Cooperative answering and Inferential Erotetic Logic

Paweł Łupkowski

Chair of Logic and Cognitive Science
Institute of Psychology
Adam Mickiewicz University
Poznań, Poland
Pawel.Lupkowski@amu.edu.pl

Abstract

This paper addresses the issue of applicability of erotetic search scenarios, a tool developed within Inferential Erotetic Logic, in the area of cooperative answering for databases and information systems. Short descriptions of cooperative answering and Erotetic Search Scenarios are given. Some basic cooperative answering phenomena are modeled within the framework of Erotetic Search Scenarios.

1 Introduction

The issue of cooperative answering is important in the field of databases and information systems. Databases and information systems in general offer correct answers (as far as these systems contain valid data). The problem is to ensure that the answers will be also non-misleading and useful for a user.[1] To solve this problem certain specific techniques were developed. The most important are the following:

- consideration of specific information about a user's state of mind,
- evaluation of presuppositions of a query,
- detection and correction of misconceptions in a query (other than a false presupposition),
- formulation of intensional answers,
- generalization of queries and of responses.

A detailed description of the above techniques may be found in (Gaasterland et al., 1994) and (Godfrey, 1997). For their implementation in various database and information systems see e.g. (Godfrey et al., 1994), (Gal, 1988), (Benamara and Dizier, 2003b).

[1] H. P. Grice in (Grice, 1975) points out three features of what we may call a *cooperative answer*. It should be (i) correct, (ii) non-misleading, and (iii) useful answer to a query.

In this paper we will consider, among others, the following important aspect of cooperative answering: providing additional information useful for a user confronted with a failure. As Terry Gaasterland puts it:

> On asking a query, one assumes there are answers to the query too. If there are no answers, this deserves an explanation. (Gaasterland et al., 1994, p. 14)

We may consider a well known example here (cf. (Gal, 1988, p. 2)). Imagine that a student wants to evaluate a course before registering in it. He asks the secretary:

Q: *How many students failed course number CS400 last semester?*

The course CS400 was not given last semester, so the secretary would easily recognise the student's false assumption and correct it in her answer:

A_1: *Course number CS400 was not offered last semester.*

However, for most database interface systems the answer would be:

A_1^*: *None.*

Without additional explanations given, this response would be misleading for the student, and thus uncooperative from our perspective.

We will be addressing this issue, focussing our attention on the following cases: (a) the answer to a question is negative, (b) there is no answer available in a database, and (c) the asked question bares a misconception (i.e. it requests for information impossible to obtain from the database). We are going to make use of the Erotetic Search Scenarios framework, developed within A. Wiśniewski's Inferential Erotetic Logic

Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue

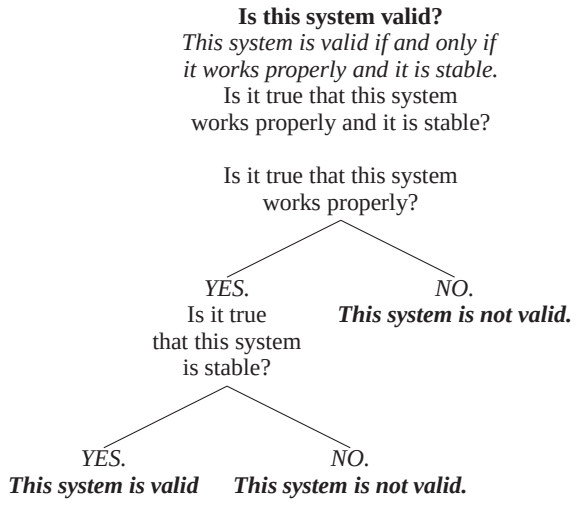
(cf. (Wiśniewski, 1995)). The reason for this choice is that Inferential Erotetic Logic provides the concept of validity of inferences which involve questions.[2]

For the reasons of space, only an informal characteristics of Erotetic Search Scenarios (thereafter referred to as *e-scenarios*) will be given here. The exact definition and many examples can be found in (Wiśniewski, 2001), (Wiśniewski, 2003) or (Wiśniewski, 2004).

