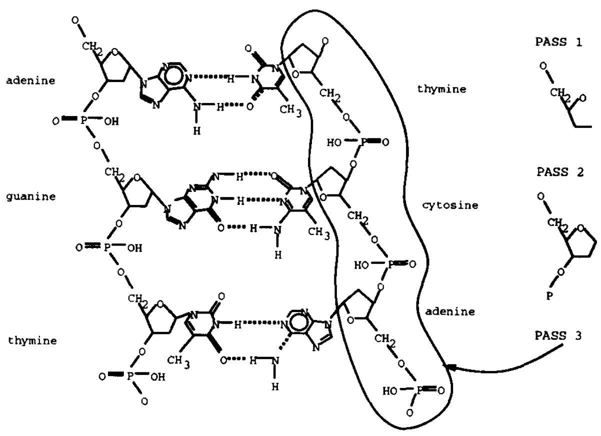

Fig. 2. Portion of DNA molecule and discovered substructure.

instance to nodes inside the instance now connect to the new node. Edges internal to the instance are removed. The program is then run a second time, with heavier weight given to substructures which utilize the previously discovered substructure. The increased weight reflects increased attention to this substructure. Fig. 2 shows the results after each pass. Note that on the third pass, SUBDUE linked together the instances of the substructure in the second pass to find the chains of the double helix. Results indicate that SUBDUE can discover pertinent substructures and find a hierarchical description of the input data by replacing previously discovered substructures on successive passes.

VI. CONCLUSIONS

Automated knowledge discovery is essential for extracting information from databases [2]. Extracting knowledge from structural databases requires the identification of repetitive substructures in the data. The previous examples show how SUBDUE's heuristic search and inexact graph match can discover interesting and repetitive substructures in real structural domains. Applying SUBDUE to scene analysis assists in compression of the image and identification of similar objects in the scene. Application to chemical analysis assists the discovery of previously unknown molecules and cognitive compression of the compound by abstracting over newly discovered molecules. Further experimentation is underway in both artificial domains and other real domains in order to determine the effects of parameters and reduce the computational requirements of SUBDUE's substructure discovery algorithm.

REFERENCES

[1] H. Bunke and G. Allerman, "Inexact graph matching for structural pattern recognition," *Pattern Recognition Lett.*, vol. 1, no. 4, pp. 245-253, 1983.
[2] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge discovery in databases: An overview," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley, Eds. Cambridge, MA: MIT, 1991, pp. 1-27.
[3] L. B. Holder, "Empirical substructure discovery," in *Proc. 6th Int. Workshop Machine Learning*, 1989, pp. 133-136.
[4] L. B. Holder, D. J. Cook, and H. Bunke, "Fuzzy substructure discovery," in *Proc. 9th Int. Conf. Machine Learning*, 1992, pp. 218-223.
[5] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*. New York: McGraw-Hill, 1975.

Avoiding Misconstruals in Database Systems: A Default Logic Approach

Andrea S. Hemerly, Marco A. Casanova, and Antonio L. Furtado

***Abstract*—This paper describes a cooperative interface that, using suitable user models, alters the processing of the user's queries to include additional information that will block faulty inferences. In a sense, the interface actively teaches the user facts about the database that he did not explicitly asked for. User interaction with the database then becomes a learning and discovery process guided by the queries he poses to the interface. The paper also introduces a semantics for user models that captures, with the help of default logic, the nonmonotonic behavior users normally exhibit. Finally, the paper contains results showing that the cooperative interface generates enough additional information to block all faulty inferences.**

***Index Terms*—Cooperative user interfaces, database systems, default logic, user modeling.**

I. INTRODUCTION

When interacting with a database, a user is typically tempted to infer further information from that explicitly obtained from previous queries. However, since his world model is often faulty or incomplete, a fact he infers may be false with respect to the database. Such facts are often called "misconstruals" in the literature. For example, a client of a video tape renting shop, after consulting the tape database and verifying that a certain tape has not been rented, may inadvertently infer that the tape is available, when it has actually been reserved. A more cooperative database would respond that the tape is in the store, but it is not available because it has been reserved.

To address the problem of false inferences, we propose a *cooperative interface* that offers additional information to the user whenever he is about to infer information that contradicts that stored in the database. The interface essentially simulates user inferences and compares the result with what can be obtained from the database. We assume that users just query the database (that is, no updates are allowed).

The rest of this short paper briefly describes the cooperative interface and its environment. The reader is referred to [4] and [5] for the basic concepts of logic programming and default logic, respectively, and to [2] for the full formal description of the concepts here introduced.

