

Querying Temporal Databases Using Controlled Natural Language*

Rani Nelken Nissim Francez
Computer Science Department
The Technion
Haifa 32000, Israel

Abstract

Recent years have shown a surge in interest in temporal database systems, which allow users to store time-dependent information. We present a novel controlled natural language interface to temporal databases, based on translating natural language questions into SQL/Temporal, a temporal database query language. The syntactic analysis is done using the Type-Logical Grammar framework, highlighting its utility not only as a theoretical framework but also as a practical tool. The semantic analysis is done using a novel theory of the semantics of temporal questions, focusing on the role of temporal preposition phrases rather than the more traditional focus on tense and aspect. Our translation method is considerably simpler than previous attempts in this direction. We present a prototype software implementation.

1 Introduction

Traditionally, database management systems were designed to store snapshot information, valid at a particular moment of time (state). However, many applications require handling dynamic time-dependent information, pertaining not only to the present, but also to the past and future. Adding temporal support to databases has proved to be a surprisingly

thorny issue (Tansel et al., 1993). A recent drive to consolidate research efforts has led to the design of a consensus temporal data model and associated temporal database query language, SQL/Temporal, an extension of the popular Structured Query Language (SQL) (Snodgrass, 2000).¹ SQL/Temporal represents a significant improvement over standard SQL in allowing programmers to express temporal queries (Snodgrass, 2000). Since temporal database (TDB) implementations are still in their infancy (Boehlen, 1995), there is little practical experience with SQL/Temporal; let alone experience of non-expert users. However, it is our belief that such users are bound to find the expression of complex temporal queries in SQL/Temporal to be extremely difficult.

In an attempt to counter this problem, we present a translation method from controlled natural language (NL) to SQL/Temporal. Following the standard pipeline architecture of such methods, translation is done via an intermediate meaning representation language illustrated in Figure 1. NL questions are first parsed using a grammar in the Type Logical Grammar (TLG) framework (Carpenter, 1998; Morrill, 1998). Simultaneously with parsing, the NL question is translated into a formula of a formal language called L_{Allen} (Toman, 1996), based on the interval operators of (Allen, 1983). The translation is based on an independently motivated novel semantics of sentences modified by *temporal Preposition Phrases (PPs)* (Pratt and Francez 1997; 2000, Nelken and Francez 1999). The constructed L_{Allen} formula is then trans-

* This work was carried out as part of the research project "Semantics of Natural Language Temporal Questions and Interfaces to Temporal Database Systems" sponsored by the Fund for interdisciplinary research, administered by the Israeli Academy of Science. We thank Michael Böhlen, Bob Carpenter and Andreas Steiner for each allowing us to incorporate their software within our own. We also thank Yoad Winter and the anonymous referees for helpful comments on a previous version of this paper. The work of the second author was partially supported by the fund for the promotion of research in the Technion.

¹Through continued design, SQL/Temporal has evolved from predecessor versions named TSQL2 (Snodgrass, 1995) and ATSQL2 (Boehlen et al., 1996). It is expected to be incorporated within the new version of SQL named SQL3.

lated into an SQL/Temporal query, which is subsequently submitted to a prototype TDB implementation for evaluation. Finally, the TDB's answer is presented to the user.

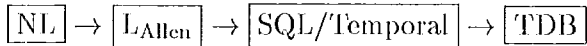


Figure 1: The translation pipeline

We have implemented this method as a prototype software tool, called *QWERTY*, (almost) an acronym for “*Querying with English of Relational Temporal Databases*”. Parsing and translation to L_{AllEn} is done using the TLG Theorem Prover (Carpenter, 1999). The translation from L_{AllEn} to SQL/Temporal is done using an adaptation of a temporal logic (TL) to ATSQL2 translator of (Boehlen et al., 1996). The resulting query is submitted to a prototype TDB implementation, called *TimeDB* (Steiner, 1997). The different modules are coupled into an integrated system implemented in Sicstus Prolog on a UNIX platform with a WWW-based graphical front-end. We discuss some of the directions required in order to turn the system from a research prototype to a working tool.

2 Related work

There is voluminous literature on the design of NL interfaces to general (non-temporal) databases (see (Perrault and Grosz, 1988; Copestake and Jones, 1990; Androutsopoulos et al., 1995) for surveys) and by now their main advantages and disadvantages are well understood. Much less work has been devoted to the design of NL interfaces to TDBs (Clifford, 1990; Ilinrichs, 1988) or other computer systems involving a temporal dimension (Crouch and Pulman, 1993). Of particular relevance is (Androutsopoulos, 1996), who presents a linguistically motivated translation method from NL queries into TSQL2 using an HPSG (Pollard and Sag, 1994) grammar and a TL as an intermediate representation language. Our approach shares many characteristics with (Androutsopoulos, 1996), but there are also important differences, which we point out throughout the paper.

