In pursuit of decidable ‘logical form’

Michael Minock

TCS/CSC
KTH Royal Institute of Technology, Stockholm, Sweden
minock@kth.se

Abstract

Natural language interfaces commonly map to some type of *logical form* that represents the meaning of the user’s utterance. It is rare however, that this logical form is *bona fide* logic (e.g. expressions that may be tested for satisfiability, etc.). We explore an approach that limits logical form to decidable logics and exploits this property by deeply integrating theorm proving into analysis, pragmatics, reasoning and generation. We explore this in the context of providing natural language interfaces to databases. While support for queries beyond first-order is a requirement (e.g. queries computing average values), we isolate these non-first-order constructs via *markers*, flagging sub-expressions for special treatment within the non-linguistic reasoning component. We shall demonstrate our approach and its merits at SLTC and report on several threads of ongoing work to extend our approach.

1. Introduction

Most natural language interfaces map user utterances to a *logical form* that, although formal, is not semantic in the sense of being able to be conjoined with other logical formulas, transformed into a logical sentence and tested for satisfiability. A system that can truly treat logical form as logic, derives many benefits, including, a principled way to resolve spurious ambiguity during analysis, the capability of producing equivalent paraphrases of equivalent logical expressions (addressing the weak form of the *logical equivalence problem* (Shieber, 1994)), and finally a method to allow the modular integration of sound and complete reasoning into the back-end reasoning and pragmatic components. Below (in section 3) we will will elaborate further, but, in short, systems that do not map to decidable logical form, largely miss out on these advantages.

A problem, however, is that “first-order logic is undecidable!”. Yes, first-order logic is only semi-decidable, but large fragments of it are decidable in the relational case (no function symbols)(Börger et al., 1997; Bernays & Schönfinkel, 1928), moreover, with the advent of modern theorem provers (Weidenbach, et al., 2009) and increasing computational resources, they can be decided quickly. That said, often we must go beyond first-order logic, for example if we need to support aggregation queries (e.g. computing an average salary). We argue that these constructs may be isolated in a principled fashion via *markers*.

2. Approach

Our work is part of the long-term, open-source C-PHRASE effort (Minock, 2010a)(`www.c-phrase.org`). We have identified a logic fragment that captures a quite broad class of queries over entity-relationship modeled databases. The logic, specified in Codd’s tuple calculus, is based on the decidable, Bernays-Schönfinkel class (Bernays & Schönfinkel, 1928) class (Prefix class $\exists^*\forall^*$ over arbitrary arity predicates and no function symbols). Because we restrict our core query language to returning only whole tuple values, we can keep the language closed under query difference. This query language is decidable for query emptiness and containment and is closed under query difference, intersection and union. In addition, we have identified a guarded fragment(Andreka et al., 1998) which keeps these properties, and permits alternation, though not cyclic queries (see (Minock, 2010b) for details).

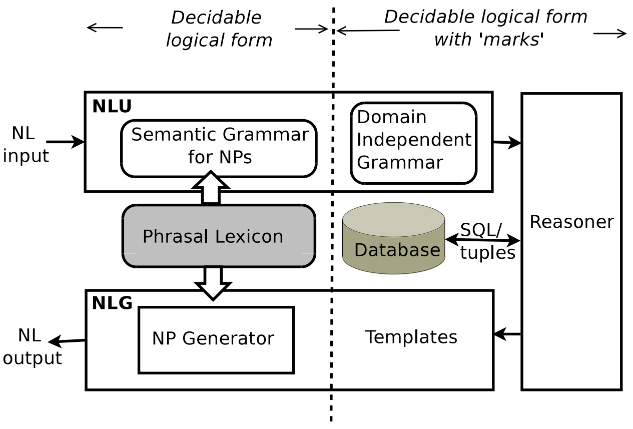

Figure 1: Early/late phases use logical form without marks.

Figure 1 shows our basic approach. User utterances are processed through an NLU (natural language understanding) component which passes *marked* logical forms to a reasoning component which in turn accesses a database (domain data) and generates a response through an NLG (natural language generation) component. The phrasal lexicon is a store of associations between lexical information (in the form of phrases) to elementary expressions in our decidable logic over the relations of the database. This lexicon constitutes what must be built (or perhaps learned) for each new database. From this lexicon, a semantic grammar that recognizes noun phrases is generated. In reverse, this lexicon is used to a generate noun phrase paraphrases. Thus this domain lexicon provides all the information necessary to both analyze and generate noun phrases (including complex pre- and post-modifiers, recursive relative clauses, compound heads, adjuncts, coordination, etc.). The NLU component also includes a domain independent grammar (based on a simplification of X-Bar theory (Jackendoff,1977)) that covers the full interpretation of user utterances. This domain independent grammar inserts *marks* into logical form during processing. In turn templates in the NLU component are selected based on marks. The key point of figure 1 is that logical form may contain marks only on the right side of the diagram, otherwise logical form must be free of marks. Marks themselves are simply keywords associated with chunks of the decidable logical form. When marks are removed, the resulting expression is within the decidable fragment.