II. THE ENVIRONMENT OF THE COOPERATIVE INTERFACE

In this section, we adapt—and simplify—the environment for cooperative interfaces introduced in [1] by proposing a new semantics for user models, defined with the help of default logic.

The environment of the cooperative interface consists of a deductive database, a user model, a log, and a cooperativeness domain. A single user model is considered, for simplicity, and is defined as a default theory with special characteristics. The log

Manuscript received October 1, 1992: revised June 15, 1993.
A. S. Hemerly and M. A. Casanova are with the Centro Científico Rio, IBM Brasil, Av. Presidente Vargas 824/844 Centro, 20071 Rio de Janeiro RJ, Brasil.
A. L. Furtado is with the Departamento de Informática, Pontífica Universidade Católica do RJ, R. Marquês de São Vicente 225 Gávea, 22453 Rio de Janeiro RJ, Brasil.
IEEE Log Number 9212803.

contains all answers the user obtained during his dialog with the database and it also indicates that certain facts hold (or do not hold) in the database. The cooperativeness domain defines the class of user inferences the interface has to simulate.

More precisely, a *deductive database* is a stratified normal program $\mathfrak{D}$ containing only allowed clauses. A *query* over $\mathfrak{D}$ is an allowed conjunction Q. A *user model for* $\mathfrak{D}$ is a stratified normal program $\mathfrak{U}$ containing only allowed clauses. A *log* λ is a set of literals of the form s_L and f_L, where L is a simple ground positive literal.

The *cooperativeness domain* $\boldsymbol{d}$ is always fixed as the set of all inferences of conjunctions of positive facts (this concept will be made more precise at the beginning of Section III). We also consider that every fact derivable solely from the user model is also derivable from the database. These decisions follow from what we call the *log initialization problem*. That is, if we allowed the user to make false inferences even before starting the dialog with the database, we would have to initialize the log to block such inferences even before the user submitted his first query.

We now present an interpretation, based on default theories, for normal programs in the presence of logs. We thus establish an interpretation for user models, and also for deductive databases, if we ignore the log in the following definitions. The interpretation captures the idea that the user can infer both positive and negative facts (through the closed world assumption), and that, when receiving an answer from the database that contradicts a fact he has previously derived, the user is forced to revise his beliefs by dropping false conclusions.

Let Λ be a fixed first-order alphabet. The *extended alphabet corresponding to* Λ, denoted by $\overline{\Lambda}$, is obtained by adding to Λ the predicate symbols $\bar{p}$, s_p, and f_p, with arity n, for each predicate symbol p of arity n in Λ. If L is a literal of the form $P(t)$, we denote $\bar{p}(t)$, $s\_p(t)$ and $f\_p(t)$ by $\overline{L}$, s_L and f_L, respectively. The *companion set of defaults* of Λ, denoted by D, is the set of nonnormal closed defaults $: \neg\overline{L}/\neg L$ and $\overline{L}: \neg f\_L/L$, for every ground literal L over Λ.

Let P be a normal program. The *barred transform* of P, denoted by $\overline{P}$, is defined as: i) if $A \leftarrow L_1, \cdots, L_m$ is a clause in P, then $\overline{A} \leftarrow L_1, \cdots, L_m$ is a clause in $\overline{P}$; ii) $(\neg L \leftarrow f\_L)$ and $(\overline{L} \leftarrow s\_L)$ are clauses in $\overline{P}$, for all literals L of the form $p(x_1, \cdots, x_n)$, where p is a predicate symbol of arity n in Λ.

Let λ be a log. The *default theory associated with P and* λ, denoted by Δ_P^λ, is the pair (D, P_λ), where $P_\lambda = \overline{P} \cup \lambda$. The *initial default theory associated with P*, denoted by Δ_P^0, is the default theory associated with P and λ, for $\lambda = \emptyset$.

Let Q be a conjunction of literals. An answer substitution θ for Q from P is *correct* iff all extensions of Δ_P^0 contain $\forall Q\theta$.

We denote by $\Delta_{\mathfrak{D}}$ the initial default theory associated with $\mathfrak{D}$. We can prove that, since $\mathfrak{D}$ is stratified, $\Delta_{\mathfrak{D}}$ has only one extension.

