Generation of Referring Expressions (GRE) is a key task of Natural Language Generation (NLG) systems (e.g., Reiter and Dale, 2000, section 5.4). The task of a GRE algorithm is to find combinations of properties that allow the generator to refer uniquely to an object or set of objects, called the target of the algorithm. Older GRE algorithms tend to be based on a number of strongly simplifying assumptions. For example, they assume that the target is always one object (rather than a set), and they assume that properties can always only be conjoined, never negated or disjoined. Thus, for example, they could refer to a target object as “the small violinist”, but not as “the musicians not holding an instrument”. As a result of such simplifications, many current GRE algorithms are logically incomplete. That is, they sometimes fail to find an appropriate description where one does exists.\footnote{We use the term ‘description’ to denote either a combination of properties or its linguistic realization.} To remedy such limitations, various new algorithms have been proposed in recent years, each of which removes one or more simplifying assumptions. They extend existing GRE algorithms by allowing targets that are sets (Stone, 2000; van Deemter, 2000), gradable properties (van Deemter, 2000, 2006), salience (Krahmer and Theune, 2002), relations between objects (Dale & Haddock, 1991; Horacek, 1997), and Boolean properties (van Deemter, 2001, 2002).

Recently a new formalism, based on labelled directed graphs, was proposed as a vehicle for expressing and implementing different GRE algorithms (Krahmer et al., 2001, 2003). Although the formalism was primarily argued to support relatively simple descriptions (not involving negations or disjunctions, for example), we will show that it can be used beyond these confines. Far from claiming that this will solve all the problems in this area, we do believe that a common formalism...
would be extremely useful, as a basis for comparing and combining existing algorithms. An additional advantage is that the computational properties of graphs are well understood and efficient algorithms for manipulating graphs are available ‘off the shelf’ (e.g., Mehlhorn, 1984).

In this paper, we will explore to what extent the graph-based approach to GRE can be extended to express a variety of algorithms in this area. Our discussion will be limited to semantic aspects of GRE and, more specifically, to the problem of constructing combinations of properties that identify a referent uniquely (i.e., constructing a distinguishing description). Our main finding will be that most existing GRE algorithms carry over without difficulty, but one algorithm, which focusses on the generation of Boolean descriptions that also contain relational properties, does not. For this reason, we propose an alternative algorithm that produces different types of Boolean descriptions from the original algorithm, using graphs in a very natural way.

The paper is structured as follows. In section 2 we briefly describe the basic graph-based GRE approach. Then, in section 3, we describe how various earlier GRE algorithms aimed at the generation of sets, gradable properties, salience and negated properties can be reformulated in terms of the graph approach. In section 4 we describe two graph-based algorithms for the generation of full Boolean expressions, one based directly on van Deemter (2001, 2002) and one new alternative. In the concluding section, we list some of the new questions that come up when the different types of algorithms discussed in this paper are combined.

2. Graph-based GRE

A number of GRE algorithms were proposed in the 1990s, of which the Incremental Algorithm from Dale and Reiter (1995) is probably the best known. These ‘basic’ GRE algorithms generate distinguishing descriptions of individual objects. The descriptions generated consist of conjunctions of atomic properties that are represented in a shared Knowledge Base (KB) that is formalized as an attribute/value structure. Using the attributes Type, Size, and Holds, for example, a very simple KB may look as follows:

\[
\text{DOMAIN: } \{s_1, s_2, s_3, s_4\}
\]

Type: Musician = \{s_1, s_2\}, Technician = \{s_3\}, Trumpet = \{s_4\}

Size: Big = \{s_1, s_3\}, Small = \{s_2, s_4\}

Holds: s_4 = \{s_2\}