a weighty simplification, particularly where interactive NLG systems are concerned (i.e., where the user of the system can speak or write). Yet, all the issues discussed here are as relevant for interactive settings as for non-interactive ones, because the same types of domain knowledge are relevant in both situations, and because any referring expression that can be used in a non-interactive setting can also be used interactively. Everything we shall say about proper names, sets, and quantifiers, for example, will be relevant for REs generated in interactive contexts. Our focus on ‘first mention’ descriptions will mean, however, that anaphoric phenomena will be out of reach. We shall turn to issues of interaction and dialogue briefly at the end of the chapter, in section 6.4.

We follow [Dale and Reiter, 1995] in focussing on the content of a description (i.e., the problem of content determination), assuming that any combination of properties can be expressed by the NLG module responsible for linguistic realisation. Accordingly, when we speak of a referring expression (RE), we will often mean a formal expression, rather than an actual linguistic form. Accordingly, the word ‘description’ will refer to the semantic content of a linguistic expression only, for example as a set of properties, whose expression in language will be left aside. We shall not assume that a description is always expressed by a single noun phrase: if several sentences are needed then so be it.

This chapter will start by summarising existing work that has sought to extend the expressive power of REG, then present novel work regarding Proper Names and, particularly, regarding question 3 above. In section 2, we start with a REG algorithm that, in terms of its expressive power, is representative of the algorithms submitted to the REG challenges ([Gatt and Belz, 2010]). Strikingly, these algorithms can only generate REs that express conjunctions of atomic one-place predicates, but we shall summarise how these algorithms were later modified to express other Boolean connectives. In section 3, we summarise earlier work on relational descriptions and argue that, to do full justice to relational descriptions, and implicit knowledge, it is necessary to link REG with serious Knowledge Representation. This will lead on to a discussion of the curious situation that has now arisen in Computational Linguistics, where most researchers in Natural Language Generation – who routinely generate text from a Knowledge Base – choose to ignore modern techniques for representing and manipulating knowledge. In section 4, we make a small detour, addressing a different kind of limitation of REG algorithms by showing, for the first time, how proper names can be incorporated into REG algorithms. Section 5 will take a purely theoretical stance, asking what would need to be done if we wanted to make REG algorithms “logically complete”, thereby addressing question 3 (above). The concluding section 6 will discuss how much is gained by extending REG algorithms as proposed in earlier sections, arguing that there are substantial benefits to be had here, both practically (in terms of generating intelligible REs) and theoretically (in terms of obtaining insight into the human language competence).

Telling the story of the expressive power of REG will allow us to take stock of a line of research that has occupied me repeatedly over the last 12 years or so, to move this work one step further forward, and to ask what lies ahead. One lesson that emerges is “Be careful what you think you can do without”. For instance, you may only be interested in reference to individual objects, and conclude from this that you can do without reference to sets. This conclusion is not warranted, since there are situations where the easiest way to refer to an object is via a set, as when we say “The father of the children”, where your aim is to refer to the father, but you accomplish this via a reference to his children. Or, you may not be interested in Proper Names, and conclude that you
can do without them. Again, the conclusion does not follow, since the best way to refer to a person whose name you don’t know may be via someone or something whose name you do know, as when we refer to a person as “The author of Waverley” (which makes use of a proper name), an example from Bertrand Russell that was set in a time when it was not generally known who had written the novel entitled “Waverley”.

2 An algorithm for generating Boolean REs

It is commonplace to start an article on REG by paraphrasing the Incremental Algorithm of [Dale and Reiter, 1995], a well known algorithm aimed at generating human-like REs efficiently. (For a discussion of these and other classic REG algorithms, see the survey [Krahmer and Van Deemter, 2012].) In this chapter, however, we bypass the “humanlikeness” criterion, because our subject is the expressive power of algorithms. For this reason, we shall use a different starting point, first described in [van Deemter and Halldórsson, 2001], which is based on principles that will be useful to us in the more complex situations that we shall study later on.

Suppose the REG task is the following: There is a domain $D$ with a target referent $r$ in it and a number of other entities, which we call distractors. Suppose, furthermore, there is a set $\mathbb{P}$ of atomic properties (i.e., properties that are represented as semantic primitives, without any logical structure), that can be used to identify $r$. The basic idea of the algorithm is to simply list all properties that are true of the referent $r$, and intersect their extensions:

$$\bigcap\{P \in \mathbb{P} : P(r)\}$$

The most direct way to turn this simple idea into an algorithm is by calculating, for each of the properties $P$ in $\mathbb{P}$, its extension $[[P]]$, to assemble the ones for which $r \in [[P]]$ in a list that forms a tentative description and to intersect the extensions of this tentative description (i.e., intersect the $[[P]]$). If the resulting intersection equals the singleton $\{r\}$ then the tentative description is an RE that identifies $r$. To see how this works, consider a KB whose domain $D$ is a set of entities $(a,b,c,d,e)$, over which four properties have been defined:

- dog: $\{a, b, c, d, e\}$ (all entities in the domain are dogs)
- poodle: $\{a, b\}$
- black: $\{a, c\}$
- white: $\{b, e\}$

The assumption – to be made precise later – is that the listing of extensions is complete. For instance, $a$ and $b$ are the only poodles (whatever else they may be), and $d$ is neither black nor white (whatever else it may be). Here’s what happens when $a$ and $c$, respectively, play the role of the target referent. The algorithm finds an RE for $a$, but not for $c$:

$$\text{Description}(a) = \text{dog} \cap \text{poodle} \cap \text{black} = \{a\}$$
$$\text{Description}(c) = \text{dog} \cap \text{black} = \{a, c\}$$

Is this algorithm logically complete? The answer depends on whether other operators than set intersection (i.e., other than logical conjunction) are taken into account. Classic REG algorithms have tended to disregard set complementation (i.e., negation) [van Deemter, 2002]. To see that this matters, consider that if it is permitted to describe an animal as “not a poodle” (negating the property of being a poodle), and similarly for all atomic properties, then all five objects become distinguishable. Dog $c$, for example, is identified by the combination $\text{black, Poodle, White}$. It’s
the black dog that’s not a poodle and not white. (Once again there is a superfluous property here, and once again we do not care.)

Is negation important? If all non-poodles were known to be spaniels then the property of not being a poodle would become superfluous of course. More generally, if for each property \( P \) in the KB, the KB also contains a property coextensive with \( \overline{P} \) (the complement of \( P \)) then negation does not add to the expressive power of the generator. Descriptions of the form \( \overline{X} \cup \overline{Y} \) do not add expressive power: \( \overline{X} \cup \overline{Y} \) can be written equivalently as \( \overline{X} \cap \overline{Y} \): the negations can be “pushed down” until it is only applied to atomic properties. In other words, if reference to just one individual is our goal, one might argue that negation is not an important problem. But what if we want to refer to a set of entities? Until recently, REG has aimed to produce references to a single object, but references to sets are ubiquitous in most text genres. We shall see that, in simple cases, some fairly small modifications allow classic REG algorithms to refer to sets [van Deemter, 2002].

Before we proceed, two issues are worth getting out of the way. First, one can describe sets both in terms of set theory and in terms of logical symbols. We can talk about a set of objects as either the set-theoretic union \( A \cup B \), or in logical jargon as the set of those objects that are members of \( A \) or \( B \) (or both), that is \( \{x \mid x \in A \lor x \in B\} \). It will be convenient for us to switch between both perspectives, sometimes talking about (logical) negation and disjunction, sometimes about (set-theoretic) complementation and union. Secondly, when taking the logical perspective, disjunction (\( \lor \)) may be realised as conjunction (“and”). Thus, the set just described is perhaps most naturally referred to as “the As and the Bs” (e.g., “the black dogs and the white dogs”).

What happens if we want to be able to refer to sets of entities? First, it will be useful to look a bit differently at what we did when referring to individual entities such as \( a \): we accumulated all their properties into a tentative description. Let’s call the tentative description constructed for the target referent \( r \) (i.e., the intersection of the extensions of all the properties true of \( d \)) the satellite set of \( d \). Now let’s carry out this construction for each element of the domain:

\[
\begin{align*}
Satellites(a) &= \text{dog} \cap \text{poodle} \cap \text{black} = \{a\} \\
Satellites(b) &= \text{dog} \cap \text{poodle} \cap \text{white} = \{b\} \\
Satellites(c) &= \text{dog} \cap \text{black} = \{a, c\} \\
Satellites(d) &= \text{dog} = \{a, b, c, d, e\} \\
Satellites(e) &= \text{dog} \cap \text{white} = \{b, e\}
\end{align*}
\]

Satellite sets are useful constructs, not just because they can be used in algorithms, but because they show us which entities can be referred to at all. The reason is that a satellite set results from conjoining all the properties of a given entity. In doing so, we give ourselves the best possible chance of success: if a distractor is not ruled out by this conjunction then it cannot be ruled out at all. Thus, the satellite set of an object \( o \) is the set of objects from which \( o \) cannot be distinguished. If this set contains only \( o \) then \( o \) can be identified by conjoining the extensions of all \( o \)’s properties, but if it contains one or more other objects, then it cannot. In the example at hand, it is easy to see that only \( a \) and \( b \) can be distinguished from everything else, namely by conjoining the properties \text{dog, poodle, and white}. (The fact that the property \text{dog} is superfluous is immaterial for present purposes, because we don’t aim for “natural” descriptions yet.) In other words, if your aim is to
identify $a$ then the satellite set construct gives you a way of doing that. If your aim is to identify $c$, and the four properties listed above are the only ones available to you then satellite sets show you that you should give up, because no algorithm will be able to do it for you. If your aim is to identify the set $\{a, c\}$ then satellite sets show you that that’s possible: the conjunction of $\text{black}$ and $\text{dog}$ will do it. (Likewise, the property $\text{black}$ on its own will do it.)