E-scenarios are defined in terms of syntax and semantics. But:

> Viewed pragmatically, an e-scenario provides us with conditional instructions which tell us what questions should be asked and when they should be asked. Moreover, an e-scenario shows where to go if such-and-such a direct answer to a query appears to be acceptable and goes so with respect to any direct answer to each query. (Wiśniewski, 2003, p. 422)

For instance, let us imagine that we ask if a given system is valid and at the same time we construe the relevant concept of validity as follows: a system is valid just in case it works properly and is stable. How can one cope with this problem? A solution may be offered by an e-scenario. We can present this e-scenario as a downward tree with the main question as the root and direct answers to it as leaves. The relevant e-scenario for our exemplary problem is:



[2]For comparison of Inferential Erotetic Logic and J. Hintikka's Interrogative Model of Inquiry (with *interrogative game* as a central concept) see e.g. (Wiśniewski, 2004, p. 139–140).

The exemplary e-scenario, as well other e-scenarios, may be written in a formal language. Let us use here a language which we will call L_2; this language resembles a language characterised in (Wiśniewski, 2001, p. 20–21). The 'declarative' part of L_2 is a first-order language with identity and individual constants, but without function symbols. A *sentence* is a declarative well-formed formula (d-wff for short) with no occurrence of a free variable. The metalinguistic expression Ax refers to d-wffs of L_2 which have x as the only free variable. $A(x/c)$ designates the result of the substitution of an individual constant c for the variable x in Ax.

The vocabulary of the 'erotetic' part of L_2 consists of the signs: $?$, $\{$, $\}$, S, U, and the comma.

Questions of L_2 are expressions of the following forms:

(i) $?\{A_1, A_2, \ldots, A_n\}$
where $n > 1$ and $A_1, A_2, \ldots, A_n$ are syntactically distinct sentences of L_2,

(ii) $?\mathsf{S}(Ax)$
(iii) $?\mathsf{U}(Ax)$
where x is an individual variable and Ax is a d-wff of L_2 which has x as the only free variable.

Direct answers are defined as follows. For a question of the form (i), each of $A_1, A_2, \ldots, A_n$ is a direct answer to the question. For a question of the form (ii), a direct answer to it is a sentence of the form $A(x/c)$, where c is an individual constant. Direct answers to questions of the form (iii) fall under the schema:

$$A(x/c_1) \wedge \ldots \wedge A(x/c_n) \wedge \forall x(Ax \rightarrow x = c_1 \vee \ldots \vee x = c_n)$$

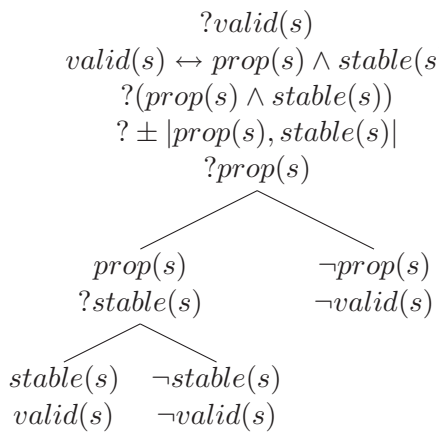
where $n \geq 1$ and $c_1, \ldots, c_n$ are distinct individual constants.

A question of the form (i) can be read, 'Is it the case that A_1, or is it the case that A_2, ..., or is it the case that A_n?'. A question of the form (ii) can be read, 'Which x is such that Ax?', whereas a question of the form (iii) can be read, 'What are all of the x's such that Ax?'.

For brevity, we will adopt a different notation for some types of questions. A question of the form $?\{A, \neg A\}$ ('Is it the case that A?') will be abbreviated as $?A$. The so-called (two-argument) *conjunctive questions*, namely $?\{A \wedge B, A \wedge \neg B, \neg A \wedge B, \neg A \wedge \neg B\}$ (to be read, 'Is

it the case that A and is it the case that B?'), will be abbreviated as $? \pm |A, B|$ — cf. (Wiśniewski, 2003, p. 399).

Here is the exemplary e-scenario written in the language L_2 ($valid$ stands for 'system is valid', $prop$ stands for 'system works properly', and $stable$ for 'system is stable'; the letter 's' is an individual constant, a name of the system):



As above, the e-scenario has a tree-like structure with the main question as the root and direct answers to it as leaves. Other questions are auxiliary. Either an auxiliary question has another question as the immediate successor (cf. e.g., '$?(prop(s) \wedge stable(s))$') or it has all the direct answers to it as the immediate successors (cf. e.g., '$?prop(s)$'). In the latter case the immediate successors represent the possible ways in which the relevant request for information can be satisfied, and the structure of the e-scenario shows what further information requests (if any) are to be satisfied in order to arrive at an answer to the main question. If an auxiliary question is a 'branching point' of an e-scenario, it is called a *query* of the e-scenario. Among auxiliary questions, only queries are to be asked; the remaining auxiliary questions serve as 'erotetic' premises only.