As an example, let $\Lambda = \{$*available, present, reserved, macunaima*$\}$, where *available*, *present* and *reserved* are unary predicate symbols and *macunaima* is a constant. Let $\mathfrak{U}$ be a user model that contains only one clause:

$\mathfrak{U}$.1. *available* $(x) \leftarrow$ *present*(x), $\neg$*reserved* (x)

Let $\mathfrak{D}$ be the following deductive database:

$\mathfrak{D}$.1. *available* $(x) \leftarrow$ *present*(x), $\neg$*reserved*(x)

$\mathfrak{D}$.2. *present*(*macunaima*)

$\mathfrak{D}$.3. *reserved*(*macunaima*)

Assume that the log λ is empty. The initial default theories associated with $\mathfrak{U}$ and $\mathfrak{D}$ are $\Delta_{\mathfrak{U}}^0 = (D, \overline{\mathfrak{U}})$ and $\Delta_{\mathfrak{D}} = (D, \overline{\mathfrak{D}})$, where (representing the symbols just by their first letter):

$$\overline{\mathfrak{D}} = T \cup \{\bar{p}(m), \bar{r}(m), \bar{a}(x) \leftarrow p(x), \neg r(x)\}$$

$$\overline{\mathfrak{U}} = T \cup \{\bar{a}(x) \leftarrow p(x), \neg r(x)\}$$

$$T = \{\bar{a}(x) \leftarrow s\_a(x), \bar{p}(x) \leftarrow s\_p(x), \bar{r}(x) \leftarrow s\_r(x),$$
$$\neg a(x) \leftarrow f\_a(x), \neg p(x) \leftarrow f\_p(x), \neg r(x) \leftarrow f\_r(x)\}$$

$$D = \left\{ \frac{: \neg\bar{p}(m)}{\neg p(m)}, \frac{\bar{p}(m): \neg f\_p(m)}{p(m)}, \frac{: \neg\bar{a}(m)}{\neg a(m)}, \right.$$
$$\left. \frac{\bar{a}(m): \neg f\_a(m)}{a(m)}, \frac{: \neg\bar{r}(m)}{\neg r(m)}, \frac{\bar{r}(m): \neg f\_r(m)}{r(m)} \right\}.$$

The unique extension of $\Delta_{\mathfrak{D}}$ is $\mathcal{E} = Th(\overline{\mathfrak{D}} \cup \{p(m), r(m), \neg a(m)\})$, which states that tape Macunaíma is not available. Note that any positive literal that can be inferred from $\Delta_{\mathfrak{U}}^0$ is in $\mathcal{E}$.

To conclude this section, we observe that the concept of default proof defined in [5] cannot be used here because it applies only to normal default theories. However, we refer the reader to [2], where we introduced the concept of *generic default proof*, that applies to generic default theories, and proved soundness and completeness results, similar to those found in Reiter's original paper. We also extended the results and proved soundness and completeness results with respect to answer computation, in the usual sense of logic programming.

III. The Cooperative Interface

Let $\mathfrak{D}$ be a deductive database and $\mathcal{E}$ be the unique extension of $\Delta_{\mathfrak{D}}$. Given an inference from a user model and a log, the interface essentially tests if the inference does not contradict $\mathfrak{D}$. This is formalized by the concept of *the certification test for* $\mathfrak{D}$, which is a Boolean function $\varphi_{\mathfrak{D}}$ defined as follows.

We first observe that, in the context of defaults, the cooperativeness domain $\boldsymbol{d}$ becomes the set of all generic default proofs of conjunctions of positive facts.

Given any user model $\mathfrak{U}$ and any generic default proof $\mathfrak{p}$ from $\Delta_{\mathfrak{U}}$ in $\boldsymbol{d}$, we define that $\varphi_{\mathfrak{D}}(\mathfrak{p}) =$ *false* iff there is a default δ used in $\mathfrak{p}$ such that, if δ is of the form $: \neg\overline{L}/\neg L$, then $L \in \mathcal{E}$, and if δ is of the form $\overline{L}: \neg f\_L/L$ then $\neg L \in \mathcal{E}$. If $\varphi_{\mathfrak{D}}(\mathfrak{p}) =$ *false*, we say that $\mathfrak{p}$ is *uncertified*; otherwise, we say that $\mathfrak{p}$ is *certified*.

