Negotiative Spoken-Dialogue Interfaces to Databases

Johan Boye and **Mats Wirén**
Voice Technologies
TeliaSonera Sweden
Johan.Boye@teliasonera.com, Mats.Wiren@teliasonera.com

Abstract

The aim of this paper is to develop a principled and empirically motivated approach to robust, negotiative spoken dialogue with databases. Robustness is achieved by limiting the set of representable utterance types. Still, the vast majority of utterances that occur in practice can be handled.[1]

1 Introduction

The need for spoken dialogue with databases is rapidly increasing as more and more non-technical people access information through their PCs, PDAs and mobile phones. The related research area of natural-language (text-based) interfaces to databases has a long tradition, going back at least to around 1970. Still, a kind of culmination of this research occurred already in the 1980s (for an excellent overview, see Androutsopoulos et al. 1995). The high-end systems of this time, for example, TEAM (Grosz et al. 1985), LOQUI (Binot et al. 1991) and CLE/CLARE (Alshawi 1992, Alshawi et al. 1992) used linguistically-based syntactic analysis and powerful intermediary languages for semantic representation, and were able to engage in continuous dialogue involving complex phenomena such as quantification, anaphora and ellipsis. In part, this was possible because the systems were alleviated from the kind of noisy input generated by speech recognition, since text-based input was the only realistic option at the time.

Although some of today's commercial spoken-language information services also provide access to databases, they do not aim at being general database interfaces in the sense of the systems mentioned above. Rather, they provide a limited view of the database which is relevant to a particular task, for example, finding a train trip that fulfills certain constraints. This restricts the user to asking certain types of question that are compatible with the task, whereas other types are not allowed. In many cases, the system also keeps the initiative to itself by treating the user as an answer-supplier.

An advantage of this cautious approach to spoken dialogue with databases is that it allows a very simple semantic representation of user utterances, typically as flat slot–filler structures representing the propositional contents. For example, the utterance "I'd like a two-room apartment on the South Side" could be represented as [*number_of_rooms* = 2, *area* = *south_side*]. Hence, robust methods like phrase-spotting can be used for extracting the meaning even from noisy input.

However, progress in speech recognition and robust parsing during the last decade opens up the possibility of constructing more general spoken-dialogue database interfaces. In this paper, we will investigate *negotiative* spoken-dialogue interfaces to relational databases. "Negotiative" here means that the user is able to discuss and compare several alternative solutions to a problem (Larsson 2002).

[1] The authors would like to thank the Adapt group at TeliaSonera and KTH for discussions and feedback, and the participants at a presentation of this material at Göteborg University for valuable comments. This work was supported by the EU/HLT-funded project NICE (IST-2001-35293).

More specifically, the goal of the paper is to find an empirically based trade-off between robustness and expressiveness in spoken database dialogue: On the one hand, by deliberately restricting the expressiveness of the semantic representation formalism (keeping it suitably coarse-grained), we make it *apt for robust parsing*. On the other hand, the representation must be sufficiently fine-grained to *capture the distinctions actually made in spoken database dialogue*. We present a formalism that meets these goals, as well as a robust parsing algorithm for mapping speech recognizer output to this representation. The approach has been used successfully in Adapt (Bell et al. 2001), a multimodal dialogue system in an apartment domain.

2 Negotiative dialogue

The following piece of dialogue with the Adapt system provides an illustration of the kind of negotiative dialogue we are aiming to handle.

1. **User:** Are there any two-room apartments on the South Side that cost less than two million?
2. **System:** Do you have any other requirements?
3. **User:** A balcony would be nice.
4. **System:** There are seven such apartments. They are shown on the map.
5. **User:** I see ... this apartment on Horn street ... how much does it cost?
6. **System:** The apartment on Horn Street costs 1,900,000 kronor.
7. **User:** Ok ... is there anything cheaper?
8. **System:** The apartment on Blekinge street costs 1,500,000 kronor.
9. **User:** Hmm ... please tell me more about that apartment.

In utterances 1 and 3, the user homes in on the kind of apartment he wants to consider (the *specification* phase). The system briefly presents a number of results (in utterance 4) by indicating their position on the map, after which the user starts asking questions about the individual apartments, comparing their relative merits (utterances 5, 7, 9). This *negotiation* phase is what distinguishes negotiative dialogue systems from simpler systems.

