Modular Generation of Relational Query Paraphrases

MICHAEL MINOCK
Department of Computing Science, University of Umeå, C445 MIT-Huset, Umeå 90343, Sweden (E-mail: mjm@cs.umu.se)

Abstract. This article proposes a novel technique to generate natural language descriptions for a wide class of relational database queries. The approach to describing queries is phrasal and is restricted to a class of queries that return only whole schema tuples as answers. Query containment and equivalence are decidable for this class and this property is exploited in the maintenance and use of a phrasal lexicon. The query description mechanism is implemented within the Schema Tuple Query Processor (STEP) system (http://www.cs.umu.se/~mjm/step). Because the said query class is also closed over elementary set operations, it may be reasoned with in a relatively unrestricted manner. This enables a modular separation between a reasoning component and a ‘tactical’ realization component. To demonstrate this modularity, this fragment is shown to be adequate for several cooperative reasoning techniques. Thus the cooperative information system serves as the ‘strategic’ component, deciding what to say, while the generation system acts as the ‘tactical’ component, deciding how to say it. Naturally expressions within the said query language are the interchange language between these two components.

Key words: cooperative information systems, logical form equivalence, modularity, natural language generation, paraphrasing SQL, relational databases, tuple relational calculus

1. Introduction

There is a long standing question within natural language generation about the degree to which a strategic component, which decides ‘what to say’, should be isolated from a tactical component, which decides ‘how to say it’. Advocates of a high degree of integration cite the myriad ways in which an interleaved strategy, with bi-directional information flow, improves the quality of generation. Advocates of a strict, modular separation, point to the fact that separating generation into two (or more) modules has desirable engineering properties. In either case, it was observed in the early 1990’s that fielded NLG systems tend to have pipelined, modular architectures with the vast majority using some type of semantic network as the knowledge representation language (Reiter, 1994). The strategic decision of content
determination usually results in an utterance specification which is realized by a tactical generator using unification grammars (Kay, 1979; Penman Project, 1989; Elhadad, 1993).

The work here adopts a modular approach in the context of providing natural language generation for cooperative information systems. Cooperative information systems are dialog systems over relational (or deductive) databases that draw their main inspiration from Grice’s maxims (Grice, 1975). Such systems are based on the semantics of relational databases, and thus the deduction is typically restricted to standard first-order logic. When information must be related to the user, utterance specifications are just fragments of logic embedded within one specific communication act or another. These communication acts may be as simple as requests for query confirmation, to intensional comments about either the database’s integrity or state relative to the user’s query. In all of these cases however, content may be expressed in the form of ‘query’ expressions, thus illustrating the primacy of query paraphrasing (Codd, 1974; Lowden and de Roeck, 1986; Ljungberg, 1991).

Pragmatic aspects of specific communication acts are simply templates that fuse together any number of noun phrase query paraphrases with canned text. In summary, from the perspective of natural language generation, the cooperative information system is viewed as providing the services of a strategic generator, ‘deciding what to say’, while the query paraphraser component serves the role of tactical generator, ‘deciding how to say it’.

Appelt (1987) and Shieber (1994) pointed out a rather significant challenge to building such a modular generation system. If one is to maintain a separation between the reasoning component and the tactical component, then the reasoning component must be free to manipulate logical expressions in any way that it sees fit. In turn the tactical component must then be able describe resulting logical expressions, no matter how awkward or redundant. Since one would like to communicate the meaning expressed within the logical expression, then it seems that all equivalent logical formulas should in fact have the same sets of natural language descriptions. The alternative, to map more or less complex logic to more or less complex natural language renderings runs into practical difficulties because unrestricted deduction within the reasoner is almost certain to yield great redundancy. For example the reasoning component may call for descriptions such as the “artists that painted paintings painted by artists that are male,” when the more direct, equivalent\(^1\) description “Male artists that have painted paintings” is not generated because it is not recognized as equivalent.

Thus it seems, as a minimum, that the tactical component be able to recognize when complex input logical expressions are equivalent to simpler forms. This is the so called problem of logical form equivalence. Slightly stronger, logical form subsumption is a determination of whether the set of objects satisfying one formula are necessarily the superset of the objects