For referring to sets, conjunction and negation of atoms do not always suffice. This can be seen from both examples above again, where it is impossible to refer to the set $\{a, b, d, e\}$ with just these two connectives: we need to add full negation, in which case we can say $\text{Black} \cap \overline{\text{Poodle}}$ (which describes the target set as the complement of the set containing just the earlier-described individual $c$), or disjunction, using the description $\text{Poodle} \cup \overline{\text{Black}}$ (“the poodles and the ones that are not black”). One approach, proposed in [van Deemter and Halldórsson, 2001], is to conjoin all the literals (i.e., atomic properties and their negations) that hold true of each of the $n$ element of the set, and to disjoin these $n$ conjunctions (i.e., to disjoin the description of each of the $n$ elements):

$\bigcup_{d \in S} \bigcap_{A \in S_d} [[A]],$

where $S_d = \{A : A \in \mathbb{P}_{\text{neg}} : d \in [[A]]\},$

where $\mathbb{P}_{\text{neg}} := \mathbb{P} \cup \{\overline{P_i} : P_i \in \mathbb{P}\}.$

In more detail, the idea is to start adding to $\mathbb{P}$ the properties whose extensions are the complements of those in $\mathbb{P}$. The resulting set, which contains negative as well as positive properties, is called $\mathbb{P}_{\text{neg}}$. For each element $d$ of $S$, the algorithm finds those properties in $\mathbb{P}_{\text{neg}}$ that are true of $d$, and the intersection of (the extensions of) these properties is formed; this intersection is called $\text{Satellites}(d)$:

For each $d \in S$ do

$S_d := \{A : A \in \mathbb{P}_{\text{neg}} : d \in [[A]]\}$

$\text{Satellites}(d) = \bigcap_{A \in S_d} [[A]]$

A tentative description is constructed by forming the union of all the $\text{Satellites}(d)$ (for each $d$ in $S$). A $\text{Description}$ is successful if it evaluates to the target set $S$; otherwise the algorithm returns $\text{Fail}$. If $\text{Fail}$ is returned, no Boolean description of $S$ is possible.

**Description By Satellite sets (DBS):**

$$ \text{Description} := \bigcup_{d \in S} (\text{Satellites}(d)) $$

If $\text{Description} = S$ then Return $\text{Description}$ else $\text{Fail}$

DBS is computationally cheap: it has a worst-case running time of $O(n.p)$, where $n$ is the number of objects in $S$, and $p$ the number of atomic properties. Rather than searching among many possible unions of sets, a target $S = \{s_1, ..., s_n\}$ is described as the union of $n$ Satellites sets, each of which equals the intersection of those (at most $p$) sets in $\mathbb{P}_{\text{neg}}$ that contain $s_i$. Descriptions can make use of the Satellite sets computed for earlier descriptions, causing a further reduction of time. Satellites sets can even be calculated off-line, for all the elements in the domain, before the need for specific referring expressions has arisen.\(^1\)

\(^1\)Compare Bateman (1999), where a KB is compiled into a format that brings out the commonalities between objects before the content of a referring expression is determined.
The descriptions produced by DBS are often lengthy. To identify the target set $S = \{c,d,e\}$, for example, the property *poodle* would have sufficed. The algorithm, however, uses three satellite sets:

- $S_c = \{\text{dog, black, poodle, white}\}$. $\text{Satellites}(c) = \{c\}$
- $S_d = \{\text{dog, white, poodle, black}\}$. $\text{Satellites}(d) = \{d,e\}$
- $S_e = \{\text{dog, white, poodle, black}\}$. $\text{Satellites}(e) = \{d,e\}$

Consequently, the boolean expression generated is

$$(\text{dog} \cap \text{black} \cap \text{poodle} \cap \text{white}) \cup (\text{dog} \cap \text{white} \cap \text{poodle} \cap \text{black}) \cup (\text{dog} \cap \text{white} \cap \text{poodle} \cap \text{black}).$$

The algorithm’s profligacy makes it easy, however, to prove the following completeness theorem:

**Theorem 1a. Full Boolean Completeness:** For any set $S$, $S$ is equivalent to a Boolean combination of properties in $\mathbb{I}_\neg$ if and only if $\bigcup_{d \in S} (\text{Satellites}(d))$ equals $S$.

**Proof:** The implication from right to left is obvious. For the reverse direction, suppose $S \neq \bigcup_{d \in S} (\text{Satellites}(d))$. Then for some $e \in S$, $\text{Satellites}(e)$ contains an element $e'$ that is not in $S$. But $e' \in \text{Satellites}(e)$ implies that every property in $\mathbb{I}_\neg$ that holds true of $e$ must also hold true of $e'$. It follows that $S$, which contains $e$ but not $e'$, cannot be obtained by a combination of Boolean operations on the sets in $\mathbb{I}_\neg$.

Under reasonable assumptions, the completeness of the DBS algorithm follows directly:

**Theorem 1b. Completeness of DBS.** Assume there are at most finitely many properties. Then if an individual or a finite set can be individuated by any Boolean combination of properties defined on the elements of the domain, then DBS will find such a combination.

The limitation to finite sets is unrelated to the idea of using $\bigcup_{d \in S} \bigcap_{A \in S_d} [A]$ but stems from the fact that DBS addresses the elements $d \in S$ one by one. Likewise, the limitation to finite sets of properties stems from an assumption that the set $\mathbb{I}_\neg : d \in [A]$ needs to be constructed in its entirety. Implementations of DBS that do not make these assumptions are conceivable.

A brief note on related algorithms. A logic-based mechanism for optimising REs, using techniques familiar from the simplification of electronic circuits, such as the Quine-McCluskey algorithm [McCluskey, 1965] was proposed in [van Deemter, 2002]. The proposal of [van Deemter and Halldórsson, 2001] is open to the same techniques. Additionally, generation can be streamlined by only adding information to a description that changes its extension. For example, in the computation of $S_c$ above, adding *white* does not change the satellite set $S_c$. Gardent [Gardent, 2002] drew attention to situations where earlier proposals produced very lengthy descriptions, and proposed a reformulation of REG as a constraint satisfaction problem (see also [Horacek, 2004]). Horacek’s paper also introduces the idea of generating descriptions of sets in Disjunctive Normal Form (DNF; unions of intersections of literals), leading to more naturalistic descriptions. (This idea has been taken forward in later work [Gatt, 2007], [Khan et al., 2012].) We leave these matters aside, returning to our main theme: the expressive power of REG algorithms.

---

2The statement and proof of this theorem has been updated from [van Deemter and Halldórsson, 2001] because of a slight flaw in the original.
The proposals discussed above use properties that apply to individual objects. To refer to a set, in this view, is to say things that are true of each member of the set. Such REs may be contrasted with *collective* ones (e.g., “the lines that run parallel to each other”, “the group of 4 people”, etc.) whose semantics throws up many problems (e.g., Scha and Stallard [Scha and Stallard, 1988] or Lønning [Lønning, 1997]). Initial ideas about the generation of collective plurals were proposed in [Stone, 2000]; a version based on Dale and Reiter’s incremental algorithm can be found in [van Deemter, 2002]. The algorithm proposed there can be easily modified to make use of the satellite set of a set (rather than of an individual, as in the present section).

3 Knowledge Representation

This section will show how recent work has started to demonstrate the benefits of coupling REG with modern Knowledge Representation (KR) techniques. We show that these techniques have allowed REG algorithms to produce REs (i.e. descriptions that single out a referent) in a much larger range of situations than before: In the terminology of section 1, they have increased the expressive power of REG algorithms.

3.1 Relational Descriptions

Many REG algorithms are restricted to one-place predicates, shunning relations involving two or more arguments. This does not mean that REs like “the man who feeds a dog” is impossible to generate: the database can model dog-feeding as if it were a one-place predicate without internal structure, instead of a relation between two animals; this means that the algorithms in question are unable to identify one entity via another, as when we say “the man who feeds a dog (who chased a cat, who ...)”, and that the REG program is forced to represent “feeds a dog”, “feeds a black dog”, and so on, as if they were logically unrelated properties.

One early paper does discuss the generation of relational descriptions from a KB with 1-place relations (like “suit”) and 2-place relations (like “to wear”), making a number of important observations about them [Dale and Haddock, 1991]. First, it is possible to identify an object through its relations to other objects without identifying each of these objects separately. Consider a situation involving two cups and two tables, where one cup is on a table and the other is on the floor. In this situation, neither “the cup” nor “the table” is distinguishing, but “the cup on the table” succeeds in identifying a cup. Secondly, descriptions of this kind can have any level of ‘depth’: in a complex situation, one might say “the white cup on the red table in the kitchen”, and so on. To be avoided, however, are repetitions that can arise from descriptive *loops*, since these do not add information. It would, for example, be useless to describe a cup as “the cup to the left of the saucer to the right of the cup to the left of the saucer …”. Dale and Haddock’s proposal has been modified in later years, often in relation with the Incremental Algorithm [Horacek, 1996, Krahmer and Theune, 2002, Kelleher and Kruijff, 2006, Viethen and Dale, 2008]. These extensions were not aimed so much at increasing the expressive power of the algorithm, but at avoiding expressions that would threaten to lead to linguistically awkward (e.g., lengthy) descriptions.

Yet, the area of relational descriptions cries out for a further increase of expressive power. It is nice to be able to generate “the cup on the table” and “the table with a cup on it”. But why stop there, instead of pressing on to generate “the table with two cups on it” as well, and “the table that only has a teapot on it”? But first we need to discuss some other limitations of existing REG
algorithms.3

3.2 Knowledge Representation and REG

All REG algorithms discussed so far start from simple, tailor-made Knowledge Bases, which express little else than atomic information. This is quite representative of what happens in REG, except that some REG algorithms group properties together around attributes, such as type, colour, size, and so on. For example, the small KB of section 2 might be represented as follows:

- **Type**: dog \{a, b, c, d, e\}, poodle \{a, b\}
- **Colour**: black \{a, c\}, white \{b, e\}

This separation of a property into an Attribute and a Value does not change what objects can be referred to. Some early REG algorithms made use of generic information, but this never went beyond representing the fact that one 1-place property (e.g., being a dog) subsumes another (e.g., being a poodle). To someone versed in modern Knowledge Representation (KR), this is surprising. Here are some of the things that these KBs are unable to express, and that the algorithms discussed so far cannot make use of when referring:

a. Kees is Dutch or Belgian.

b. The relation “part of” is transitive.

c. For all \(x, y, z\), if \(x\) designed \(y\) and \(z\) is a part of \(y\) then \(x\) designed \(z\).

d. For all \(x, y, x\) is to the left of \(y\) if and only if \(y\) is to the right of \(x\).