An e-scenario consists of *paths*, each of which leads from the main question through premises, auxiliary questions and answers to them, to a (direct) answer to the initial question. Viewed pragmatically, a path delivers some 'if/then' instructions. For instance, the instructions given by the leftmost path of our exemplary e-scenario are presented by Algorithm 1.

The key feature of e-scenarios is that auxiliary questions appear in them on the condition they are *erotetically implied* (in the sense of Inferential Erotetic Logic). Erotetic implication, Im, is a semantical relation between a question, Q, a (possibly empty) set of d-wffs, X, and a question, Q_1. Without going into details let us only say that Im ensures the following: (a) if Q has a true direct answer and X consists of truths, then Q_1 has a true direct answer as well ('transmission of soundness and truth into soundness'), and (b) each direct answer to Q_1, if true, and if all the X-es are true, narrows down the class at which a true direct answer to Q can be found ('open-minded cognitive usefulness'). For details see (Wiśniewski, 2003).

if *the main question is* $?valid(s)$ and *the initial premise is* $valid(s) \leftrightarrow prop(s) \wedge stable(s)$ **then**
  ask $?prop(s)$;

if *the answer is* $prop(s)$ **then**
  ask $?stable(s)$;

if *the answer is* $stable(s)$ **then**
  the answer to the main question is $valid(s)$;

Algorithm 1: Instructions given by the leftmost path of the exemplary e-scenario

Our exemplary e-scenario is based upon the following logical facts (A and B perform here the role of metalinguistic variables for sentences of L_2):

$\mathsf{Im}(?C, C \leftrightarrow A \wedge B, ?(A \wedge B))$
$\mathsf{Im}(?(A \wedge B), ? \pm |A, B|)$
$\mathsf{Im}(? \pm |A, B|, A)$
$\mathsf{Im}(? \pm |A, B|, B)$

2 Erotetic Search Scenarios and basic cooperative answering behaviours

In this section we will consider a very simple ('toy') example of a database. The database will be in a deductive database form. This exemplary database (thereafter we will refer to it as DB) will serve as a testing field for e-scenarios applicability in the domain of cooperative answering.

Each deductive database consists of:

- Extensional database ($\mathcal{EDB}$) — build out of facts,
- Intensional database ($\mathcal{IDB}$) — build out of rules,
- Integrity constraints ($\mathcal{IC}$).

For simplicity and notation coherence, we do not use the Datalog notation usually applied in similar contexts. Instead, we will be using the language L_2 described in the previous section.

In our case $\mathcal{EDB}$ consists of the following facts (where usr stands for 'is a user' and $live$ stands for 'lives in'):

$usr(a)$ $\quad$ $live(a,p)$
$usr(b)$ $\quad$ $live(b,zg)$
$usr(c)$ $\quad$ $live(c,p)$
$usr(d)$

The $\mathcal{IDB}$ contains the following rules (where loc_usr means 'is a local user'):

$$loc\_usr(x) \rightarrow usr(x)$$

$$loc\_usr(x) \rightarrow live(x,p)$$

$$usr(x) \wedge live(x,p) \rightarrow loc\_usr(x)$$

As for the $\mathcal{IC}$, there is only one, saying that it is not possible to live in zg and p at the same time.

$$\neg(\exists x(live(x,zg) \wedge live(x,p)))$$

Questions carried out against the DB may be polar questions asking if objects have some properties. For example, one can ask if object a_i is a 'user'; this is expressed by '$?usr(a_i)$'. A direct answer is either 'yes' or 'no' (expressed by $usr(a_i)$ and $\neg usr(a_i)$, respectively). But we may as well ask about an example of an object satisfying the condition 'is a user'. This question is expressed in L_2 by '$?\mathsf{S}(usr(x))$'. We can also ask of all the objects satisfying the condition 'is a user', by means of '$?\mathsf{U}(usr(x))$'. Then an answer is supposed to list all the objects in the database having the property of being a user (e.g. '$usr(a) \wedge usr(b)$').