If $A_1, \cdots, A_p$ are positive literals and $\neg B_1, \cdots, \neg B_q$ are negative literals in a query Q, we define *expand*$(Q) = \{s\_A_1, \cdots, s\_A_p, f\_B_1, \cdots, f\_B_q\}$. We now present a high-level description of the cooperative interface:

```
input:       a query $Q$
output:      an answer $\theta$ for $Q$ or fail
parameters:  a deductive database $\mathfrak{D}$, a user model $\mathfrak{U}$, a log $\lambda$
               and the cooperativeness domain $d$

1) Find an answer substitution $\theta$ of $Q$ from $\Delta_{\mathfrak{D}}$ such that $Q\theta \in \mathcal{E}$,
   where $\mathcal{E}$ is the unique extension of $\Delta_{\mathfrak{D}}$; if it fails, return fail and stop.
2. Add all literals in expand($Q\theta$) to $\lambda$.
3. While $\lambda$ changes, do:
   a) Simulate all generic default proofs $\mathfrak{p}$ of $A \in \{A \mid \overline{A} \leftarrow B_1, \cdots, B_m \in \mathfrak{U}\}$ from $\Delta_{\mathfrak{U}}$.
   b) If $\varphi_{\mathfrak{D}}(\mathfrak{p}) = false$ and $\delta$ is the last $\varphi_{\mathfrak{D}}(\mathfrak{p})$-failure default of $\mathfrak{p}$, then:
      i) if $\delta$ is of the form $: \neg\overline{L}/\neg L$, then add $s\_L$ to $\lambda$;
      ii) if $\delta$ is of the form $\overline{L}: \neg f\_L/L$, then add $f\_L$ to $\lambda$.
4. Return $\theta$.
```

During a session, the user may pose several queries to the system, which are then passed to the cooperative interface. Consider that the log is initially empty ($\lambda_0 = \emptyset$). The behavior of the system induces a *cooperative dialog* C, defined by a sequence of triples $(Q_1, \theta_1, \lambda_1), \cdots, (Q_n, \theta_n, \lambda_n)$, where for each $i \in [1, n]$, Q_i is a query, θ_i is an answer substitution for Q or the expression *fail*, and λ_i is a log. Note that the cooperative dialog induces a sequence of default theories $\Delta^0_{\mathcal{U}}, \cdots, \Delta^n_{\mathcal{U}}$, where $\Delta^i_{\mathcal{U}}$ is the default theory associated with $\mathcal{U}$ and λ_i, for $i \in [0, n]$.

As a simple example of the behavior of the interface, let $\mathfrak{D}$, $\mathcal{U}$, $\mathcal{E}$, $\overline{\mathcal{U}}$, and T be as in the example of Section II and again represent the symbols by their first letter. Assume that the log is initially empty. Suppose that the user starts the dialog with the query $Q = p(x)$. The interface computes the answer substitution $\theta = \{x/m\}$ and adds *expand*$(Q\theta)$ to the log, which now becomes $\lambda' = \{s\_p(m)\}$. The default theory associated with $\mathcal{U}$ is $\Delta^{\lambda'}_{\mathcal{U}} = (D, \mathcal{U}_{\lambda'})$, where $\mathcal{U}_{\lambda'} = \overline{\mathcal{U}} \cup \lambda'$.

At this point, the user is able to infer $a(m)$. The interface then proceeds to simulate default proofs. In particular, it simulates the following generic default proof of $a(x)$ from $\Delta^{\lambda'}_{\mathcal{U}}$:

$$0.1 \leftarrow a(x) \qquad D_0 = \left\{\frac{\bar{a}(m): \neg f\_a(m)}{a(m)}\right\}$$

$$0.2 \ \Box$$

$$1.1 \leftarrow \bar{a}(m) \qquad D_1 = \left\{\frac{\bar{p}(m): \neg f\_p(m)}{p(m)}, \ \frac{: \neg \bar{r}(m)}{\neg r(m)}\right\}$$

$$1.2 \leftarrow p(m), \ \neg r(m)$$

$$1.3 \leftarrow \neg r(m)$$

$$1.4 \ \Box$$

$$2.1 \leftarrow \bar{p}(m) \qquad D_2 = \emptyset$$

$$2.2 \leftarrow s\_p(m)$$

$$2.3 \ \Box$$

It can be shown that the sequence $\mathfrak{p} = (D_0, D_1, D_2)$ is a generic default proof of $a(x)$, in the sense of [2]. Observe that the clauses numbered "0.x" refute the negation of $a(x)$, those numbered "1.x" refute the conjunction of the preconditions of the defaults in D_0, whereas those numbered "2.x" do the same for the defaults in D_1.