The disjunctive fact (a) is arguably what one knows about my nationality after seeing an email message from me that was written in Dutch; it is a piece of information that cannot be represented if only atomic information is available (unless “being Dutch or Belgian” is treated as a primitive property). Rules (b), (c), and (d) might be implicitly present in a KB of atomic facts, but there is substantial mileage in representing these rules explicitly, because it allows a knowledge engineer to represent information in a much more economical and insightful way. Consider (b), for example, in a situation where a complex machine is described. Suppose part \(p_1\) is part of \(p_2\) which is part of \(p_3\) which is part of \(p_4\). Axiom (b) allows us to leave it at that: four other facts (that \(p_1\) is also part of \(p_3\) and \(p_4\), and that \(p_2\) is part of \(p_4\)) do not need to be represented separately but can be deduced from the three facts that are represented. In large domains the savings become enormous.

Dissatisfaction with existing REG representation formats was perhaps first expressed by Krahmer and colleagues, who proposed the use of labelled directed graphs [Krahmer et al., 2003], a much-studied formalism that can be used for representing atomic information involving 1-one 2-place predicates. While this graph-based framework has since gained in popularity, and avoids certain problems that earlier treatments of relational descriptions had to face\(^4\), the graph-based framework is not particularly suitable for deduction, or for the expression of logically complex information [van Deemter and Krahmer, 2007]. In the first move that aimed explicitly at letting REG benefit from frameworks designed for the representation and manipulation of logically structured information, [Croitoru and van Deemter, 2007] proposed to analyse REG as a projection problem in Conceptual Graphs. More recently, [Areces et al., 2008] analysed REG as a problem in Description Logic (DL), a formalism which, like Conceptual Graphs, is designed for representing and reasoning with knowledge. It is this idea (traceable at least to [Gardent and Striegnitz, 2007]) that we

---

3 A more elaborate discussion of the subject matter of section 3 can be found in [Ren et al., 2010].

4 In particular, graphs automatically avoid the risk of infinite regress, as in “the bowl to the left of the cup to the right of the bowl to the left of the cup ...”
shall use as our starting point here. The idea is to generate a DL concept such as $\text{Dog} \cap \exists \text{love}.\text{Cat}$ (the set of dogs intersected with the set of objects that love at least one cat), and to check how many individuals turn out to be instances of this concept. If the number is one, the concept is a RE referring to this individual. Fig.1 depicts a small KB involving relations between several people and animals. Here the above-mentioned formula identifies $d_1$ as “the dog that loves a cat”, singling out $d_1$ from the five other objects in the domain.

![Figure 1: Edges from people to animals denote feed relations; edges between animals denote love relations.](image)

3.3 Description Logic for REG

Description Logic (DL) comes in different flavours, based on decidable fragments of first-order logic. A DL-based KB describes concepts, relations, and their instances. One of the most prominent DLs is $\text{SROIQ}$ [Horrocks et al., 2006]. A $\text{SROIQ}$ ontology consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. $\mathcal{A}$ contains axioms about specific individuals, e.g.

- $a : C$. This means that $a$ is an instance of the concept $C$, i.e., $a$ has the property $C$.
- $(a, b) : R$. This axiom means that $a$ stands in the $R$ relation to $b$.

$\mathcal{T}$ contains generic information concerning concepts and relations. This generic information can either take the form $A \sqsubseteq B$, where $A$ and $B$ are concepts (and certain other conditions are fulfilled), or it can express one of a number of properties of a relation $R$, such as the fact that $R$ is symmetric, irreflexive, or transitive; it can also say that two relations $R$ and $S$ are disjoint.

The notion of a concept is recursively defined. All atomic concepts are concepts. Furthermore, if $C$ and $D$ are concepts, and $R$ is a binary relation (also called a role), then so are

- $\top$, $\bot$
- $\neg C$
- $C \cap D$
- $C \cup D$
- $\exists R.C$
- $\forall R.C$
- $\leq nR.C$
- $\geq nR.C$
- $\exists R.\text{Self}$
- $\{a_1, \ldots, a_n\}$

$\top$ is the top concept (denoting the entire domain), $\bot$ the bottom concept (denoting the empty set), $n$ a non-negative integer, $\exists R.\text{Self}$ the self-restriction (the set of objects standing in the relation $R$ to themselves), $a_i$ are individual names and $R$ is a relation which can be atomic or the inverse of another relation ($R^{-1}$). We use $CN$, $RN$ and $IN$ to denote the set of atomic concept names, relation names and individual names, respectively. An interpretation $\mathcal{I}$ (also called a model) is a pair $\langle \Delta^I, \mathcal{I} \rangle$ where $\Delta^I$ is a non-empty set and $\mathcal{I}$ is a function that maps each atomic concept $A$ to the set $A^I \subseteq \Delta^I$, each atomic role $r$ to $r^I \subseteq \Delta^I \times \Delta^I$ and individual $a$ to $a^I \in \Delta^I$. The interpretation of complex concepts is defined inductively, for example $(C \cap D)^I = C^I \cap D^I$, etc.

$\mathcal{I}$ is a model of the KB $\Sigma$, written $\mathcal{I} \models \Sigma$, iff all the axioms in $\Sigma$ are satisfied in $\mathcal{I}$. Notably, a $\Sigma$ can have multiple models. For example when $\mathcal{T} = \emptyset, \mathcal{A} = \{a : A \sqcup B\}$, there can be a model
\(I_1\) such that \(\Delta^{I_1} = \{a\}, a^{I_1} = a, A^{I_1} = \{a\}, B^{I_1} = \emptyset\), and another model \(I_2\) s.t. \(\Delta^{I_2} = \{a\}, a^{I_2} = a, B^{I_2} = \{a\}, A^{I_2} = \emptyset\). For details, see [Horrocks et al., 2006].

DL is often employed in situations where an “Open World” is required, where there can exist individuals not mentioned in the KB. As we shall soon see, this view does not sit easily with REG. In DL, there are different ways of deviating from the Open World perspective. One solution is to close an ontology partly or wholly with a DBox, \(D\) [Seylan et al., 2009]. A DBox is syntactically similar to the ABox, except that \(D\) contains only atomic formulas. Every concept or relation appearing in \(D\) is closed. Their extension is exactly defined by the contents of \(D\), i.e. if \(D \not\models a : A\) then \(a : \neg A\), so it is the same in all the models. Concepts and relations not appearing in \(D\) remain open. DL reasoning can be exploited to infer implicit information. For example, given \(\mathcal{T} = \{\text{Dog} \subseteq \exists \text{feed} . \text{Woman}\} \), every dog is fed by some woman) and \(\mathcal{A} = \{d1 : \text{Dog}, w1 : \text{Woman}\}\), there must be some Woman who feeds d1. When the domain is closed using \(D = A\) we can infer that this Woman is w1 even though there is no explicit relation between \(w1\) and \(d1\).

However, closing ontologies by means of the DBox restricts the use of implicit knowledge (from \(\mathcal{T}\)). The interpretations of the concepts and relations appearing in \(D\) are fixed, so no implicit knowledge concerning them can be inferred. We introduce the notion of an NBox (with \(N\) short for Negation as Failure). The effect of an NBox on a KB is as follows:

Suppose \(O\) is a KB \((T, A, N)\), where \(T\) is a TBox, \(A\) an ABox and \(N\) an NBox that is a subset of \(CN \cup RN\). Then, firstly, \(\forall x \in IN, \forall A \in N \cap CN\), if \(O \not\models a : A\) then \(O \models (x) : \neg A\). Secondly, \(\forall x, y \in IN, \forall r \in N \cap RN\), if \(O \not\models (x, y) : r\), then \(O \models (x, y) : \neg r\).

Like the DBox approach, the NBox \(N\) defines conditions in which “unknown” should be treated as “failure”. But, instead of hard-coding such conditions, it specifies a vocabulary to which this treatment should be applied. Unlike the DBox approach, inferences on this Negation as Failure (NAF) vocabulary are still possible.

3.3.1 Applying DL to familiar REG problems

DL forces one to be explicit about a number of assumptions that are usually left implicit in REG. In discussing these, it will be useful to consider our old example KB again:

\text{TYPE:} \{a, b, c, d, e\}, \text{poodle} \{a, b\}

\text{COLOUR:} \{black \{a, c\}, white \{b, e\}\}

Closed World Assumption (CWA): REG tacitly assumes a closed world. For example, if there are poodles other than those listed in the KB, then these may include \(c\), and if this is the case then “black poodle” no longer individuates \(a\) uniquely. Our REG proposal does not depend on a specific definition of CWA. In what follows, we use the NBox to illustrate our idea. The domain will be considered to be finite and consisting of only individuals appearing in \(A\).\(^5\)

Unique Name Assumption (UNA): REG tacitly assumes that different names denote different individuals. This matters when there are entities that share all their properties. For example, if \(a\) and \(b\) were the same woman (and CWA applied) then the property poodle alone would individuate this individual, and reference would become trivial, which is not the case if \(a \neq b\).\(^6\)

\(^5\)These assumptions cause no problems for the treatment of Proper Names of section 4. If the Proper Name “Shona Doe” is in the NBox, for example, then only those individuals that can be deduced to have this name have it. A Proper Name that is not listed may or may not be a name of an individual in the KB, just like any other property that is not listed.

\(^6\)UNA concerns the names (i.e., individual constants) of DL, not the Proper Names of the English language (section 4).
Below we shall follow these assumptions. We assume that the KB includes $A$, $T$ and $N$. In the usual syntax of SROIQ, negation of relations is not allowed in concept expressions. So, strictly speaking, one cannot compose a concept $\exists \neg feed.Dog$. However, if $feed \in N$, then we can interpret $(\neg feed)^2 = \Delta^2 \times \Delta^2 \setminus feed^2$. For this reason, we will here use negated relations as part of concept expressions, but we need to remain aware that their handling is computationally less straightforward than that of other constructs.