E-scenarios are applied by a layer located between a user and the DB (let us call it a 'cooperative layer'). The layer proceeds a question asked by a user by carrying out the relevant auxiliary questions/queries against the DB in a way determined by an e-scenario. The received answers to queries are then transformed into an answer to the main question. The scheme of such a system is presented in Figure 1.

For the purposes of this paper we need to choose some e-scenarios that might be used to work with the exemplary DB. Most of them will fall under a certain general schema; in designing the schema we rely on the following logical facts:

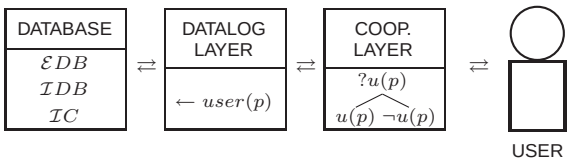
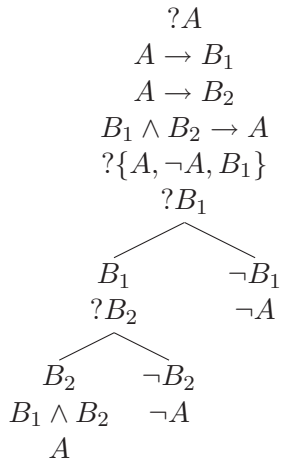


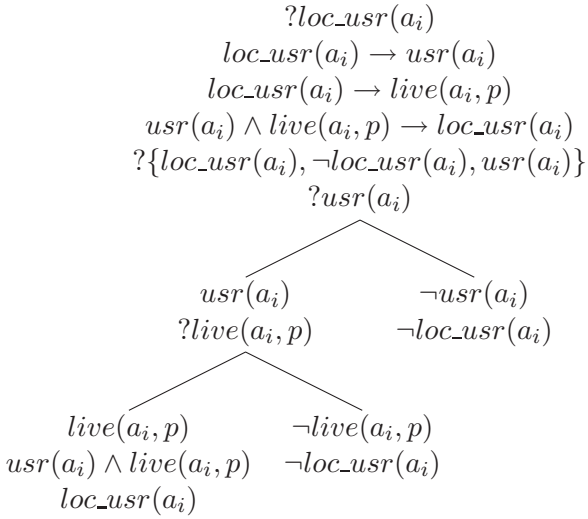
Figure 1: Scheme of the cooperative database system using e-scenarios

$\mathsf{Im}(?A, C \rightarrow A, ?\{A, \neg A, C\})$
$\mathsf{Im}(?\{A, \neg A, C\}, ?C)$
$\mathsf{Im}(?A, B_1 \wedge B_2 \rightarrow A, B_1, A \rightarrow B_2, ?B_2)$

Here is the schema:



We may adapt the above schema to our specific needs, obtaining e-scenarios for the concepts that occur in DB. For instance, the relevant e-scenarios for questions of the form 'Is a_i a local user?' fall under the schema (we will refer to it as ESS1).



Note that the $\mathcal{IDB}$ rules are used as premises of the e-scenario.

ESS1 should be regarded as a plan of questioning for the main question. It tells us which sub-

sequent questions should be executed against the DB and in which order/when it should be done.

Let let us see how ESS1 may be executed against the DB.

First, we consider an example of a question whose answer in view of the $\mathcal{IDB}$ is affirmative. This will help us to understand how ESS1 is executed against the DB. The question is, 'Is a is a local user?'. The question is executed against $\mathcal{IDB}$ as follows:

Q1: Is a a local user?

$$?loc\_usr(a)$$
$$loc\_usr(a) \rightarrow usr(a)$$
$$loc\_usr(a) \rightarrow live(a,p)$$
$$usr(a) \wedge live(a,p) \rightarrow loc\_usr(a)$$
$$?\{loc\_usr(a), \neg loc\_usr(a), usr(a)\}$$
$$?usr(a)$$
$$usr(a)$$
$$?live(a,p)$$
$$live(a,p)$$
$$usr(a) \wedge live(a,p)$$
$$loc\_usr(a)$$

answer: a is a local user ($loc\_usr(a)$),
since:
$loc\_usr(b) \rightarrow usr(b)$,
$loc\_usr \rightarrow live(b,p)$,
$usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$
and we know that:
a is a user **and** a lives in p