We begin our presentation of the translation method with a brief overview of the TDB, as its structure determines many of the design choices taken in devising the translation method.

3 The TDB

A TDB is a two-sorted first-order structure. The domain consists of a *Data Domain*, D , and a *Temporal Domain* of intervals, T_I , defined as follows. Let T_P be a set (of time points) with a discrete linear order without endpoints, \leq . T_I is defined as the set of pairs: $T_I = \{\langle a, b \rangle \mid a \leq b \wedge a, b \in T_P \cup \{-\infty, \infty\}\}$. A *relational database schema* is a set of single-sorted predicate symbols (R_1, \dots, R_k) . Given a relational database schema ρ , a TDB schema ρ' is the set of two-sorted predicate symbols (R'_1, \dots, R'_k) , where the sort of R'_i is $D^{arity(R_i)} \times T_I$. A TDB *instance* of schema ρ' is a set of relations $R'_i \subseteq D^{arity(R_i)} \times T_I$, where each R'_i is finite.

For instance, assume a database schema ρ consisting of a single binary predicate symbol *work*, storing for each employee the department in which she is employed. The TDB schema ρ' consists of the relation *work'*, called a *valid-time state table*, which adds a temporal argument to the original relation, called the *valid-time* of the table. The temporal argument can be used to store the history (and perhaps even future plans) of departments in which employees are employed. Following a suggestion of (Androutsopoulos, 1996), we also include relations mapping names of calendrical items to temporal intervals in ρ' . For instance, we store a relation *year'* mapping the year 2000 to the interval [1.1.2000-31.12.2000] (which in turn is mapped to an element of T_I).

We now describe the translation process.

4 The translation process

The translation process accepts input NL questions in a controlled subset of NL. Restricting input language in this way enables effective processing of a sufficiently rich fragment while avoiding many of the well-known problems of unrestricted NL. We use a formal grammar in the TLG framework. Our grammar is specially designed for use with a particular TDB schema. Future work will allow easier configuration of the grammar with respect to the schema.

Our grammar is based on work on an independently motivated theory of the semantics of temporality. Most of the research in this field (see (Steedman, 1997) for a survey) has focused on the issues of tense and aspect. We handle tense, but purposefully not aspect, which plays

an important role in (Androutsopoulos, 1996). Aspect, which is used to reflect speakers' temporal viewpoint with respect to reported situations is an important facet of NL temporality. However, its relevance to TDBs is questionable, as it is unlikely that a realistic TDB would actually encode such subjective viewpoints. Moreover, handling aspect requires postulating a more complex data model. For instance, (Androutsopoulos, 1996) augments the TDB model with event-like "occurrence identifiers", and adds an additional argument to temporal relations indicating whether a given event has culminated or not. While such devices may perhaps be linguistically justified, it is unclear whether the TDB community would adopt such augmentations of the model.

Instead, following (Pratt and Francez, 2000; Nelken and Francez, 1999) our focus is on sentences modified by temporal PPs. These PPs are analyzed as variants of standard generalized quantifiers (Barwise and Cooper, 1981), in which quantification is over time. Using this framework, we handle questions that refer explicitly to the temporal dimension (e.g. *When/during which year ...*) as well as questions in which temporality is implied by the TDB context (e.g. *Did Mary work in marketing?, Which employees worked in marketing?*). We handle both clausal and phrasal temporal PPs (e.g. *after John worked in R&D, during every year*). An important strength of this semantic theory is that it allows for arbitrary iteration of PPs (e.g. *one month during every year until 1992*). In addition, our grammar also handles quantification over individuals (e.g. *some employee*), coordination and negation.

Input questions are parsed using a lexicalized type-logical grammar. Lexical items are associated with a syntactic category and a higher-order lambda-term representing its semantics. Taking advantage of TLG's elegantly tight coupling of syntax and semantics, parsing and construction of a semantic representation in the form of an L_{Allen} formula proceed simultaneously, in a bottom-up fashion. We have found using TLG to be advantageous over a feature-structure based formalism (such as HPSG as in (Androutsopoulos, 1996)), since formula construction is an integral part of the parsing and does not require complex ad-hoc manipulations

of feature structures.