As a simple example, the query *"what is average population western US states?"*, is interpreted as the tuple calculus expression: $\{\langle\textbf{:avg}\, x.population\rangle\, | State(x) \wedge x.name \in \{'California', 'Nevada', ...\}\}$, where the material within the brackets is a *mark* of type **:avg**. From the point of view of the reasoner these marks signal different processing strategies. In advanced cases, the reasoner breaks queries with marks into a series of decidable queries without marks. These decidable queries may be reasoned with, and, at the time of paraphrasing, based on the earlier marks, a series of generation calls on decidable queries may be glued together to accurately reflect the meaning of the reasoning steps. Marks such as **:avg**, **:count**, **:sum**, etc. may resolve directly to SQL, which has mechanisms to process these constructs. Some more complex marks include **:consequent** when the user is specifying business rules in natural language (e.g. "a delivery date for an order must always come after the date it was placed"), **:complete** (e.g. "All grades have now been reported for DD1368 this term"), generalized quantifiers such as **:over-n-for-every-m** (e.g. "give the customers who have ordered most of the cheese types."), etc.

3. Utilizing decidability

The primary advantage of using decidable logical form is how it modularizes generation. A typical approach to generate paraphrases of a formula is to walk its syntactic form and, piece by piece, generate language to reflect its meaning. Long ago it was recognized that an alternative to this would be to determine logically what the formula meant, and then generate descriptions based on that (Grosz et al., 1987). Thus if two expressions are logically equivalent, but syntactically different, then they should map to exactly the same set of natural language paraphrases. For example the redundant expression $\{\langle\textbf{:avg}\, x.population\rangle\, | State(x) \wedge (\exists y)(State(y) \wedge x.id = y.id \wedge y.name \in \{'California', 'Nevada', ...\})\}$ generates exactly the same paraphrase(s) as the more direct query above. This enables the reasoning component to work with logical formulas in an unrestricted manner, and then hand the formula to a generator that bases generations on the meaning of the formula, not on the redundant and overly complex syntactic form into which it is transformed. In (Minock,2006) we addressed the paraphrase of logical meaning expressions as a problem of finding re-writings of a logical formula into an equivalent set of elementary formulas with attached lexical information. Such lexical attachments are combined to generate a paraphrase of the exact meaning of the input query. Arguably this addresses at least the weak variant of the well-known *logical-equivalence problem* (Shieber, 1994).

While generation was the original motivation of our exploration of decidable logical form, we have found that there are benefits to having decidability to support *cooperative query answering*, the inclusion of domain constraints, knowledge and even domain instances in reasoning (see (Minock,2006) for initial work on this). Finally another benefit of decidability is the resolution of *spurious ambiguity* during analysis. When parsing, especially when using robust (or learned semantic grammar) techniques, it is common that semantically equivalent queries will be found through multiple analysis paths. Testing for logical equivalence is a clean method to resolve such spurious ambiguity. Finally because the phrasal lexicon is restricted to expressions in the decidable logic, we may organize it in a hierarchical fashion (i.e. as a subsumption hierarchy). Conflicting lexical items are easily identified by inspection. This organization helps with authoring (or perhaps in the future, learning) the phrasal lexicon over a new database.

4. Relevance and future

The topics addressed here have a long history (Codd, 1974; Copestake & Sparck Jones, 1990; Androutsopoulos & Ritchie., 2000), as well as active renewed interest (see for example (Popescu et al,2003; Minock,2005; Lemon & Liu, 2006; Zettlemoyer & Collins, 2007; Cimiano et al., 2007; Wong & Mooney, 2007; Boye & Wirén, 2008; Wu, 2013; Li & Jagadish, 2014)). While earlier work (Alshawi et al,1992), proposed *quasi-logical form* as an intermediary between surface language and full logical formulas (Alshawi et al,1992), such work did not expressly concern itself with decidability. Moreover that work was following the *transportable approach* (Grosz et al., 1987) in which the target representation was domain independent logical form that, via a separate mechanism, was translated to a formal query over a domain model (i.e. database schema). While the transportable approach was seen as a way to leverage linguistic theory for NLIs as well as grammar-based machine translation, complexity in configuring such systems to work over database led us to the more practical 'semantic grammar' approach followed by C-Phrase.