To handle negotiative dialogue, the system must be able to distinguish specification utterances like “I’d like an apartment with a balcony” from utterances seeking information about a particular apartment, like “Does that apartment have a balcony?”. The system must also be able to handle references to different objects in the same utterance, like “Is there anything cheaper than that apartment on King’s street?”.

3 Domain models

From the user’s point of view, the purpose of a dialogue as exhibited above is to retrieve information about a set of interrelated objects, such as apartments, prices and addresses. The set of all such objects in the domain, together with their relations, constitutes the *domain model* of the system. From the system’s point of view, the goal is then to translate each user utterance into an expression denoting a subset of the domain model (namely, the subset that the user is asking for), and to respond by either presenting that subset or ask the user to change the constraints in case the subset cannot be readily presented.[2]

We will assume that each object in the domain model is typed, and to this end we will assume the existence of a set of *type symbols*, e.g. $\mathsf{apartment}$, $\mathsf{integer}$, money, $\mathsf{street\_name}$ etc., and a set of *type variables* $\mathsf{t}_1, \mathsf{t}_2, \ldots$ ranging over the set of type symbols. Each type symbol *denotes* a set of objects in an obvious way, e.g. $\mathsf{apartment}$ denotes the set of apartments. Both type symbols and type variables will be written with a $\mathsf{sans\ serif}$ font, to distinguish them from symbols denoting individual objects and variables ranging over individual objects, which will be written using an *italicized* font. The expression $\mathsf{t} : x$ is taken to mean the assertion “x is of type t”.

Objects are either simple, scalar or structured.

[2]Naturally, this is somewhat idealized, as there are meta-utterances, social utterances, etc. that are not translatable to database queries. Still, 96 % of the utterances in our Adapt corpus correspond to database queries (compare Section 6).

Objects representable as numbers or strings are simple (such as objects of the type money or $\mathsf{street\_name}$). Scalar objects are sets of simple objects, whereas structured objects have a number of attributes, analogous to C structures or Java reference objects. Typically, structured objects correspond to real-world phenomena on which the user wants information, such as apartments in a real-estate domain, or flights and trains in a travel planning domain.

We will use the notation $x.a$ to refer to attribute a of object x. For example, an apartment has the attributes $size$, $number\_of\_rooms$, $price$, $street\_name$, $accessories$, etc., with the respective types $\mathsf{square\_meters}$, $\mathsf{integer}$, money, $\mathsf{street\_name}$, $\mathsf{set(accessory)}$, etc. Hence if $\mathsf{apartment} : x$ is a true assertion, then so is $\mathsf{square\_meters} : (x.size)$.

Thus, a (structured) object o_1 might be related to another (simple, scalar or structured) object o_2 by letting o_2 be the value of an attribute of o_1. For instance, an apartment a is related to "King's street" by letting $a.street\_name = Kings\_street$. There is a standard transformation from this kind of domain models into relational database schemes (see e.g. Ullman 1988, p. 45), but domain models can also be represented by other types of databases.

We will further assume that types are arranged in a subtype hierarchy. The type t_1 is a subtype of t_2 (written as $\mathsf{t}_1 \leq \mathsf{t}_2$) if $\mathsf{t}_2 : x$ is a true assertion whenever $\mathsf{t}_1 : x$ is a true assertion.

4 Semantic representation formalism

In this section, we describe expressions called "utterance descriptors", which constitute the semantic representation formalism used internally in the Adapt system.

4.1 Constraints

Constraints express desired values of variables and attributes. The following are all examples of constraints:

- $x.street\_name = King\_street$
- $x.price < 2,000,000$
- $balcony \in x.accessories$
- $x.street\_name \in \{King\_street, Horn\_street\}$

4.2 Set descriptors

Set descriptors are expressions denoting subsets of the domain model. They have the form $?\mathsf{t} : x\ (P)$, where P is a conjunction of constraints in which the variable x occurs free. Such a set descriptor denotes the set of all objects x of type t such that P is a true assertion of x. Thus,

$$?\mathsf{apartment} : x\ (x.area = South\_side \\ \wedge\ x.number\_of\_rooms = 2)$$

denotes the set of all apartments whose $area$ attribute has the value $South\_side$ and whose $number\_of\_rooms$ attribute has the value 2.