Given a KB, every DL concept denotes a set. If this set is not empty (i.e., it is either a singleton set or a set with at least 2 elements) then the concept can be seen as referring to this set. It is this idea [Gardent and Striegnitz, 2007] that Areces et al. explored, focussing on an ABox, without a TBox. For example, the domain in Fig.1 can be formalised as follows:

\[
\begin{align*}
T_1 &= \emptyset \\
A_1 &= \{ w_1 : Woman, w_2 : Woman, d_1 : Dog, d_2 : Dog, c_1 : Cat, c_2 : Cat, \\
&(w_1,d_1) : feed, (w_2,d_1) : feed, (w_2,d_2) : feed, (d_1,c_1) : love \}
\end{align*}
\]

The algorithm proposed by Areces and colleagues computes all the similarity sets associated with a given KB. Similarity sets generalise the satellite sets of [van Deemter and Halldórsson, 2001] by taking 2-place relations into account (without which DL doesn’t come into its own), and by being explicitly parameterised on a given logic. The parametrisation to a logic is important, because which referring expressions are expressible, and which sets of objects are referable, depends on the logic that provides the REs. (We return to this issue in section 5.) Here we follow the generalisation of Areces et al., but using the word “satellite” instead of the word “similarity” for consistency with section 2, and because it is more suggestive of a non-symmetrical relation (after all, $a$ can be a satellite of $b$ without $b$ being a satellite of $a$). Thus, if $L$ is a DL and $M$ a model then

- $j$ is an $L$-satellite of $i$ relative to $M$ iff, for every concept $\varphi$ in $L$ such that, given $M$, $i \in \| \varphi \|$ implies $j \in \| \varphi \|$.

The $L$-satellite set of an individual $i$ relative to $M$ is defined as \{ $j : j$ is an $L$-satellite of $i$ relative to $M$ \}.

**Theorem:** $L$ contains a RE for $i$ relative to the domain $M$ iff the L-satellite set of $i$ relative to $M$ equals \{ $i$ \}.  

**Proof:** Trivial.

The algorithm of Areces et al. works as follows: It first finds out which (sets of) objects are describable through increasingly large conjunctions of (possibly negated) atomic concepts, then tries to extend these conjunctions with complex concepts of the form $(-) \exists R1.\text{Concept}$, then with concepts of the form $(-) \exists R2.(\text{Concept} \cap (-) \exists R1.\text{Concept})$, and so on. At each stage, only those concepts that have been found through earlier stages are used. Consider, for instance, the KB above. The algorithm starts partitioning the domain stepwise, where each step makes use of all previous ones. During step (3), for example, $w_2$ could only be identified because $d_2$ was identified during stage (2). We simplify by not showing all the concepts generated during the various stages:

1. $Dog = \{ d_1, d_2 \}, Woman = \{ w_1, w_2 \}, Cat = \{ c_1, c_2 \}$.
2. $Dog \sqcap \exists love.Cat = \{ d_1 \}, Dog \sqcap \neg \exists love.Cat = \{ d_2 \}$.
3. $Woman \sqcap \exists feed.(Dog \sqcap \neg \exists love.Cat = \{ w_2 \}$,
$Woman \sqcap \neg \exists feed.(Dog \sqcap \neg \exists love.Cat = \{ w_1 \}$.

We considered this a potentially attractive approach to the issues addressed by [Dale and Haddock, 1991] and [Krrahmer et al., 2003], and took it as the starting point for the proposal of [Ren et al., 2010], which will concern us in the next section.
3.3.2 Applying DL to new REG problems

Certain extensions to this proposal flow naturally from DL’s power to represent non-atomic information. Since Areces and colleagues consider only the ABox, the KB always has a fixed single model. Consequently their algorithm essentially uses model-checking, rather than full reasoning. We will show how reasoning has to be taken into account when implicit information is involved.

Suppose we extend Fig.1 with background knowledge saying that one should always feed any animal loved by an animal whom one is feeding, while also adding an edge to the “love” relation (Fig.2) between d2 and c2: Suppose we close the domain, using an NBox, as follows:

Figure 2: An extension of Fig.1. Dashed edges denote implicit relations, inferred using the TBox.

\[
\begin{align*}
T_2 &= \{\text{feed} \circ \text{love} \sqsubseteq \text{feed}\} \\
A_2 &= A_1 \cup \{(d2, c2) : \text{love}\} \\
N_2 &= \{\text{Dog, Woman, feed, love}\}
\end{align*}
\]

The TBox axiom enables the inference of implicit facts: \((w1, c2) : \text{feed}, (w2, c1) : \text{feed}\), and \((w2, c2) : \text{feed}\) can be inferred using DL reasoning under the above NBox \(N_2\). But further extensions are called for, of a possibly less obvious nature. The algorithms discussed so far take a somewhat limited view of relational descriptions: they allow us to talk about “the woman who feeds a cat”, \((\text{Woman} \sqcap \exists \text{feed} \text{Cat})\) but not about “the woman who feeds two cats”, for example, allowing only existential quantification but excluding other quantifiers. But if only existential quantifiers are used then some referents cannot be distinguished uniquely. In fact, the algorithm fails to identify any individual in Fig.2, because none of the relevant satellite sets are singletons. For example, the satellite set of \(w_1\) (and \(w_2\)) is \(\{w_1, w_2\}\), a set with two elements. The present section will indicate how all the individuals in the KB become referable if other quantifiers and inverse relations are allowed.

But first, we need to ask what level of expressivity ought to be achieved. This is a difficult question, but in attempting to answering it we can benefit from the conceptual apparatus developed in an area of (first) mathematical logic and (later) the formal semantics of natural language, known as the theory of Generalized Quantifiers (GQ), where quantifiers other than all and some are studied (e.g., [A.Mostowski, 1957], [van Benthem, 1986], [Peters and Westerstahl, 2006]). GQs are these researchers answer to a more general question, namely what quantifiers are possible (in logic, and in natural language). Generalized Quantifiers can occur in many different contexts, for example in the context \(Q N V\) (where the noun \(N\) and the verb \(V\) both denote sets), as in “All (some, ten, most, etc.) cats slept”. The most general format for REs that involves a relation \(R\) is The \(N1\) who \(R\) \(Q\) \(N2\)’s, where \(N1\) and \(N2\) denote sets, \(R\) denotes a relation, and \(Q\) a generalized quantifier, as in the women who feed some dogs. An expression of this form is a RE that refers to an individual entity if it denotes a singleton set. Using a set-theoretic notation, this means that the following set has a cardinality of 1:

\[
\{y \in N1 : Qx \in N2(Ryx)\}
\]
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 one or more dogs is one. If \( Q \) is the quantifier \( \text{at least two} \), then it says that the number of women who feed at least two dogs is one; and so on. It will be convenient to write the formula above in the standard GQ format where quantifiers are cast as relations between sets of domain objects \( A \) and \( B \). Using the universal quantifier as an example, instead of writing \( \forall x \in A(x \in B) \), 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 \( y \).

To show 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 SROIQ. The findings are summarised in Tab.1, varying on the example of people feeding dogs. OWL2, a widely used ontology language for the semantic web that is based on SROIQ, can express any of the following types of descriptions, plus disjunctions and conjunctions of anything it can express.

<table>
<thead>
<tr>
<th>( QAB )</th>
<th>( DL )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( \geq n ) Dog ( {z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>2 ( \geq n ) Dog ( \neg{z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>3 ( \geq n \neg) Dog ( {z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>4 ( \geq n \neg) Dog ( \neg{z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>5 ( \leq n ) Dog ( {z</td>
<td>F(y,z)} )</td>
</tr>
<tr>
<td>6 ( \leq n ) Dog ( \neg{z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>7 ( \leq n ) Dog ( {z</td>
<td>Feed(y,z)} )</td>
</tr>
<tr>
<td>8 ( \leq n \neg) Dog ( \neg{z</td>
<td>Feed(y,z)} )</td>
</tr>
</tbody>
</table>

Having all the different quantifiers of SROIQ gives GROWL the ability to refer in a wide class of situations. When \( n = 1 \), type 1 becomes \( \exists \text{Feed.Dog} \), i.e. the existential quantifier. When \( n = 0 \) type 7 becomes \( \forall \text{Feed.Dog} \), i.e. the quantifier only. When \( n = 0 \) type 6 becomes \( \forall \neg \text{Feed.Dog} \), i.e. the quantifier all. In types 2, 4, 6 and 8, negation of a relation is used. This is not directly supported in SROIQ but, as we indicated earlier, given \( \text{Feed} \in \mathcal{N} \), \( \neg \text{Feed} \) can be used in concepts.

Together, these constructs allows the expression of a description such as “women who feed at least 1 but at most 7 dogs”, by conjoining type 1 (with \( n = 1 \)) with type 5 (with \( n = 7 \)). Using negation, it can say “women who do not feed all dogs and who feed at least one nondog” (\( Woman \cap \neg \forall \neg \text{Feed} \neg \text{Dog} \cap \exists \text{Feed} \neg \text{Dog} \)). In addition to Tab.1, SROIQ can even represent relations to self, such as “the dog who loves itself” by \( \text{Dog} \land \exists \text{Love.Self} \), which was infeasible in the approach discussed in [Gardent and Striegnitz, 2007]. Using conjunctions, one can express exact numbers, as in \( \geq 1 \text{Feed.Cat} \cap \leq 1 \text{Feed.Cat} \) (“Men who feed exactly one cat”), or intervals, as in \( \text{Man} \cap \geq 10 \text{Sell.Car} \cap \leq 20 \text{Sell.Car} \) (“Men who sell between 10 and 20 cars”), and so on. Disjunctions extend expressivity even further.

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 all quantifiers expressible in Van Benthem’s binary tree of numbers [van Bentheim, 1986] –
which is generally regarded as highly general— but much else besides (such as the quantifier only, for example). It is true that some quantifiers routinely expressed in English RES are not expressible in OWL2; examples include most and many, and infinitely many [van Deemter, 1984]. It appears, however, that few of these matter in terms of the power of RES to identify their referent, at least in domains of finite size. To see why, consider this simple example: $c_1$ is the only cat that is loved by most women, but it is also the only cat that is loved by two women. In this way, the problematic quantifier most is replaced by an one that is easily expressible in OWL2, namely the quantifier two.