Using a particular grammar helps reduce some of the ambiguity inherent in unrestricted NL. For instance, whereas in general a preposition such as *at* is ambiguous between a temporal and a locative interpretation, the choice of the complement NP relative to a given schema-induced grammar deterministically fixes the interpretation. As another example, whereas iterating several temporal PPs (e.g. *during some month every year*) opens up exponential scoping possibilities, some choices are eliminated by world knowledge, which is encoded in the grammar (e.g. *every year* must have higher scope than *some month* since months are included in years and not vice-versa). In cases of remaining ambiguity, the user is presented with all the distinct possibilities. Future work will allow the user to make informed choices between different possible readings, e.g. by presenting him with NL paraphrases of the alternatives.

We translate NL questions into L_{Allen} . The main reason for not translating directly to SQL/Temporal is that the latter is not closed for sub-formulae, i.e. a sub-formula of a well-formed query is not necessarily well-formed. Since L_{Allen} is closed for sub-formulae, compositionally constructing formulae while parsing in a bottom-up fashion becomes much easier.

L_{Allen} is defined as follows (Toman, 1996). Let ρ be the database schema (R_1, \dots, R_k) . Let:

$$L ::= R_i(x, I) | L \wedge L | \neg L | \exists x.L | \exists I.L | x = y | I \sigma J$$

where x, y are variables over D , \mathbf{x} is a vector of such variables, I, J are variables or constants over T_I , and σ is one of the operators: *precedes, meets, overlaps, equals, contains*. L_{Allen} is defined as the set of formulae $\varphi \in L$ that contain at most one free variable over T_I . The *answer* to a formula φ relative to a TDB \mathcal{D} is $\{\mathbf{x}, I | \mathcal{D} \models \varphi(\mathbf{x}, I)\}$.

To illustrate, consider the NL question: *During which years did Mary work in marketing?* The L_{Allen} representation for it is constructed in a bottom-up manner. The meaning representation of the main clause *Mary worked in marketing* is constructed as:

$$\lambda I. \exists J (work(mary, marketing, J) \wedge J \subseteq past \wedge J \subseteq I)$$

In this formula, I denotes a Reichenbachian-like reference time, J denotes a time interval

during which Mary worked in marketing, which is located in the past (the contribution of the tense) and is included within I .

The meaning of the full question is constructed by applying the meaning of the interrogative temporal PP **during which year** to the meaning of the clause. Without going into details, the result is:

$$\text{year}(I) \wedge \exists J(\text{work}(\text{mary}, \text{marketing}, J) \\ \wedge J \subseteq \text{past} \wedge J \subseteq I)$$

The effect of applying the PP is that the variable I is now both free and restricted to be the time of a year. The answer to the formula is the set of intervals I that are years, and during which there is an interval J contained in the past, during which Mary worked in marketing. We allow iterated PPs to apply in a similar manner.

Not every L_{Allen} formula corresponds to an evaluable SQL/Temporal query. In particular, formulae might have an infinite answer. Formulae that are safe from this and related problems are termed *domain-independent* (Gelder and Topor, 1991; Abiteboul et al., 1995). Domain independence is an undecidable semantic property. However, we impose certain syntactic restrictions on generated formulae that ensure it. These restrictions also simplify the translation task from L_{Allen} to SQL/Temporal. This translation is based on a modification of the translator from first-order TL over time-points to ATSQL2 of (Boehlen et al., 1996).

The syntactically restricted version of L_{Allen} we use has the unique advantage of being very close both to the language used in (Nelken and Francez, 1999) on the one hand and to SQL/Temporal on the other. The semantics of NL temporal expressions is often expressed using explicit reference to intervals. Likewise, SQL/Temporal has Allen-style operators over intervals. Androutsopoulos (1996) uses a customized TL as an intermediate language, in which temporal relations are encoded using temporal operators rather than explicit reference to intervals. We have found using a syntactically restricted version of L_{Allen} to be advantageous, as it actually simplifies the translation.

Continuing our previous example, the resulting L_{Allen} formula is subsequently translated into the following SQL/Temporal query:

```
NONSEQUENCED VALIDTIME
SELECT DISTINCT a0.c1 AS c1
FROM work' AS a1, year' AS a0
WHERE VALIDTIME(a0) contains
VALIDTIME(a1)
AND a1.c1 = 'mary'
AND a1.c2 = 'marketing'
AND PERIOD(TIMESTAMP'beginning',
TIMESTAMP'now') contains VALIDTIME(a1)
```

The query asks for the first argument of the relation instance **year'** such that the relation instance **work'** includes a tuple consisting of 'Mary', 'marketing' and a valid time, which is temporally included in the valid time of the year, as well as in the interval starting at the 'beginning' of time and ending 'now' - viz. the past. The TDB responds by returning a table containing exactly the requested year names.