We may also add existentially quantified "placeholder" variables to a set descriptor without changing its semantics. For instance, the set descriptor above is equivalent to:

$$?\mathsf{apartment} : x\ \exists \mathsf{integer} : y\ (x.area = South\_side \\ \wedge\ x.number\_of\_rooms = y\ \wedge\ y = 2)$$

Thus, set descriptors can also have the form $?\mathsf{t}_1 : x\ \exists \mathsf{t}_2 : y\ (P)$, where P is a conjunction of constraints in which x and y occur.

4.3 Representing context

Utterances may contain explicit or implicit references to other objects than the set of objects sought. For example, when the user says "A balcony would be nice" in utterance 3 of the dialogue fragment of section 2, he further restricts the context (the set of apartments) which was obtained after his first utterance.

Obviously, an utterance cannot be fully interpreted without taking the context into account. Thus the context-independent interpretation of an utterance is a function, mapping a dialogue context (in which the utterance is made) to the final interpretation of the utterance. In our case, a dialogue context is always an object or a set of objects (a subset of the domain model), and the final interpretation is a set descriptor, denoting the set of objects that are compatible with the constraints imposed by the user.

Accordingly, the context-independent interpretation of "A balcony would be nice" is taken to be

$$\lambda S\ ?\mathsf{apartment} : x \\ (balcony \in x.accessories\ \wedge\ x \in S)$$

where S is a parameter that can be bound to a subset of the domain model. Thus the expression above can be paraphrased "I want an apartment from S that has a balcony". The idea is that the ensuing stages of processing within the dialogue interface will infer the set of objects belonging to the context, upon which the functional expression above can be applied to that set, yielding the final answer. In the dialogue example of section 2, S will be bound to the set of apartments obtained after utterance 1.

An utterance may contain more than one implicit reference to the context. For example, "Is there a cheaper apartment?" (utterance 7 of the dialogue fragment of section 2) contains one implicit reference to a set of apartments from which the selection is to be made, and another implicit reference to an apartment with which the comparison is made (i.e. "I want an apartment from S which is cheaper than the apartment y").[3] Hence the representation is:

$$\lambda\mathsf{apartment}\!:\!y\ \lambda S\ ?\mathsf{apartment}\!:\!x\ (x.\mathit{price} < y.\mathit{price}\ \wedge\ x \in S)$$

The contextual reasoning carried out by the Adapt system then amounts to applying this expression first to the apartment mentioned in utterance 6, and then to the set of apartments introduced by utterance 4.

Therefore, we define an *utterance descriptor* to be an expression of the form $\lambda X_1\ \ldots \lambda X_n\ U$, where X_i is either a set variable or a typed variable $\mathsf{t}\!:\!x$, and where U is a set descriptor in which the variables of $X_1 \ldots X_n$ occur free. Thus, an utterance descriptor is a function taking n arguments (representing the context), returning as result a subset of the domain model.

Yet an example is given by the utterance "How much does the apartment cost", which is represented by

$$\lambda\mathsf{apartment}\!:\!y\ ?\mathsf{money}\!:\!x\ (y.\mathit{price} = x)$$

Utterance descriptors can also contain type variables, when sufficient type information is lacking. For instance, "How much does it cost?" would be represented by

$$\lambda\mathsf{t}\!:\!y\ ?\mathsf{money}\!:\!x\ (y.\mathit{price} = x)$$

[3] For comparisons with sets of objects, see section 7.

4.4 Minimization and maximization

In many situations one is interested in the (singleton set of the) object which is minimal or maximal in some regard, for example the "biggest apartment" or the "cheapest ticket". To this end, we will further extend the notion of utterance descriptor.

Consider an expression of the form

$$?\mathsf{t}_1\!:\!x\ \mu\mathsf{t}_2\!:\!y\ (P)$$

where t_2 is a numerical type and P is a conjunction of constraints in which x and y occur. We will take such an expression to denote the (singleton) set obtained by first constructing the set denoted by $?\mathsf{t}_1\!:\!x\ (P)$, and then selecting the object whose value for y is minimal. For instance, the utterance "Which is the cheapest apartment?" would by represented as

$$\lambda S\ ?\mathsf{apartment}\!:\!x\ \mu\mathsf{money}\!:\!y\ (x.\mathit{price} = y\ \wedge\ x \in S)$$

When applied to a context set S, the function above returns an expression denoting the singleton set of the apartment in S whose price attribute has the minimal value. There is also a analogous maximization operator M.