3.3.3 Generating SROIQ-enabled RES

Here we follow [Ren et al., 2010] in showing how the descriptions of the previous section can be generated in REG. Our overall strategy is similar to Areces et al.’s, generating RESs for all subsets of the domain simultaneously, by computing satellite sets. Like its predecessor, the algorithm applies a generate-and-test strategy that composes increasingly complicated descriptions from the vocabulary. Unlike its predecessor, it uses DL reasoning to test whether a description denotes the referent, allowing it to make use of TBox information. Moreover, it uses a large variety of quantifiers, instead of only $\exists$. The algorithm gives preference to syntactically simple descriptions over complex ones. The complexity of a concept can be defined in different ways, depending on whether a quantified expression such as $\exists \text{Feed.Dog}$ is considered more complex, less complex, or equally complex as a conjoined one such as $\text{Dog} \sqcup \text{Cat}$. Ultimately, the choice between different definitions of complexity should be guided by empirical research, but these matters will continue to be left aside here.

Different descriptions can mean the same, e.g. $\neg \forall R.A \equiv \exists R.\neg A$. In the context of the present paper, we are uninterested in the question which of these should be preferred, so we generate descriptions in their unique negation normal form (NNF). A NNF has $\neg$ in front of only atomic concepts (including $\top$ and $\bot$), or self restrictions. The NNF of $\neg C (\neg R)$ is denoted $\neg C (\neg R)$. The use of NNF will substantially reduce the redundancies generated by the algorithm. The algorithm takes a KB $\Sigma$ as its input and outputs a queue $D$ of descriptions. It uses the following sets:

1. The set $RN$ of relation names. For any relation name $R$, $RN$ contains $R$, $\neg R$, $R^-$, and $\neg R^-$. $RN$ contains nothing else.
2. The set $CN$ of concept names. $CN$ contains $\top$ and all expressions of the form $\exists R.Self$ where $R \in RN$. For all of these, and for any concept name $A$, $CN$ also contains the NNF ($\neg A$). $CN$ contains nothing else.
3. The natural numbers set $N$ containing $1, 2, \ldots, n$ where $n$ is the number of individuals in $\Sigma$.
4. The construct set $S$ containing all the connectives supported by SROIQ, namely $\\{\neg, \sqcap, \sqcup, \exists, \forall, \leq, \geq, =\}$.

We are now ready to sketch the algorithm, which is called GROWL.

Construct — Description($\Sigma, CN, RN, N, S$)

INPUT: $\Sigma, CN, RN, N, S$

OUTPUT: Description Queue $D$

1: $D := \emptyset$
2: for each $e \in CN$ do
3: $D := Add(D, e)$
4: for each $d = fetch(D)$ do
for each \( s \in S \) do

if \( s = \sqcap \) or \( s = \sqcup \) then

for each \( d' \in D \) do

\( D := \text{Add}(D, d s d') \)

if \( s = \exists \) or \( s = \forall \) then

for each \( r \in RN \) do

\( D := \text{Add}(D, s r.d) \)

if \( s = \leq \) or \( s = \geq \) or \( s \) is = then

for each \( r \in RN \), each \( k \in N \) do

\( D := \text{Add}(D, s k r.d) \)

return \( D \)

This sketch leaves the order in which constructs, relations, integers and conjuncts/disjuncts are chosen unspecified. In Step 1, \( D \) is initialised to \( \emptyset \). Step 2 to 3 use the procedure \text{Add} to add suitable elements of \( CN \) to \( D \). The details of \( \text{Add} \) depend on the requirements of the application. The version presented in [Ren et al., 2010] uses a simple heuristic, whereby more complex descriptions are only added if they have smaller extensions than all elements of the existing \( D \). Crucially, all these procedures take implicit knowledge (i.e., TBox axioms) into account.

From Step 4 to 14, elements of \( D \) are processed recursively, one by one. \text{fetch}(D) retrieves the first unprocessed element of \( D \). New elements are added to the end of \( D \), making \( D \) a first-come-first-served queue. Processed elements are not removed from \( D \). For each element \( d \) of \( D \), Step 6-14 generates a more complex description and add it at the end of \( D \). Let us call this the \text{extension} of \( D \). Steps 7 and 8 conjoin or disjoin \( d \) with each element of \( D \). Steps 10 and 11 extend \( d \) with all relations of \( RN \). Steps 13 and 14 extend \( d \) with all relations in \( RN \) and all numbers in \( N \). During the extension of \( D \), we do not consider \( s = \neg \) because, in NNF form, negation \( \neg \) appears only in front of atomic concept names.

\( D, RN, N, S \) are all finite, hence Steps 5 to 14 terminate for a particular \( d \in D \). Steps 5-14 generate descriptions of increasing complexity.

\text{GROWL} generates certain types of REs for the first time, for example ones with universal and numerical quantifiers. Although this may be of linguistic interest, this fact alone does not make it relevant for the task that we have set ourselves in this chapter. It is easy to see, however, that the algorithm also makes some entities \text{referable} for the first time. For example, if we apply the algorithm to the KB in Fig.2, the following solutions emerge:

1. \( \{w1\} = Woman \sqcap \exists \neg feed.Cat \), the woman that does not feed all cats.
2. \( \{w2\} = \leq 0 \neg feed.Cat \), the woman that feeds all cats.
3. \( \{d1\} = Dog \sqcap \leq 0 \neg feed.Woman \), the dog that is fed by all women.
4. \( \{d2\} = Dog \sqcap \exists \neg feed.Woman \), the dog that is not fed by all women.
5. \( \{c1\} = Cat \sqcap \leq 0 \neg feed.Woman \), the cat that is fed by all women.
6. \( \{c2\} = Cat \sqcap \exists \neg feed.Woman \), the cat that is not fed by all women.

The further question how important this added expressive power is will be discussed in section 6. Like the other algorithms discussed in this chapter, the algorithm discussed in the above focusses on finding REs, leaving aside which of all the possible ways in which an object can be referred to.
is “best”. Empirical validation of our algorithm – a difficult enterprise in its own right – possibly based on descriptions elicited by human speakers – must be left for future research.

The examples in this section show only the simplest kind of use that can be made of implicit information. Consider an example from [Croitoru and van Deemter, 2007]. Suppose a KB in Description Logic expressed the TBox axiom that every table has exactly one cup on it:

\[
\text{Table} \sqsubseteq (\leq 1 \text{HasOn.Cup} \cap \geq 1 \text{HasOn.Cup})
\]

If there is only one wooden table in the domain then this axiom justifies the RE “the cup on the wooden table”. Examples of this kind show how radical a departure from standard REG we are proposing, because it enables the generation of references to individuals whose existence is inferred rather than explicitly stated. We shall return briefly to this issue at the end of this chapter.

4 A detour: Adding Proper Names to REG

The descriptions on which the previous section focussed can be quite complex. But REG has also let down the structurally simplest of all REs, namely proper names. Some studies have investigated the choice between proper names and other REs [Henschel et al., 2000], [Piwek, 2008]. A handful of works have studied first-mention REs [Siddharthan et al., 2011], but never in connection with REG as understood in this chapter, where generation starts from a Knowledge Base rather than a text. In the present section, I sketch how proper names can find a proper place within REG. The account that I propose will not extend the expressiveness of REG. The logical representations proposed here for representing proper names in the KB will not be based on individual constants, but on properties such as “... has the name ‘Mary’”. Making use of an existing mechanism (i.e., a mechanism for manipulating properties) seems preferable to inventing a new one.

The reason for disregarding proper names in REG is presumably that if proper names are allowed, reference is easy: to refer to an entity, just generate its proper name! But, on ten seconds’ reflection, this is not a defensible position. The first and perhaps most obvious reason is that the choice between a proper name and a full description is often pragmatically important, and so is the choice between different versions of a proper name. References to people come with social implications to do with personal familiarity, affection, and social hierarchy. Consider a family comprised of a mother and father with a son and a daughter, where the son refers to the daughter. The son can choose between a proper name, and a description like “my sister”; the choice would tend to indicate how familiar he believes the hearer to be with the referent. If he wanted to refer to her while addressing his mother, for example, he could say “my sister”, “my sibling”, “your daughter”, “my father’s daughter”, etc. Yet, only (a version of) the proper name would be considered normal. Relation-denoting words like “Aunt” complicate matters further, by adding a descriptive element to a name. The expression ‘Aunt Agatha’, for example, would be quite normal to use when referring to someone who is the aunt of the speaker, while addressing herself or another family member (except possibly where that family member is the referent’s son or daughter); it would be dis-preferred in most other situations. The pragmatic complexities are clearly considerable.

The second reason for taking proper names seriously in REG is purely semantic, deriving from the fact that proper names are sometimes, but not always, known. Suppose, in 1997, you asked me who was the most influential author of the year 1996, and I responded “that must surely be the author of the novel Primary Colors”, then this would have been a good description of Joe Klein, the political commentator who had written this book, about a character resembling the then-president Bill Clinton, anonymously. While this example is extreme, there is nothing extreme about using
proper names as part of descriptions: we routinely refer to people as “the CEO of Honda” (where “Honda” is a proper name denoting a company), “the Principal of Aberdeen University”, and so on, particularly in situations where the role of the person is more important than his or her name. Surely, it is time for REG to turn proper names into first-class citizens.

To take these matters into account, we shall assume that each individual in the KB comes not just with a number of properties but sometimes with a proper name as well that will be assumed to be known to the hearer. Some individuals may have a proper name, while others may not. For simplicity we shall assume that reference by proper name is preferred over reference by properties, so “Joe Klein’s dog” is preferred over “The dog of the author of Primary Colors”.

This example gets us to the third reason for treating proper names seriously: one likes to think of proper names as unambiguous, but in practise, proper names are seldom unique. There must be numerous individuals named Joe Klein, for example, which is why I introduced him above as “Joe Klein, the political commentator who ...”. A straightforward approach will regard a proper name as a property that is true of all individuals that have this name. For example,

\[\text{(being named) Joe Klein is a property of all those individuals named Joe Klein}\]
\[\text{(being named) Joe is a property of all those individuals named Joe}\]
\[\text{(being named) Klein is a property of all those individuals named Klein}\]

If this sounds naive then it may help to realise that, in realistic settings where domains may be very large, REG must always come with a mechanism for the management of salience (e.g., [Krahmer and Theune, 2002]). Thus, “the dog” doesn’t denote the only dog in the universe, but only the most salient dog. We assume here that the same idea applies to proper names, thus “Joe Klein” will individuate an entity if it is the most salient entity in the KB whose name is Joe Klein. Similarly, “the political commentator Joe Klein” will be considered a legitimate reference to \(o\) if \(o\) is the most salient political commentator in the KB whose name is Joe Klein. To make this work for parts of names as well as full proper names, the easiest approach is to associate a set of names with each individual, for example:

occupation: political commentator
nationality: American, names: \{Joe Klein, Joe, Klein\}.