The diagram shows that only one path of ESS1 has been 'activated' or 'actualized' in order to get the answer (here the leftmost path). As long as a successful execution of an e-scenario is concerned, this is always the case. To indicate the unrealized questioning options we left the corresponding branches empty. For **Q1** the process of ESS1 execution against the DB boils down, in essence, to carrying out two subsequent queries, namely '$?usr(a)$', and (after receiving the affirmative answer '$usr(a)$'), the query '$?live(a,p)$'. The answers obtained entail the affirmative answer to the main question, which states that a is a local user, '$loc\_usr(a)$'. This answer is then provided (with additional explanations) to the user. Needles to say, the answer received can be regarded as cooperative.

Now we shall turn to other questions, chosen in such a manner that some more complex cooperative behaviors will be needed. First, let us consider the following question, 'Is b a local user?'.

Q2: Is b a local user?

$$?loc\_usr(b)$$
$$loc\_usr(b) \rightarrow usr(b)$$
$$loc\_usr(b) \rightarrow live(b,p)$$
$$usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$$
$$?\{loc\_usr(b), \neg loc\_usr(b), usr(b)\}$$
$$?usr(b)$$
$$usr(b)$$
$$?live(b,p)$$
$$\neg live(b,p)$$
$$\neg loc\_usr(b)$$

After ESS1 execution against the DB, one gets the negative answer to the main question. However, negative answers to polar questions are often less expected than affirmative ones; in some cases a negative answer can even be regarded as a failure. But we can easily supplement a negative answer received with an *explanation*. We do it by making use of the path just executed and the premises involved. Here is an example of an explanation:

answer: b is not a local user,
since: $loc\_usr(b) \rightarrow usr(b)$,
$loc\_usr \rightarrow live(b,p)$,
$usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$
and we know that: b does not live in p
$?live(b,p)$: $\neg live(b,p)$

The explanation contains information about the initial premises of the e-scenario (which reflect $\mathcal{IDB}$ part of the DB) and confront them with gained pieces of information. What is more it points out the query that failed, so a user knows exactly what information was not obtained from the DB.

As the example shows, e-scenarios allow to generate explanations of this kind in a quite easy way. The relevant procedure can be briefly described as follows.

We produce a list on the basis of the e-scenario's part just activated. We enumerate elements of the list consecutively; as a result we obtain an index, i.e. a sequence of indices.

1. $?loc\_usr(b)$
2. $loc\_usr(b) \rightarrow usr(b)$
3. $loc\_usr(b) \rightarrow live(b,p)$
4. $usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$

5. $?\{loc\_usr(b), \neg loc\_usr(b), usr(b)\}$
6. $?usr(b)$
7. $usr(b)$
8. $?live(b,p)$
9. $\neg live(b,p)$
10. $\neg loc\_usr(b)$

By means of the list we can identify the main question and the initial premises. The main question will be a formula of the form ?_ (i.e. formula beginning with the question mark ?') with index number 1. Then we identify a formula of the form ?_ that has the lowest index number greater than 1. Let the index number be k. All formulas with index numbers larger than 1 and lower than k are the initial premises; we write them down consecutively.

2. $loc\_usr(b) \rightarrow usr(b)$
3. $loc\_usr(b) \rightarrow live(b,p)$
4. $usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$

The task of finding the next remaining element of the explanation reduces to the issue of finding, on the list, a formula of the form ?_ with the largest index number. In this way the last 'active' query is identified. Then the query and the next element of the list (i.e. direct answer to this query) will be used in the explanation.

8. $?live(b,p)$
9. $\neg live(b,p)$

A formal description of the procedure is presented as Algorithm 2.

Let us now consider another example. By the way, the example shows how e-scenarios can decrease the number of queries executed agains the DB. The question is, 'Is e a local user?'

Q3: Is e a local user?