5 Robust parsing

The robust parsing algorithm consists of two phases, pattern matching and rewriting. In the latter phase, heuristic rewrite rules are applied to the result of the first phase. When porting the parser to a new domain, one has to rewrite the pattern matcher, whereas the rewriter can remain unaltered.

5.1 Pattern matching phase

In the first phase, a string of words[4] is scanned left-to-right, and a sequence of constraints and meta-constraints, triggered by syntactic patterns, are collected. The constraints will eventually end up in the body of the final utterance descriptor, while the purpose of the meta-constraints is to guide the rewriting phase.

The syntactic patterns can be arbitrarily long, but of course the longer the pattern, the less frequently it will appear in the input (and the more

[4] Currently, 1-best output from the speech recognizer is used.

sensitive it will be to recognition errors, disfluencies etc.). On the other hand, longer syntactic patterns are likely to convey more precise information.

The solution is to try to apply longer patterns before shorter patterns. As an example, reconsider the utterance "I'm looking for an apartment on King's street", and suppose that "apartment on S" (where S is a street), "apartment" and "King's street" are all patterns used in the first phase. If the utterance has been correctly recognized, the first pattern would be triggered. However, the utterance might have been misrecognized as "I'm looking for an apartment of King's street", or the user might have hesitated ("I'm looking for an apartment on ehh King's street"). In both cases the pattern "apartment on S" would fail, so the pattern matching phase would have to fall back on the two separated patterns "apartment" and "King's street", and let the rewriting phase infer the relationship between them.

5.2 Meta-constraints

The pattern matching rules in the pattern matcher associate a sequence of constraints and meta-constraints to each pattern. The most commonly used meta-constraint has the form $obj\,(\mathsf{t}:x)$ which is added when an object x of type t has been mentioned. For instance, in the Adapt parser, the pattern "apartment" would yield

$$obj\,(\mathsf{apartment}:x_1)$$

whereas the pattern "King's street" would yield

$$obj\,(\mathsf{apartment}:x_2), x_2.street = Kings\_street$$

where x_1 and x_2 are variables. The existence of the object $\mathsf{apartment} : x_2$ is inferred, since in the Adapt domain model, streets can only occur in the context of the $street$ attribute of the $\mathsf{apartment}$ type. If the domain model would include also another type ($\mathsf{restaurant}$, say) that also has an attribute $street$, the pattern could instead yield:

$$obj\,(\mathsf{t}:x_2), x_2.street = Kings\_street$$

where t is a type variable.

The meta-constraint $head\_obj\,(\mathsf{t}:x)$ is a variant of $obj\,(\mathsf{t}:x)$ that conveys the additional information that x is likely to be the object sought. We will illustrate the use of this and other types of meta-constraints in section 5.4.

5.3 Rewriting phase

In the rewriting phase, a number of heuristic rewrite rules are applied (in a fixed order) to the sequence of constraints and meta-constraints, resulting in a utterance descriptor (after removing all meta-constraints). The most important rules are:

- Unify as many objects as possible.
- Identify the object sought.
- Identify contextual references.
- Resolve ambiguities.

The first rule works as follows: Suppose pattern matching has resulted in:

$$obj\,(\mathsf{apartment}:x_1), obj\,(\mathsf{t}:x_2), x_2.street = Kings\_street$$

Then checking whether the two objects x_1 and x_2 are unifiable amounts to checking whether the types $\mathsf{apartment}$ and t are compatible (which they are, as t is a variable), and checking whether an $\mathsf{apartment}$ has an attribute $street$ (which is true). Therefore the result after applying the rule is

$$obj\,(\mathsf{apartment}:x_1), x_1.street = Kings\_street$$

As for the second rule, the object sought is assumed to be the leftmost $head\_obj$ in the sequence. Failing that, it is assumed to be the leftmost obj. In the sequence above, that means $\mathsf{apartment} : x_1$, which results in:

$$?\mathsf{apartment}:x_1\ (obj\,(\mathsf{apartment}:x_1),\\ x_1.street = Kings\_street)$$

No more rewrite rules are applicable on this expression. The final result is obtained by adding the contextual argument S and by removing all meta-constraints:

$$\lambda S\ ?\mathsf{apartment}:x_1\ (x_1.street = Kings\_street\ \wedge\ x_1 \in S)$$