Since longer versions of a person’s name will be true of only some of the individuals who have a shorter version of that name, the different values of the NAME attribute will tend to subsume each other. Just like all dogs are canines, and all canines are mammals, so all people who are called Prince Andrei Nikolayevich Bolkonsky are also called Prince Andre Bolkonsky, and all of these are called Andre Bolkonsky. Subsumption relations between the different values of an attribute were proposed in [Dale and Reiter, 1995], but their role in REG has been little explored.

We shall not discuss which types of individuals tend to have commonly known proper names (people, cities and companies come to mind) and which do not (e.g., tables, trees, body parts, atomic particles). Likewise, we will say little about the choice between different versions of a proper name and their pragmatic implications, including the use of titles and honorifics (e.g., the properties of being male and being named Klein might be realised as “Mr Klein”). In all these areas there is plenty of space for future research. To sum up our proposal,

- Each individual is associated with an attribute NAMES.
• For a given individual, the set of values of NAMES can be empty (no name is known), singleton (one name) or neither (several names).

• Different individuals can share some or all of their names.

• REG algorithms will treat the NAME attribute in the same way as other attributes.

Even though the focus of this chapter is on questions of expressive power, rather than pragmatic felicity, it is worth mentioning that names are often the “canonical” way of referring to an entity. Standard mechanisms could be invoked to favour names at the expense of other properties, including the preference orders of [Dale and Reiter, 1995], or even (in view of the fact that names are often short) a preference for brevity. This can be likened to the idea that numbers can be written in canonical form by writing them in decimal form: $3 + 1$ and $\sqrt{16}$ denote a bona fide number, but only 4 is the canonical form of that number. But just as there can be good reasons for writing $10^{12}$ or $\sqrt{2}$, there can be good reasons for avoiding Proper Names as well. If you know the name of the Director of Inland Revenue, this does not mean that “Please contact the Director of Inland Revenue” would be better worded as “Please contact Mr X” (where X is his name). When reference to sets is considered, the preference for proper names as canonical REs becomes even more doubtful, because listing proper names does not necessarily make for a short description (compare the description “the citizens of China” with a listing of all the elements of this set). Descriptions differ in terms of their relevance to the communicative situation, but until pragmatic issues like this have been studied algorithmically, preference for proper names seems a reasonable assumption.

Suppose the facts on the ground are as follows:

Type: woman \{w_1, w_2, w_3\}, dog \{d_1, d_2\}, cat \{c_1, c_2\}
Names: mary \{w_1\}, shona \{w_2, w_3\}, rover \{d_1\}, doggy \{d_1, d_2\} max \{d_2, c_2\}, felix \{c_1\}
Action: feed \{(w_1, d_1), (w_2, d_2), (w_2, d_1), (w_3, d_1)\}
Affection: love \{(d_1, c_1), (w_1, d_1), (w_3, d_1)\}

Then our proposal suggest the following referential possibilities:

- $d_1$: Rover, The dog that loves Felix (“Doggy” could refer to $d_2$ as well.)
- $d_2$: The dog called Max (Just “Max” could refer to $c_2$.)
- $c_1$: “Felix”, The cat loved by Rover
- $w_3$: The woman named Shona, who loves a dog. (“Shona” could be $w_2$; “loves a dog” $w_1$.)

We are now in a position to say more precisely how proper names fit into the story of the expressive power of an REG algorithm. Proper names can evidently add to the store of entities that an English noun phrase can individuate: without proper names, one might occasionally struggle to individuate an entity ($w_3$ above is an example). In this regard, however, they are no different from other parts of speech: leave out an adjective or a verb from the language and it might become difficult to individuate an entity as well ($w_2$ above is an example). We have show that proper names can be represented as properties that are treated in the same way as other properties by REG algorithms. Thus, the device of using proper names does not add to the expressive power of REG algorithms.

5 Referability

In the Introduction to this chapter, we listed two questions that research into REG has sought to answer, and we added a third one, which had not been tackled explicitly before:

3 Given a KB, which entities can be referred to (i.e., identified)?
We have seen that the REG algorithms of the 1980’s and 1990’s were unable to refer to entities that later algorithms can refer to. The limitations that came to light arose from an inability to represent negation, disjunction, and most quantifiers. So where are we now? Is the GROWL algorithm, presented in section 3, logically complete for instance?

A problem with this question is that it does not take into account two important parameters, namely what types of information may be contained in the KB and what types of information may be expressed by the RE. For example, if REs can only express conjunction (i.e., union) between atomic properties then, given this limitation, early REG algorithms were logically complete.\(^8\) It is only when negation is added to the language that these algorithms reveal their limitations.

One also needs to make assumptions about the type of information that the KB can contain. A good way to think about this is in terms of the model within which the target referent “lives”. It would be unfair to complain that GROWL is unable to exploit epistemic modalities, for example, since these are not in the KB. In the same way, we cannot expect a REG algorithm to use 3-place relations, as in “the person who gave a present to a child” (which involve a giver, a present and a beneficiary), until such relations enter the model. In what follows, we shall show that there does exist an absolute sense in which an individual may be referable (and a REG algorithm logically complete), provided some assumptions are made concerning the information expressed in the KB.

In section 3, where we were using Description Logic, the notion of an interpretation was defined as a pair \( \langle \Delta^I, \cdot^I \rangle \) where \( \Delta^I \) is a non-empty set and \( \cdot^I \) is a function that maps atomic concepts \( A \) to \( A^I \subseteq \Delta^I \), atomic roles \( r \) to \( r^I \subseteq \Delta^I \times \Delta^I \) and individuals \( a \) to \( a^I \in \Delta^I \). Implicit in this definition is the idea that a model contains individuals, which belong to concepts (i.e., properties) and are connected to each other by roles (i.e., 2-place relations). Let us assume our present models to be exactly like that. In this way, we are simplifying matters, because we are disregarding functions, and relations of higher arity (though we believe these particular simplifications not to be crucial, see below). Additionally, we disregard individual constants, because this is common in REG. The resulting models are graphically depictable as in Figure 1.\(^9\) In connection with REG it is, moreover, reasonable to assume these states of affairs to have only finitely many entities. Under these assumptions, what would it mean for a REG algorithm to be logically complete in an absolute sense, that is, regardless of the logical language employed?

Is it ever possible to prove that one element of \( \Delta^I \) cannot be distinguished from another one? More generally, is there such a thing as “the” satellite set of an entity, regardless of the logical language under consideration? Since, clearly, negation can be expressed in some logical languages, we can simplify this question by making it symmetrical, asking under what conditions one can prove that two entities, \( x \) and \( y \), cannot be distinguished from each other. It is obvious that some models contain elements that are indistinguishable in an absolute sense. Consider a model that contains two men and two women, happily arranged as follows:

\[
\begin{align*}
  m_1 &: Man, & w_1 &: Woman, & (m_1, w_1) &: Love \\
  m_2 &: Man, & w_2 &: Woman, & (m_2, w_2) &: Love
\end{align*}
\]

There is nothing to separate the two men, or the two women for that matter: the two men, for example, take up completely analogous positions. The model can be seen as consisting of two “halves” which are completely isomorphic to each other (i.e., there exists a 1-1 mapping between

---

\(^8\)See [van Deemter, 2002], however, where it was shown that versions of the Incremental Algorithm that exploit Attribute-Value representations fall short of logical completeness if the Values of a given Attribute overlap; versions of the algorithm based on properties (i.e., without separating Attributes and Values) were shown to be logically complete.

\(^9\)Equivalently, one could use the graphs of [Krämer et al., 2003], where properties are represented as “looping” arcs, which connect a node with itself.
the two halves that respects all 1- and 2-place relations). The same is true in a variant of the model that has only one woman, loved by both men:

\[
\begin{align*}
m_1 & : Man, w_1 : Woman, (m_1, w_1) : Love \\
m_2 & : Man, w_1 : Woman, (m_2, w_1) : Love
\end{align*}
\]

This example shows overlapping halves (because \(w_1\) belongs to each half) which are nonetheless isomorphic, which once again captures the intuition that there’s nothing to separate \(m_1\) and \(m_2\).

In this example and the previous one, \(m_1\) and \(m_2\) cannot be distinguished from each other. What these observations suggest is (put informally) that two objects can only be told apart if (unlike, for example, \(m_1\) and \(m_2\) in the two examples above) they take up different parts in the “topology” of the model. This idea will underly the following account.

In order to formalise this idea, it will sometimes be convenient to disregard the direction of the role-denoting arrows in our graphs, viewing them as undirected arcs. Given an entity \(d\) in the model. This idea will underly the following account.

In order to formalise this idea, it will sometimes be convenient to disregard the direction of the role-denoting arrows in our graphs, viewing them as undirected arcs. Given an entity \(d\) in \(M\), we now define the model generated by \(d\) (abbreviated as \(M(d)\)) as the directed part of \(M\) that is reachable from \(d\). More precisely, \(M(d)\) is the result of restricting the model \(M\) (with its set of individuals \(\Delta\)) to the set \(\text{Reach}(\Delta, d)\) (“the subset of \(\Delta\) reachable from \(d\)”) consisting of all those objects in \(\Delta\) to which one can “travel” from \(d\) following the arcs regardless of their direction, also including the starting point \(d\) itself:

1. \(d \in \text{Reach}(\Delta, d)\).
2. For all objects \(x\) and \(y\) and for every role \(r\), if \(x \in \text{Reach}(\Delta, d)\) and \((x, y) \in r\) then \(y \in \text{Reach}(\Delta, d)\).
3. For all objects \(x\) and \(y\) and for every role \(r\), if \(x \in \text{Reach}(\Delta, d)\) and \((y, x) \in r\) then \(y \in \text{Reach}(\Delta, d)\).
4. Nothing else is in \(\text{Reach}(\Delta, d)\).