$?loc\_usr(e)$
$loc\_usr(e) \rightarrow usr(e)$
$loc\_usr(e) \rightarrow live(e,p)$
$usr(e) \wedge live(e,p) \rightarrow loc\_usr(e)$
$?\{loc\_usr(e), \neg loc\_usr(e), usr(e)\}$
$?usr(e)$

$\neg usr(e)$
$\neg loc\_usr(e)$

answer: e is not a local user ($\neg loc\_usr(e)$), **since:**

Data: E-scenario path as a list with index (we will denote element of the list as e and element of the index as i)
Result: Additional explanations for the answer to the initial question

1 $iq \leftarrow e$ with $i = 1$
2 $answ\_iq \leftarrow e$ with max i
3 $next\_q \leftarrow e$ of the form ?_ with min $i > 1$
4 i_{next_q} = i of $next\_q$
5 $premises \leftarrow e$ with $1 < i < i_{next\_q}$
6 $failing\_q \leftarrow e$ of the form ?_ with max i
7 $i_{failing\_q}$ = i of $failing\_q$
8 $answer\_failing\_q \leftarrow e$ with $i_{failing\_q} + 1$

Algorithm 2: Generation of additional explanations for the answer to the initial question

$loc\_usr(b) \rightarrow usr(b)$,
$loc\_usr \rightarrow live(b,p)$,
$usr(b) \wedge live(b,p) \rightarrow loc\_usr(b)$
and we know that: e is not a user
$?usr(e)$: $\neg usr(e)$

The next example illustrates how one can cope with a misconception of a question asked by a user. The question is: Does b live in zg and is a local user? We apply an e-scenario of a slightly different form than ESS1[3], i.e.

$?(live(a_i, zg) \wedge loc\_usr(a_i)$
$? \pm |live(a_i, zg), loc\_usr(a_i)|$
$?loc\_usr(a_i)$
$loc\_usr(a_i) \rightarrow usr(a_i)$
$loc\_usr(a_i) \rightarrow live(a_i, p)$
$usr(a_i) \wedge live(a_i, p) \rightarrow loc\_usr(a_i)$
$?\{loc\_usr(a_i), \neg loc\_usr(a_i), usr(a_i)\}$
$?usr(a_i)$

$usr(a_i)$ | $?live(a_i, p)$
$\neg usr(a_i)$ | $\neg loc\_usr(a_i)$ | $\neg(live(a_i, zg) \wedge loc\_usr(a_i))$

$live(a_i, p)$ | $usr(a_i) \wedge live(a_i, p)$ | $loc\_usr(a_i)$ | $?live(a_i, zg)$
$\neg live(a_i, p)$ | $\neg loc\_usr(a_i)$ | $\neg(live(a_i, zg) \wedge loc\_usr(a_i))$

$live(a_i, zg)$ | $\neg live(a_i, zg)$

[3] An additional logical fact is used here, namely: $\mathsf{Im}(?(A \wedge D), A, ?D)$.

Question **Q4** is executed against to DB as follows:

Q4: Does b live in zg and is a local user?

$$?(live(b, zg) \wedge loc\_usr(b)$$
$$? \pm |live(b, zg), loc\_usr(b)|$$
$$?loc\_usr(b)$$
$$loc\_usr(b) \rightarrow usr(b)$$
$$loc\_usr(b) \rightarrow live(b, p)$$
$$usr(b) \wedge live(b, p) \rightarrow loc\_usr(b)$$
$$?\{loc\_usr(b), \neg loc\_usr(b), usr(b)\}$$
$$?usr(b)$$

$$usr(b)$$
$$?live(b, p)$$

$$\neg live(b, p)$$
$$\neg loc\_usr(b)$$
$$\neg(live(b, zg) \wedge loc\_usr(b))$$

A simple negative answer to the initial question will not make a user aware of the misconception in the question. But when explanations are added, a user can learn about the database schema and understand better the obtained answer.[4]

answer: no ($\neg(live(b, zg) \wedge loc\_usr(b))$)
since: b is not a local user ($\neg loc\_usr(b)$)
this is due to:
$loc\_usr(b) \rightarrow usr(b)$,
$loc\_usr \rightarrow live(b, p)$,
$usr(b) \wedge live(b, p) \rightarrow loc\_usr(b)$
and: $\neg live(b, p)$
(*correction of user's misconception about the* DB *schema*)
but we know that: $usr(b)$

Yet another example shows one of the possible ways of dealing with information gaps in the database. We ask if d is a local user. During ESS1 execution against the DB a query fails since the requested information is not present in the database (this is indicated by the ⊠ symbol on the schema below). From this point, a further execution of ESS1 is no longer possible. However the executed part of the e-scenario enables us to generate a sensible explanation of this fact and to report the gained information relevant to the main question. Let us stress that the information gained in the process of ESS1 execution will always stay in connection with the main question. Consequently, we may simply report the obtained answers to queries to a user as a piece of information relevant to his/her question.