5.4 Example

The utterance "I'd like an apartment on Horn Street that is cheaper than the apartment on King's street" exemplifies the use of several rewrite rules in the Adapt system. First of all, "apartment on Horn Street" yields

$$obj\,(\mathsf{apartment}:x_1), x_1.street = Horn\_street$$

The pattern "cheaper" yields the sequence

$$\begin{aligned}&\mathit{head\_obj}(\mathsf{apartment}\!:\!x_2),\ \mathit{obj}(\mathsf{apartment}\!:\!x_3),\ x_2 \neq x_3,\\ &\quad \mathit{obj}(\mathsf{money}\!:\!y_2),\ x_2.z = y_2,\\ &\quad \mathit{obj}(\mathsf{money}\!:\!y_3),\ x_3.z = y_3,\ y_2 < y_3,\\ &\quad \mathit{ambiguous}(z, \{\mathit{price}, \mathit{monthly\_fee}\}, \mathit{default}(\mathit{price}))\end{aligned}$$

which is appended to the first sequence. The $\mathit{ambiguous}$ meta-constraint conveys that the variable z is one of the attributes price or $\mathit{monthly\_fee}$. If no further clues are against, z will be bound to price.

Finally, the pattern "apartment on King's Street" appends the sequence

$$\mathit{obj}(\mathsf{apartment}\!:\!x_4),\ x_4.\mathit{street} = \mathit{Kings\_street}$$

In the rewriting phase, objects are first unified in a left-to-right order. Thus x_1 and x_2 are unified, but the meta-constraint $x_2 \neq x_3$ prevents unification of x_2 and x_3. Instead, x_3 and x_4 are unified. The ambiguity is resolved (binding z to price). Thereafter, the variable x_2 is identified as the main object, and the implicit contextual reference argument S is added.

$$\begin{aligned}&\lambda S\ ?\mathsf{apartment}\!:\!x_2\ (x_2.\mathit{street} = \mathit{Horn\_street}\\ &\quad x_2 \in S,\ \mathit{obj}(\mathsf{money}\!:\!y_2),\ x_2.\mathit{price} = y_2,\\ &\quad x_2 \neq x_3,\ x_3.\mathit{street} = \mathit{Kings\_street},\\ &\quad \mathit{obj}(\mathsf{money}\!:\!y_3),\ x_3.\mathit{price} = y_3,\ y_2 < y_3)\end{aligned}$$

Finally, the variable x_3 is identified as a contextual reference. After removing meta-constraints, this results in:

$$\begin{aligned}&\lambda\mathsf{apartment}\!:\!x_3\ \lambda S\ ?\mathsf{apartment}\!:\!x_2\\ &\quad (x_2.\mathit{street} = \mathit{Horn\_street}\ \wedge\ x_2 \in S\\ &\quad \wedge\ x_3.\mathit{street} = \mathit{Kings\_street}\ \wedge\ x_2.\mathit{price} < x_3.\mathit{price})\end{aligned}$$

6 Discussion

Given that the goal of this paper is to find an empirically based trade-off between robustness and expressiveness in spoken database dialogue, there are two questions that need to be answered:

1. Is the parsing algorithm robust enough?
2. Is the formalism expressive enough?

For an answer to the first question, we refer to Boye and Wirén (2003). Basically, that paper demonstrates that the parser is robust in the sense of outputting utterance descriptors with a significantly higher degree of accuracy than the strings output by the speech recognizer.

To answer the second question, we need to look at the kinds of utterances that *cannot* be represented by the formalism. To this end, we have studied two corpora with transcriptions of database dialogue. One is from the Adapt apartment-seeking domain (Bell et al. 2000), comprising 1 858 user utterances, and the other is from the SmartSpeak travel-planning domain (Boye et al. 1999), comprising 3 600 user utterances. Both corpora are the results of Wizard-of-Oz data collections used for development of the systems. In both cases the wizard tried to promote user initiative as well as to simulate near-perfect speech understanding.

Below we provide a list of utterance types that are not representable as utterance descriptors, but instances of which are found in at least one of the corpora. The list was obtained by manually checking several hundred utterances from each of the two corpora and, in addition, searching the entire corpora for a variety of constructions judged to be critical.

1. Constructions involving a function of more than one structured object: "How many two- or three-room apartments are there around here?"