While our approach of inserting marks may be regarded as dodging the hard cases and kicking the problem to higher level paths in pragmatic/reasoning components, a virtue of the approach is that as new decidable classes are discovered or made practical, they may be immediately incorporated, leading to simpler pragmatic/reasoning components. This invites efforts on finding decidable theories to capture fragments expressing aggregation, generalized quantifiers, etc. Approaches similar to (Barrett et al., 2009), although supporting quantifiers in the input problems, hold some promise for future advances in this regard.

While the work has been in the context of NLIs to databases, we are considering an extension of the work to dialogue systems. To achieve this it is necessary to allow for the representation of multiple layers (e.g. the dialogue acts, the domain model, user knowledge, belief, intentions, etc.) under a single logical vocabulary under classical first order semantics. We believe a promising path to explore is to use *handles*, a device that, to our knowledge, was originally proposed under *minimum recursion semantics* (Copestake et al., 2005).

References

H. Alshawi (editor), et-al *The Core Language Engine.* MIT Press, 1995.

H. Andreka, J. van Benthem, and I. Nemeti. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

I. Androutsopoulos and G.D. Ritchie. Database interfaces. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*, pages 209–240. Marcel Dekker Inc., 2000.

C. Barrett, R. Sebastiani, S. Seshia and C. Tinelli Satisfiability Modulo Theories, Handbook of Satisfiability. IOS Press, 825–885, 2009

P. Bernays and M. Schönfinkel. Zum Entscheidungsproblem der mathematischen Logik. Mathematische Annalen, 99:342-372, 1928.

E. Börger and E. Grädel and Y. Gurevich. The Classical Decision Problem. Springer, 1997.

J. Boye and M. Wirén. Robust parsing and spoken negotiative dialogue with databases. *Natural Language Engineering*, 14(3):289-312, 2008.

P. Cimiano, P. Haase, and J. Heizmann. Porting natural language interfaces between domains: an experimental user study with the ORAKEL system. In *Intelligent User Interfaces*, pages 180–189, 2007.

E. Codd. Seven steps to rendezvous with the casual user. In *IFIP Working Conference Data Base Management*, 179–200, 1974.

A. Copestake and K. Sparck Jones. Natural language interfaces to databases. *The Natural Language Review*, 5(4):225–249, 1990.

A. Copestake, D. Flickinger, I. Sag and C. Pollard. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3(1):281-332, 2005.

B. Grosz, D. Appelt, P. Martin and F. Pereira. Team: An experiment in the design of transportable natural-language interfaces. *AI*, 32(2):173–243, 1987.

R. Jackendoff. *X-bar-Syntax: A Study of Phrase Structure, Linguistic Inquiry Monograph 2.* MIT Press, 1977.

Y. Li and H. Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases In *VLDB*, 73–84, 2014.

O. Lemon and X. Liu. DUDE: a Dialogue and Understanding Development Environment, mapping Business Process Models to Information State Update dialogue systems. In *EACL (demonstration systems)*, 2006.

M. Minock. C-Phrase: A system for building robust natural language interfaces to databases. *Journal of Data and Knowledge Engineering (DKE)*, 69(3):290–302, 2010.

M. Minock. Describing and deriving certain answers over partial databases. *Journal of Intelligent Information Systems*, 35(2):245-260, 2010.

M. Minock. Modular generation of relational query paraphrases. *Research on Language and Computation*, 4(1):9–37, 2006.

M. Minock. A Phrasal Approach to Natural Language Interfaces over Databases. *NLDB*, 333–336, 2005.

A. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *Intelligent User Interfaces*, 2003.

S. Shieber. The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190, 1994.

C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda M. and P. Wischnewski SPASS version 3.5. In *22nd International Conference on Automated Deduction (CADE2009)*, pages 140–145, 2009.

Y. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL-2007*, pages 960–967, 2007.

W. Wu. Proactive Natural Language Search Engine: Tapping into Structured Data on the Web. In *Extending Database Technology (EDBT)*, pages 143–148. 2013.

L. Zettlemoyer and M. Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP*, 2007.