Using this perspective, we shall prove that an object is referable if and only if it can be referred to by means of a formula of First-Order Predicate Logic With Identity (FOPL). The key is the following theorem, which states that two elements, \(a\) and \(b\), are indistinguishable using FOPL if and only if the models \(M(a)\) and \(M(b)\) generated by \(a\) and \(b\) are isomorphic, with \(a\) and \(b\) taking up analogous positions in their respective submodels. We write \(\varphi(x)^M\) to denote the set of objects in \(M\) (i.e., in \(\Delta\)) that satisfy the FOPL formula \(\varphi(x)\).

The proof will make use of the idea that, given finite model \(M\), a FOPL formula describing \(M\) completely up to isomorphism can be “read off” \(M\):

**Reading off** a description of \(r\) from a submodel model \(M'\) of a model \(M\):

1. Logically conjoin all atomic \(p\) such that \(M' \models p\).
2. Close off all properties and relations by adding to the conjunction, for each constant \(a\) and property \(C\) in \(M\) such that \(M' \not\models C(a)\), the new proposition that \(\neg C(a)\). Similarly, for all constants \(a\) and \(b\) in \(M\) and for each relation \(R\) in \(M\), such that \(M' \not\models a R b\), add the new proposition \(\neg a R b\).
3. Add inequalities \(a \neq b\) for all pairs of constants \(a, b\) occurring in the conjunction.
4. In the resulting conjunction, replace all constants, except \(r\), by (different) variables.
5. Quantify existentially over all the variables in the conjunction by adding \(\exists\) at the start of the formula.

Step 2 is reminiscent of McCarthy-style circumscription [McCarthy, 1980]. Note that this step takes into account all constants and relations that occur in \(M\), regardless of whether they occur in \(M'\). To see that this is necessary suppose \(M\) is as follows, and our referent is \(m_1\):
then $m_1$ cannot be identified without noting that he does not love $w_2$ and $w_3$, individuals living outside $M(m_1)$. Step 2 accomplishes his by adding $\neg \text{Love}(m_1, c_2)$ and $\neg \text{Love}(m_1, c_3)$. After step 5, this comes out as the information that the referent $m_1$ (unlike the distractors $m_2$, $w_1$, $w_2$ and $w_3$) loves exactly one child. The complexity of step 2 is linear in the size of $\Delta$; the complexity of the entire "Reading off" process is quadratic in the size of $\Delta$ (because of step 3).

Referability Theorem: Consider a finite model $M$ and arbitrary elements $a$ and $b$ of this model. Now $a$ and $b$ are FOPL-indistinguishable $\iff$ there exist a bijection $f$ from $M(a)$ to $M(b)$, such that $f$ respects all properties and 2-place relations and such that $f(a) = b$. \textbf{Proof:} We prove the two directions of the theorem separately.

$\Leftarrow$: Let $f$ be a bijection from $M(a)$ to $M(b)$ as specified. Then one can prove using induction following the structure of formulas that $a \in \varphi(x)^M$ iff $b \in \varphi(x)^M$, for any FOPL formula $\varphi(x)$ (i.e., $a$ and $b$ are FOPL-indistinguishable). Note that this equivalence holds across all of $M$ (i.e., we do not just prove that $a \in \varphi(x)^{M(a)}$ iff $b \in \varphi(x)^{M(b)}$). The key factor is that objects in $M$ that lie outside $M(a)$ are irrelevant to the description of $a$, and analogous for $M(b)$.

$\Rightarrow$: Suppose there does not exist a function $f$ as specified. Then there exist a FOPL formula $\varphi(x)$ such that $\varphi(a)$ is true and $\varphi(b)$ is false (hence $a$ and $b$ are not FOPL-indistinguishable) and conversely. A suitable $\varphi(x)$ can be read off $M(a)$.

Equality ($=$) is crucial here because, without it, one cannot distinguish between, on the one hand, an object that stands in a given relation to one object and, on the other hand, an object that stands in that relation to two different objects. To be referable, an object has to be distinguishable from all other objects in the domain. It therefore follows from the Referability Theorem that an element $a$ of a graph $M$ is referable using FOPL unless there exists some other element $b$ of $M$ such that there exist a bijection $f$ from $M(a)$ to $M(b)$ with $f(a) = b$. Thus, FOPL can characterise precisely those domain elements that take up a unique place in the model. In other words, given the assumptions that we made, then if FOPL cannot refer to a domain element, this element cannot be referred to at all. In other words, FOPL is a good yardstick to measure REG algorithms against.

The finiteness assumption is crucial too, because for infinite models the second half ($\Rightarrow$) of the Referability Theorem does not always hold. To see this, consider, as a counterexample against it, a large model $M$ where there are (for simplicity) no properties, where $R$ is the only 2-place relation, and where $a$ and $b$ take up different but similar positions in the graph of the model, with $\forall x(aRx \leftrightarrow x \in \{a_i : i \in \mathbb{N}\})$ and $\forall x(bRx \leftrightarrow x \in \{b_i : i \in \mathbb{N}\})$, causing $b$ to be related to countably many objects, while $a$ is related to uncountably many objects (such as $a_1, a_2, ..., a_\pi$,...). Now the generated models $M(a)$ and $M(b)$ are clearly not isomorphic, yet no FOPL formula $\varphi$ exists such that $\varphi(a) \land \neg \varphi(b)$ (or the other way round). When only finite models are taken into account, the Theorem does go through; moreover, the proof of its second half ($\Rightarrow$) is constructive. This suggests the possibility of new REG algorithms, which is an issue we shall turn to soon.

If Proper Names are treated as properties (rather than individual constants) in the fashion of section 4 then Proper Names do not change this story. Consider a model just like the first model of this section, but with Proper Names added.

\[
\begin{align*}
m_1 : Man, w_1 : Woman, (m_1, w_1) : Love \\
m_2 : Man, w_2 : Woman, (m_2, w_2) : Love
\end{align*}
\]
If \( w_1 \) and \( w_2 \) are only known by the name of Shona then, once again, the two women cannot be
told apart, and neither can the two men. But if the two women are known by their different family
names as well (as in the KB above) then all four inhabitants of the model can be distinguished from
each other, and the submodel generated by \( m_1 \) is not isomorphic to the one generated by \( m_2 \).

Our statement of the Referability Theorem was limited to properties and 2-place relations,
but it appears that some generalisations can be proven straightforwardly. If models can include
relations of higher arity, for example, then the concept \( \text{Reach}(\Delta, d) \) can be redefined to include all
objects that can be reached through any kind of relation. For instance, to cover 3-place relations,
if \( x \in \text{Reach}(\Delta, d) \) and \((x, y, z) \in \tau^I \) then \( y \in \text{Reach}(\Delta, d) \) and \( z \in \text{Reach}(\Delta, d) \). The
Referability theorem itself should be reformulated analogously, after which it appears that its proof
will go through in the same way as before.

In recent years, REG algorithms have made great strides forward, in terms of their empirical
adequacy but also in terms of their expressive power. To show that even the GROWL algorithm is
not complete, however, let us consider a type of situation mentioned by [Gardent and Striegnitz,
2007] in the context of a different DL-based algorithm, consider a model \( M \) containing two
men and three children. One man adores and criticises the same child. The other adores one child and
criticises the other:

\[
\begin{align*}
\text{m}_1 : & \text{Man}, c_1 : \text{Child}, (m_1, c_1) : \text{Adore}, \\
& (m_1, c_1) : \text{Criticise} \\
\text{m}_2 : & \text{Man}, c_2 : \text{Child}, c_3 : \text{Child}, \\
& (m_2, c_2) : \text{Adore}, (m_2, c_3) : \text{Criticise}
\end{align*}
\]

\( M(m_1) \) is not isomorphic to \( M(m_2) \), so \( m_1 \) and \( m_2 \) are distinguishable from each other in FOPL.
A suitable formula \( \varphi \) true for \( m_1 \) but false for \( m_2 \) can be read off \( M(m_1) \). Part of this (lengthy)
formula says that \( \exists x (\text{Adore}(m_1, x) \land \text{Criticise}(m_1, x)) \). This information about \( m_1 \) would be false of
\( m_2 \), hence FOPL with equality can separate the two men.

So how about GROWL? The DL-style formula \( \text{Man} \sqcap \exists \text{Adore} \sqcap \exists \text{Criticise}.\text{Child} \) (“the man
who adores and criticises one and the same child”) comes to mind, but conjunctions of relations
(such as \( \text{Adore} \sqcap \text{Criticise} \)) are not supported by OWL2, so this is not an option for GROWL. OWL2
is expressive enough to produce \( (\text{Man} \sqcap \exists \text{Adore}.\text{Child}) \sqcap \text{Man} \sqcap \exists \text{Criticise}.\text{Child} \) but, in the
model at hand, this denotes \( \{m_1, m_2\} \), without distinguishing between the two. We state here
without proof that OWL2 (and hence also GROWL) is unable to distinguish \( m_1 \) and \( m_2 \) from each
other.\(^{10}\) This is not so much the fault of the GROWL algorithm, but of OWL2 (i.e., basically the
logic \( \mathcal{SROIQ} \)), but from the point of view of the present section, which focusses one the question
which entities can be individuated in any logic, it is a limitation. In practical terms, it means that the
search for more expressive DLs (which are still computationally efficient) has to go on.

It is worth noting that the procedure of “reading off” a formula off a submodel implements
the idea of a Satellite set in a setting where full FOPL (with identity) is available: because the
formula read off the submodel generated by an entity \( m_1 \) spells out all that is known about \( m_1 \), this
formula does not only represent a logically exhaustive attempt at distinguishing this entity from
one particular distractor \( m_2 \); the formula distinguishes \( d_1 \) from every other entity in the model.

\(^{10}\) No similar restriction hold regarding the disjunction of relations (i.e., roles). For even though role disjunction, as in
\( \text{Man} \sqcap \exists \text{Adore} \sqcup \exists \text{Criticise}.\text{Child} \) (“the men who adore or criticise a child”), is inexpressible, an equivalent concept can
be expressed through disjunctions of concepts, as in \( (\text{Man} \sqcap \exists \text{Adore}.\text{Child}) \sqcup (\text{Man} \sqcap \exists \text{Criticise}.\text{Child}) \).
from which is can be distinguished at all: if the formula happens to be true for any other entity, \(d\), in the model, then \(d\) cannot be distinguished from \(m_1\) at all, hence \(m\) is a member of the Satellite set of \(m_1\).