However, since $r(m) \in \mathcal{E}$, $\mathfrak{p}$ is not certified with respect to $\mathfrak{D}$ and $: \neg \bar{r}(m) / \neg r(m)$ is the $\varphi_{\mathfrak{D}}(\mathfrak{p})$-failure default. Then, the interface includes the literal $s\_r(m)$ in λ', which then becomes $\lambda'' = \{s\_p(m), s\_r(m)\}$. The default theory associated with $\mathcal{U}$ and λ'' is $\Delta^{\lambda''}_{\mathcal{U}} = (D, \mathcal{U}_{\lambda''})$, where $\mathcal{U}_{\lambda''} = T \cup \overline{\mathcal{U}} \cup \lambda''$. Note that $a(m)$ cannot be inferred from $\Delta^{\lambda''}_{\mathcal{U}}$. The answer induced by $Q\theta$ and λ'' is "The tape Macunaíma is present, but it is not available because it is reserved."

We conclude with two results that we state without proof (see, however, [2]). The first one establishes the correctness of the interface, in the sense that every answer computed from the user model is indeed correct with respect to the deductive database. The second result shows that the user is forced to reason monotonically with respect to positive information; that is, if at any point during a dialog he can infer a positive fact, then he continues to do so for the next iterations.

Let $\boldsymbol{d}$ be the cooperative domain, $\mathfrak{D}$ be a deductive database, $\mathcal{E}$ be the unique extension of $\Delta_{\mathfrak{D}}$, $\mathcal{U}$ be a user model, and Q be a query. Let C be a cooperative dialog produced by the cooperative interface. Let $\Delta_{\mathcal{U}}$ be the default theory associated with $\mathcal{U}$ and a log λ in C.

Theorem 1: If there is a default proof $\mathfrak{p}$ of Q from $\Delta_{\mathcal{U}}$ in $\boldsymbol{d}$ and θ is the answer substitution computed by $\mathfrak{p}$, then the (only) extension of $\Delta_{\mathfrak{D}}$ contains $\forall Q\theta$.

Theorem 2: Let λ_i and λ_j be two logs in C such that $j > i$. Let A be a conjunction of positive literals. Let $\mathfrak{p}$ be a default proof of A from $(D, \mathcal{U}_{\lambda_i})$, where θ is the answer substitution computed by $\mathfrak{p}$. Then, there is a default proof $\mathfrak{p}'$ of $Q\theta$ from $(D, \mathcal{U}_{\lambda_j})$.

IV. Conclusions

We started by describing a default logic interpretation for user models that captures the intuition that, whenever the user needs to derive a fact during his inference process, he must check whether the current log does not indicate that the fact must be rejected. Then, we presented a cooperative interface that teaches the user more facts than he actually asked for. Finally, we indicated that the cooperative interface is correct, in the sense that any log produced during a dialog contains enough information to avoid misconstruals.

References

[1] A. S. Hemerly, M. A. Casanova, and A. L. Furtado, "Exploiting user models to avoid misconstruals," in *1st Int. Workshop Nonstandard Queries and Answers Approach*, Tolouse, France, 1991.

[2] A. S. Hemerly, A. L. Furtado, and M. A. Casanova, "An approach to user models in cooperative systems based on default logic," Rio Scientific Center, IBM Brasil, Tech. Rep. CCR-143, 1992.

[3] A. Kobsa and W. Wahlster, Eds., *User Models in Dialog Systems.* Berlin: Springer-Verlag, 1989.

[4] J. W. Lloyd, *Foundation of Logic Programming.* Berlin: Springer-Verlag, 1987.

[5] R. Reiter, "A logic for default reasoning," *Art. Intel.*, vol. 13, pp. 81-132, 1980.

Calculating Salience and Breadth of Knowledge

Lisa F. Rau

***Abstract*—As computer programs grow to contain more information, it will become more important, when faced with a new system, to be able to ask, "*What do you know about?*" This correspondence paper overviews some recently completed research [1] investigating three questions: 1) what it means for a computer to know what it knows about, 2) how a computer can construct a representation of what it knows about, and 3) how such a representation can be used for practical applications that advance the state-of-the-art in understanding the content of large databases.**

***Index Terms*—Artificial intelligence, cognitive modeling, computer science, database management, information management, information retrieval, knowledge discovery.**

I. Introduction

What does it mean for a system to have a representation of the contents of its own knowledge? Two aspects of knowledge about knowledge are described in this overview:

Breadth: Breadth of knowledge is how complete knowledge is across a dimension, and

Manuscript received June, 1992; revised December, 1992.

The author is with the Artificial Intelligence Laboratory, GE Research and Development Center, Schenectady, NY 12301.

IEEE Log Number 9212804.