Q5: Is d a local user?

$$?loc\_usr(d)$$
$$loc\_usr(d) \rightarrow usr(d)$$
$$loc\_usr(d) \rightarrow live(d, p)$$
$$usr(d) \wedge live(d, p) \rightarrow loc\_usr(d)$$
$$?\{loc\_usr(d), \neg loc\_usr(d), usr(d)\}$$
$$?usr(d)$$

$$usr(d)$$
$$?live(d, p)$$

⊠ ⊠

answer: the answer is unknown, since:
$loc\_usr(b) \rightarrow usr(b)$,
$loc\_usr \rightarrow live(b, p)$,
$usr(b) \wedge live(b, p) \rightarrow loc\_usr(b)$
and: query $?live(d, p)$ failed
but we know that: d is a user

3 Summary and further works

I have presented here only some simple examples of cooperative answering behaviours that can modelled by means of the e-scenarios framework. But, in my opinion, Inferential Erotetic Logic provides many useful tools for investigating the area of cooperative answering. Erotetic Search Scenarios framework presented in this paper allows to join two focus points of cooperative answering research: question analysis and fundamental reasoning procedures — cf. (Benamara and Dizier, 2003b, p. 63). It also allows to develop techniques which are domain unspecific (in contrast to limited domains systems like the WEBCOOP developed by Benamara and Dizier (2003a), (2003b)).

Future works will concentrate mainly on incorporating techniques developed in (Gal, 1988) (based on integrity constraints processing) into presented framework. Also new algorithmic procedures working on e-scenarios that enable implementations of specific cooperative techniques will be developed. The area of automatic generation of premises for e-scenarios using Formal Concept Analysis will also be explored.

[4] At this stage of this research integrity constraints are not employed in the process of generating cooperative answers using e-scenarios. However concerning techniques and results presented in (Gal, 1988) it would be necessary to incorporate $\mathcal{IC}$ into e-scenarios' premises to obtain better cooperative behaviours (especially when users' misconceptions are considered).

References

Farah Benamara and Patrick Saint Dizier. 2003a. Dynamic Generation of Cooperative Natural Language Responses. Ninth European workshop on Natural Language Generation, Budapest, Hungary.

Farah Benamara and Patrick Saint Dizier. 2003b. WEBCOOP: a cooperative question-answering system on the web. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, 63–66.

Terry Gaasterland, Parke Godfrey, and Jack Minker. 1994. An Overview of Cooperative Answering. In R. Demolombe and T. Imielinski, editors, *Nonstandard Queries and Nonstandard Answers*, 1–40.

Annie Gal. 1988. Cooperative Responses in Deductive Databases. PhD Thesis, University of Maryland, Department of Computer Science.

Parke Godfrey, Jack Minker, and Lev Novik. 1994. An Architecture for a Cooperative Database System. In W. Litwin and T. Risch, editors, *Proceedings of the First International Conference on Applications of Databases.* Lecture Notes in Computer Sicence, 819: 3–24.

Parke Godfrey. 1997. Minimization in Cooperative Response to Failing Database Queries. *International Journal of Cooperative Information Systems*, 6(2): 95–149.

Herbert P. Grice. 1975. Logic and Conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*, 41–58, New York: Academic Press.

Mariusz Urbański. 2001. Synthetic Tableaux and Erotetic Search Scenarios: Extension and Extraction. *Logique & Analyse*, No. 173–174–175: 69–91.

Andrzej Wiśniewski. 1995. *The Posing of Questions: Logical Foundations of Erotetic Inferences,* Kluwer AP, Dordrecht, Boston, London.

Andrzej Wiśniewski. 2001. Questions and Inferences. *Logique & Analyse*, No. 173–175: 5–43.

Andrzej Wiśniewski. 2003. Erotetic Search Scenarios. *Synthese*, No. 134: 389–427.

Andrzej Wiśniewski. 2004. Erotetic Search Scenarios, Problem-Solving, and Deduction. *Logique & Analyse*, No. 185–188: 139–166.