2. Complex and–or nesting: "A large one-room apartment or a small two-room apartment."

3. Selection of elements from a complementary set: "Are there any other apartments around Medborgarplatsen that are about 50 square meters big and that are not situated at the ground floor?"

4. Comparatives involving implicit references where the comparison is made with a *set* of objects rather than with a single object. To illustrate, assume that several flight alternatives and their departure times have been up for discussion previously in the dialogue. The user then asks: "Is there a later flight?", requesting a flight which is later than all the

previously mentioned ones.[5]

The most common of these types is (1), which accounts for 0.4 % in the apartment corpus (but does not show up at all in the travel corpus). In none of the other cases do the number of occurrences exceed 0.05 % of a single corpus. We thus conclude that the kinds of utterances that are not representable by our semantic formalism only occur marginally in our corpora.

It is also interesting to consider utterance types that we cannot handle and that *don't* appear in our current corpora. For example, TEAM (Grosz et al. 1985) handles constructions such as "Is the smallest country the least populous" (comparison involving two superlatives) and "For the countries in North America, what are their capitals?" ("each" quantification). Although it may be argued that these particular sentences are not typical of spoken negotiative dialogue, session data from other spoken database interfaces are certainly useful for the purpose of testing the formalism.

We set out by claiming that negotiative dialogue requires that we go beyond flat slot–filler structures. Indeed, even a quick look at the corpora reveals that a substantial part of the utterances *do* require the added expressiveness. Thus, in addition to trivial specification utterances such as "I'd like a two-room apartment on the South Side", one encounters numerous instances like the following that can be represented by our formalism, but not in general by flat slot–filler structures:

1. Specifications such as "I'd like an apartment with a balcony" as opposed to seeking information about a particular aspect of an apartment, like "Does that apartment have a balcony?".

2. Comparative constructions involving explicit references to different objects in the same utterance, such as "I'd like an apartment at Horn street which is cheaper than the apartment at King's street.".

3. Comparatives involving implicit references, such as "Is there anything cheaper?".

4. Superlatives: "The cheapest apartment near Karlaplan."

5. Combinations of a comparative and selection of a minimal element: "When is the next flight?", which can be paraphrased as "Give me the earliest flight that departs after the flight that you just mentioned."

7 Future work

The current Adapt parser assumes certain contextual references to refer to a single object rather than a set of objects (e.g. see the representation of "I want a cheaper apartment" in section 4.3). Obviously, a more general representation would be

$$\lambda S_2\ \lambda S_1\ ?\mathsf{apartment}{:}x\ \forall\mathsf{apartment}{:}y\ (x.price < y.price\ \wedge\ x \in S_1\ \wedge\ y \in S_2)$$

("I want apartments from S_1 cheaper than all the apartments in S_2"). It would then be the task of the contextual reasoning component to infer the set S_2. In the same vein, a more general form of representing "How much do the apartments cost" would be

$$\lambda S\ ?\mathsf{money}{:}x\ \exists\mathsf{apartment}{:}y\ (y.price = x\ \wedge\ y \in S)$$

(i.e. "I want the prices of the apartments in S"). As is clear from these examples, such an extension, allowing the user to refer to sets of objects, would require the parsing algorithm to infer which variables are bound by universal quantifiers and which are bound by existential quantifiers[6].

This extension can be realized by introducing two new meta-constraints $all\_obj(\mathtt{t} : x)$ and $some\_obj(\mathtt{t} : x)$. The pattern "cheaper" would then yield:

$$head\_obj(\mathsf{apartment}{:}x_2),\ all\_obj(\mathsf{apartment}{:}x_3),\ x_2 \neq x_3,\ obj(\mathsf{money}{:}y_2),\ x_2.z = y_2,\ obj(\mathsf{money}{:}y_3),\ x_3.z = y_3,\ y_2 < y_3,\ ambiguous(z, \{price, monthly\_fee\}, default(price))$$

signalling that x_3 should be universally quantified. Similarly, "How big" would yield:

$$head\_obj(\mathsf{square\_meters}{:}y),\ some\_obj(\mathsf{apartment}{:}x), x.size = y$$

[5] To determine whether such a comparison is made with a set of objects or with a single object, it is in general not sufficient to look only at the last utterance. Thus, to handle this, the context-independent representation of the utterance must cater for both possibilities, thereby allowing the contextual analysis to make the final verdict (see further Section 7).