**Scope for new algorithms.** Although the “reading off” process employed by the Referability Theorem was not intended to produce a REG algorithm, and although FOPL does not have the efficient reasoning support that exists for DL fragments such as SROIQ, the fact that reading off can be done constructively suggest at least the possibility of new REG algorithms. One straightforward algorithm would proceed in three steps: to generate an RE of an entity \(d\) given domain \(M\).

A primitive logically complete REG algorithm:

1. Compute the submodel \(M(d)\) generated by \(d\).
2. Read a FOPL formula \(\phi\) off \(M(d)\). Finally,
3. If \(d\) is the only object for which \(\phi\) is true then optimize \(\phi\) and produce a RE, else no RE exists.

Step 2 will often produce unnecessarily lengthy descriptions, so optimisation would be necessary. A promising direction for solutions is local optimisation [Reiter, 1990], by removing conjuncts that are not necessary for producing a uniquely referring expression (i.e., an RE). To find descriptions that meet the empirical requirements associated with question 2 of the Introduction – which is not the aim of this chapter – additional optimisation would likely have to be applied. As we shall see in the next section, it may be difficult to select the empirically most appropriate RE without considering its realisation in actual words.

This section has made explicit, for the first time, what it means for a REG algorithm to be logically complete in an “absolute”, FOPL-related, sense, and observed that against this yardstick, existing REG algorithms still fall short, because certain types of REs are still beyond our grasp, causing some entities to be un-referable even though a FOPL-based formula can identify them uniquely. The outlines of a new FOPL-based algorithm have been sketched, which is logically complete in an absolute sense, but whose details are yet to be worked out.

### 6 Why study highly expressive REG algorithms?

We have seen how the expressive power of REG algorithms has grown substantially in recent years, allowing these algorithms to generate novel kinds of REs and allowing them to identify REs that no previous algorithm was able to identify. The mechanisms investigated include Proper Names, Boolean connectives, and a range of generalised quantifiers. How useful are these additions? Could it be that our additions amount to a complex mechanism whose practical pay-offs are relatively modest? In what follows, we offer a number of reasons why this is not the case, and why the potential pay-offs from working on these issues are substantial.

#### 6.1 Sometimes, the referent could not be identified before

Sometimes, the new algorithms allow us to refer (i.e., individuate) where this was previously impossible. This is, of course, the argument that we have used time and again in this chapter: referents

\[11\] “Reading off” proceeds as specified earlier in this section. Note that \(\phi\) can contain a wide range of formulas, including negated ones such as \(\neg \exists x A(c, x)\). Hence, to test whether \(d\) is the only object for which \(\phi\) is true (step 3 above), it does not suffice to proceed analogous to [Krahmer et al., 2003], testing whether \(M(d)\) stands in the relation of of subgraph isomorphism to any other parts of \(M\).
that were not referable in the REG algorithms of, say, 1985, have become referable because of various later extensions in the expressive power of these algorithms. We will not elaborate on these arguments here.

### 6.2 Sometimes the proposed extensions generate simpler REs than before

Some of the REs that recent REG extensions can generate for the first time are simple, and occur frequently. This almost goes without saying for Proper Names, and likewise for quantifiers like “all” (“The woman who programmed all these functions”), “two” (“The man with two lovers”), and “only” (“The dog that only chases cats”). REs of these forms seem fairly common, and easy to understand.

It may well be that speakers and hearers have difficulties with some of the constructs that play a role in the new algorithms: our cognitive difficulties with negation are well attested, for instance. Still, a RE that contains a negation may be less complex than any other RE that manages to single out the referent. Consider a car park full of vehicles, of many different brands. Given the choice between “The cars that are not Hondas need to be removed” and “The Fords, Lexusses, Toyotas, Audis, (...), Seats need to be removed”, the former is shorter and probably preferable. The claim here is not that negation should be used at every opportunity, but that in some situations, negated REs are preferable to ones that contain only conjunctions and disjunctions. This argument is not limited to negation, but applies to quantified REs as well. Expressions like “The man who adores all cars”, “The woman with two suitors”, and so on, may be somewhat complex, but in a particular situation, they may well be simpler than the simplest un-quantified RE that is able to single out the referent.

### 6.3 Simplicity isn’t everything

So far, we have tacitly assumed that a complex description is worse than a simple one, suggesting that a generator should always prefer a simpler one over a complex one. But simplicity is not everything. Suppose, for example, we consider

- “The man who loves all cars” (when a car exhibition is discussed)
- “John’s uncle” (when John is salient, or after discussing someone else’s uncle)
- “The cars that are not Hondas” (in front of a Honda factory).

In all these situations, the RE contains a source of complexity (the quantifier “all”, a relation, or a negation) for which it should be rewarded rather than penalised, because it makes the RE more contextually relevant than a simpler RE would have been.

Additionally, it needs to be recognized that the concept of reference is not clear-cut. We saw in section 1 that REs can be thought of as answers to the questions Who?, What?, Which? [Searle, 1969]. However, such questions can be answered in radically different ways. Consider the question “Who is Said al-Gaddafi”. In some situations this may be answered “The second-eldest son of the president of Lybia” but in others the shoe is on the other foot, and the question “Who is the second eldest son of the current president of Lybia?” may be answered with a Proper name: “Said al-Gaddafi”. Both responses can be said to identify the referent, but they make different assumptions about the space of options within which identification is to take place. Theoretical linguists have discussed the differences between these answers, but this appears not to have led to a computationally interpretable consensus yet.
6.4 A complex content does not always require a complex syntactic form

As was explained in section 1, we have been assuming that the generator performs Content Determination before deciding about the syntactic and lexical form of the RE. We therefore should not assume that the English expression coming out at the end of the generation pipeline was always mapped directly from the logic, mapping each logical connective in the logical description to a separate English expression. A complex formula can sometimes be expressed through simple words. For example,

- \( \text{Man} \sqcap \leq 0 \neg \text{Love.Car} \) (“The man who loves all cars”: the car lover
- \( \text{Woman} \sqcap 2 \text{Givebirth.Baby} \) (“The woman who gave birth to two babies”: the woman with twins

Furthermore, there is no reason to assume that all the information in the RE must materialise within the same utterance. This point is particularly relevant in interactive settings, where reference can result from a sequence of turns in the dialogue. For example, suppose the generator produces a description \( \text{Woman} \sqcap \exists \text{Feed.(Dog} \sqcap \forall \text{Chase.Cat)} \). Parts of this RE may end up scattered over various dialogue turns, the first of which might describe someone as (1) “This woman”, the second might add that (2) “she feeds a dog”, while the third says that (3) “the dog chases only cats”. The information in the RE is dispersed over different phrases, each of which has a simple structure. By breaking complex information down into bite-size chunks, the RE as a whole may well become easier to understand. Further research is needed to test this hypothesis, and to spell out the logical constraints on the dispersal of referential information, which have to do with the monotonicity properties of the different types of quantifier. To see the problem, consider the information in a slightly different RE, \( \text{Woman} \sqcap \leq 0 \neg \text{Feed.(Dog} \sqcap \forall \text{Chase.Cat)} \). An unthinkingly direct translation might say “the woman such that there are no dogs that chase only cats whom she does not feed”. After some simple logical manipulation, this becomes “the woman who feeds all dogs that chase only cats”. The information in this RE cannot, however, be dispersed over three separate utterances, as in (1) “This woman”, (2) “she feeds all dogs”, (3) “the dogs chase only cats”, because the information in the second utterance may well be false.

6.5 Characterising linguistic competence

Finally, exploring the space of possible REs is of theoretical interest. Let us assume, against our better judgment, that empirical research will show beyond reasonable doubt that most or all of the newest batch of REs are useless: no human speaker would ever generate them, and no human hearer would ever benefit from them. I would contend that our understanding of human language and language use would be considerably advanced by this insight, because it would show us which logically possible referential strategies people actually use.

Two analogies come to mind. First, consider center-embedding, for example of relative clauses. This syntactic phenomenon can give rise to arbitrarily deep nestings, which are easily covered by recursive grammar rules yet extremely difficult to understand (e.g., “a man that a woman that a child knows loves”) [Karlsson, 2007]; moreover, there has to exist a finite upper bound on the depth of any noun phrase ever encountered. Does this mean there is no need for recursive rules? A standard response is that recursive grammar rules model the human linguistic competence (i.e., what can be done in principle), whereas human linguistic performance is best understood by additional constraints (e.g., to reflect limitations on human memory and calculation). I believe that the semantically complex expressions of the previous sections may be viewed in the same light. The second analogy transports us back to our discussion of quantifiers. The theory of Generalized
Quantifiers does not stop at a logical characterisation of the class of all possible quantifiers. Instead, the theory focusses on the question what class of quantifiers can be expressed through syntactically simple means (e.g., in one word, or within one noun phrase [Peters and Westerstahl, 2006]). It is this theory which gave rise to the apparatus that enabled us to extend REG algorithms as proposed in section 3.3.3. Once we know what REs are possible, it might be useful to ask – broadly in the spirit of the theory of Generalized Quantifiers – which of these are actually used.

6.6 Whither REG?

In its most radical – and logically speaking most natural – form, the proposals of this paper amount to a radical extension of REG, in which atomic facts are no longer the substance of the KB, but merely its seeds. The change is particularly stark if we drop the assumption that all individuals in the domain must be mentioned by name in the KB (see the end of section 3, involving the axiom that every table has exactly one cup on it). If this assumption is dropped, it becomes possible to refer to individuals whose existence is not stated but merely inferred. REs that do this are extremely frequent in written communication, for example when a newspaper writes about “President Obama’s foreign policy”, “The first day of the year”, “The mother of the newly born”, or “Those responsible for (a crime, a policy, etc.)”. All such REs were out of reach of REG algorithms until recently, but appear on the horizon now. How the axioms on which these REG algorithms are based should be obtained, and how the aim of empirical adequacy (i.e., question 2 of the Introduction to this chapter) can be achieved as well, is left here as an open question.

Acknowledgments

This chapter has benefitted from discussions in Aberdeen’s Natural Language Generation group and from comments by Helmut Horacek, Emiel Krahmer, Chris Mellish, Graeme Ritchie, and Martin Caminada.

References