5 Conclusion

The addition of the temporal dimension to database systems increases their power but also their complexity. To increase the usability of TDBs, we present a prototype controlled NL interface to a TDB. Our semantic focus is on the use of temporal generalized quantifiers, based on (Pratt and Francez, 2000), rather than tense and aspect. As argued by (Copestake and Jones, 1990), handling quantification is one of the areas in which NL interfaces have a potential advantage over both formal languages and graphical user interfaces.

In comparison with previous work, we are able to considerably simplify the translation method. First, using TLG, rather than a feature-structure formalism provides a much simpler method for constructing semantic representations. Second, using L_{Allen} as an intermediate meaning representation language yields a much more straightforward translation than using a restricted TL.

One must bear in mind, that our implementation is at the prototype stage. Turning it into a practical tool would require considerable work, as is true of most comparable systems. Future work includes increased NL coverage, adding a disambiguation module, handling nominal and temporal anaphora, allowing multiple-sentence queries, and generation of NL answers from the results presented by the TDB.

References

- S. Abiteboul, R. Hull, and V. Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- J. F. Allen. 1983. Maintaining knowledge about temporal intervals. *CACM*, 26(11):832–843, nov.
- I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases - an introduction. *Natural language Engineering*, 1(1):29–81.
- I. Androutsopoulos. 1996. *A principled Framework for Constructing Natural Language Interfaces to Temporal Databases*. Ph.D. thesis, University of Edinburgh.
- J. Barwise and R. Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.
- Michael H. Boehlen, Jan Chomicki, Richard T. Snodgrass, and David Toman. 1996. Querying TSQL2 databases with temporal logic. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, Avignon, France.
- M. H. Boehlen. 1995. Temporal database system implementations. Unpublished manuscript, Department of Mathematics and Computer Science, Aalborg University.
- B. Carpenter. 1998. *Lectures on Type-Logical Semantics*. MIT Press.
- B. Carpenter. 1999. Type-logical grammar theorem prover. <http://www.colloquial.com/tlg>.
- J. Clifford. 1990. *Formal Semantics and Pragmatics for Natural Language Querying*. Cambridge University Press, Cambridge. Cambridge Tracts in Theoretical Computer Science 8.
- A. Copestake and K. Sparck Jones. 1990. Natural language interfaces to databases. *The Knowledge Engineering Review*, 5(4):225–249.
- R. S. Crouch and S. G. Pulman. 1993. Time and modality in a natural language interface to a planning system. *Artificial Intelligence* 63, 63:265–304.
- A. Van Gelder and R.W. Topor. 1991. Safety and translation of relational calculus queries. *ACM Transaction on Database Systems*, 16(2):235–278, June.
- E. W. Hinrichs. 1988. Tense, quantifiers, and contexts. *Computational Linguistics*, 14:3–14.
- G. Morrill. 1998. *Type Logical Grammar: Categorical Logic of Signs*. Kluwer, Dordrecht.
- R. Nelken and N. Francez. 1999. A semantics for temporal questions. In Geert-Jan M. Kruijf and Richard T. Oehrle, editors, *Proceedings of Formal Grammar 1999*, pages 131–142.
- C. R. Perrault and B. J. Grosz. 1988. Natural language interfaces. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 133–172. Morgan Kaufmann Publishers Inc., San Mateo, California.
- C. Pollard and I. A. Sag. 1994. *Head Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.
- I. Pratt and N. Francez. 1997. On the semantics of temporal prepositions and preposition phrases. Technical Report UMCS-97-4-2, University of Manchester, Department of Computer Science.
- I. Pratt and N. Francez. 2000. Temporal prepositions and temporal generalized quantifiers. To appear in *Linguistics and Philosophy*.
- R. T. Snodgrass, editor. 1995. *The TSQL2 Temporal query language*. Kluwer, Norwell, MA.
- R. T. Snodgrass. 2000. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann Publishers, San Francisco, CA.
- M. Steedman. 1997. Temporality. In J. Van Benthem and A. Ter Meulen, editors, *Handbook of logic and language*, pages 895–938. Elsevier.
- A. Steiner. 1997. *A Generalization Approach to Temporal Data Models and their Implementation*. Ph.D. thesis, Department Informatik, ETH Zurich.
- A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.). 1993. *Temporal Databases: Theory, Design, and Implementation*. Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA.
- D. Toman. 1996. Point vs. Interval-based Query Languages for Temporal Databases. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART PODS*, pages 58–67, Montreal, Canada, June.