[6] The current approach avoids this issue by allowing references only to individual objects, in which case the difference between universal quantification and existential quantification disappears.

signalling that y should be existentially quantified.

Future work involves implementing and evaluating the need and robustness of this extension.

8 Related work

Approaches to handling spoken dialogue with databases can be largely divided into two types:

1. General-purpose linguistic rules and powerful semantic formalisms: not robust enough; overly expressive.

2. Pattern matching and flat slot–filler lists: robust but not expressive enough.

Several attempts at synthesizing these approaches have been made, either by "robustifying" (1) (for example, van Noord et al. 1999) or by extending (2) with the capability of handling general linguistic rules (Milward and Knight 2001). However, the semantic representations produced are still limited to that of flat slot–filler lists, and as a result they are not suitable for negotiative dialogue.

We have shown empirically that the subclass of questions handled by our approach is a useful one. In a similar vein, Popescu et al. (2003) define a class of "semantically tractable questions" to their PRECISE system. The idea is that each semantically tractable question provably yields a correct answer, whereas questions outside of the defined class are answered with an "error message" which allows the user to reformulate her question. Hence, the PRECISE system is "robust" in a conservative way, since it guarantees that no incorrect answer will ever be produced. On the other hand, PRECISE is text-based and has no dialogue capabilities, and thereby circumvents the problems introduced by speech-recognition errors and underspecified utterances.

9 Conclusion

The aim of this paper has been to develop a principled and empirically motivated approach to robust, negotiative spoken dialogue with databases. Key to our approach is the choice of semantic representation formalism. On the one hand, it is more expressive than today's commonly used, flat slot–filler lists that are limited to representing the propositional contents of utterances. On the other hand, it is still strongly restricted to make it compatible with the kind of robust parsing needed for spoken dialogue. While more empirical investigation is needed, experience with our current corpora indicates that the robustness–expressiveness tradeoff described here is a reasonable first approximation.

References

Alshawi, H. *The Core Language Engine*. The MIT Press, 1992.

Alshawi, H., Carter, D., Crouch, R., Pulman, S., Rayner, M. and Smith, A. CLARE — A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. Final report, SRI International, 1992.

Androutsopoulos, I., Ritchie, G. and Thanish, P. Natural Language Interfaces to Databases — An Introduction, *Journal of Natural Language Engineering* 1(1), pp. 29–85, 1995.

Bell, L., Boye, J., Gustafson J. and Wirén, M. Modality Convergence in a Multimodal Dialogue System. *Proc. Götalog*, pp. 29–34, 2000.

Bell, L., Boye, J. and Gustafson J. Real-time Handling of Fragmented Utterances. *Proc. NAACL Workshop on Adaptation in Dialogue Systems*, 2001.

Binot, J-L., Debille, L., Sedlock, D. and Vandecapelle D. Natural Language Interfaces: A New Philosophy. *SunExpert Magazine*, pp. 67–73, January 1991.

Boye, J., Wirén, M., Rayner, M., Lewin, I., Carter, D. and Becket, R. Language Processing Strategies and Mixed-Initiative Dialogues. In *Proc. IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 1999.

Boye, J. and Wirén, M. Robust Parsing of Utterances in Negotiative Dialogue. *Proc. Eurospeech*, 2003.

Grosz, B., Appelt, D., Martin, P. and Pereira, F. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. Technical note 356, SRI International, 1985.

Larsson, S. Issue-based Dialogue Management. Ph.D. Thesis, Göteborg University, ISBN 91-628-5301-5, 2002.

Milward, D. and Knight, S. Improving on Phrase Spotting for Spoken Dialogue Processing. *Proc. WISP*, 2001.

van Noord, G., Bouma, G., Koeling, R. and Nederhof, M-J. Robust Grammatical Analysis for Spoken Dialogue Systems. *Journal of Natural Language Engineering*, 5(1), pp. 45–93, 1999.

Popescu, A., Etzioni, O. and Kautz, H. Towards a Theory of Natural Language Interfaces to Databases. In *Proc. International Conference on Intelligent User Interfaces (IUI-2003)*, Miami, Florida, 2003.

Ullman, J. *Database and Knowledge-base Systems*, Volume I. Computer Science Press, 1988.