A Case Study on Seamless Support for Combined Knowledge Discovery and Query Answering*

Marcelo A. T. Aragão and Alvaro A. A. Fernandes

Department of Computer Science
University of Manchester
{m.aragao|a.fernandes}@cs.man.ac.uk

Abstract. Inductive and deductive inference are both essential ingredients of scientific activity. This paper takes a database-centred view some of the crucial issues arising in any attempt to combine these two modes of inference. It provides an overview of a new class of database systems that support the execution of composite tasks, each of whose steps may involve knowledge discovery (as an inductive process) and or query answering (as a deductive one). The paper then describes in detail a case study where the specific characteristics of such systems can be more vividly perceived as being relevant and nontrivial.

1 Introduction

One of the central obstacles to improving the level of automated support for scientific discovery in data-rich and knowledge-poor areas such as genomics is that only in the last few years has there been a concerted attempt to reconcile inductive and deductive inferential processes. As a result of this, the rate of growth of data stocks (e.g., in genomics, nucleotide sequences) has outstripped by far the rate of growth in knowledge stocks (e.g., in genomics, knowledge of protein function).

This paper uses a case study that, although not set in a scientific domain, crisply illustrates the significant difference (in comparison with the decoupled tools currently available) made by a recently proposed class of logic-based database systems that seamlessly combines arbitrarily complex inductive (in the form of knowledge discovery) and deductive (in the form of query answering) steps.

The remainder of the paper is structured as follows. Section 2 briefly reviews the technical development of CID systems. Section 3 shows that CIDS can provide an elegant solution to a challenging example in the bank supervision area. Section 4 contrasts our work with other attempts at integrating knowledge discovery and databases technology and indicates where CIDS are novel and uniquely useful. Finally, Section 5 draws few conclusions.

* Longer version of the article in Proc. Discovery Science 2004.

2 An Overview of Combined Inference Database Systems

The main technical goal in the proposal of CID systems [5,6] is the development of a unified formal foundation for both query answering and knowledge discovery. To achieve that, a logic-based approach [8] is adopted that builds upon the views of query answering as a deductive task [8] and of knowledge discovery as an inductive one [11]. CID systems are unique in responding to the non-trivial challenge of combining deductive and inductive processes in a single formalism that directly leads to an operational semantics closely resembling that of classical database management systems. In this section, the technical development of CID systems (reported in detail elsewhere [5,6]) is briefly reviewed.

The Underlying Logical Framework The logical framework underlying CID systems is *parametric Datalog* (or p-Datalog, for short) [17]. p-Datalog extends Datalog [8] to contexts where absolute validity is too restrictive (e.g., reasoning under uncertainty), or not a reasonable expectation (e.g., predictive models of empirical phenomena). Interpreting the parameters of p-Datalog as means to regulate the conditions under which asserted, deduced and induced information can co-exist in the system state provides CID systems with a principled treatment of this important issue, previously largely unaddressed. There are two kinds of parameters. Each clause is annotated with a *validity assessment* and a triple of functions, referred to as *validity combinators* that, taken together, determine precisely how the validity assessment of a clause is propagated in p-Datalog derivations in which it appears as a premiss. These are shown to lead to a logic that enjoys the same desirable metalogical properties as Datalog (e.g., a proof-theoretic, a model-theoretic and a fixpoint semantics that coincide) and of which Datalog (and hence relational databases) are a special case. For example, in the *p-Datalog clause* (p-clause, for short) $\mathsf{p(X,Y)} \hookleftarrow 0.8 \vdash \mathsf{q(X,Z)},\ \mathsf{p(Z,Y)}\ \langle min, min, max \rangle$, 0.8 denotes a value taken from a validity domain, and min and max denote the functions that return, respectively, the smallest and the largest value in a set. Given a validity domain [5], a validity assessment denotes the degree of validity that can be asserted of, or computed for, a given clause, in that domain. The validity combinators are referred to as the conjunctive, propagation and disjunctive combinators, respectively, Their respective role is to determine how the validity of the conjoined literals in the body are to be combined, how the validity of the body is propagated to the clause as a whole, and how the multiple validity assessments associated with alternative (i.e., disjunctively understood) derivations of the same clause are to be combined into a single validity assessment for the clause. In CID systems, validity assessments and combinators are instrumental in reconciling query answering, which, being deductive, is truth-preserving with knowledge discovery, which being inductive, is not. Specific instantiations of CID systems embody different choices of validity assessment domains and validity combinator functions. The one in this paper has been proved (in [5]) to have equivalents to many of the metalogical properties of p-Datalog [17] (and hence, of Datalog [8]).

CIDS as Deductive Database Systems A **combined inference database** (CID) $\mathcal{D}$ is a pair $\langle IDB, EDB \rangle$ where IDB is a set of p-Datalog rules, referred to as the *intensional database*, and EDB is a set of p-Datalog ground facts, referred to as the *extensional database*. The usual syntactical constraints [8] imposed on a Datalog-based deductive database and on p-Datalog programs are also imposed on a CID. Also, the clauses in IDB and in EDB are constrained to belong to a given language $\mathcal{L}_{\mathcal{D}}$. A *CID schema* defines the lexicon for $\mathcal{L}_{\mathcal{D}}$ and may include relation names whose extent is not defined either extensionally in EDB or intensionally in IDB. These undefined relation names are, precisely, those contained in an explicit language bias. Their definition is, therefore, to be computed by inductive inference, thereby characterizing discovered knowledge.

Unlike a deductive database, in which the inferential apparatus is fixed, the inferential apparatus of CID system is formalized separately. Thus, a CID system is a pair $\langle \mathcal{D}, \mathcal{I}_{\mathcal{D}} \rangle$ where $\mathcal{D}$ is a CID and $\mathcal{I}_{\mathcal{D}}$ is a non-empty set of inference rules. A CID inference rule takes as premisses, and infers as consequences, p-clauses in $\mathcal{L}_{\mathcal{D}}$. CID inference rules are either deductive (in which case they have as consequences p-facts, i.e., information in extensional form), or inductive (in which case they have as consequences p-rules, i.e., information in intensional form). In either case, the definition of a CID inference rule specifies precisely which validity assessment consequences should be annotated with, given the validity assessments and combinators that annotate the premisses involved.

A single underlying logic and a unified inferential apparatus, taken together, reconcile deductive inference and inductive inference in a principled manner. Thus, compositionality arises, as expected, as the characteristic property of processes that combine CID inference rules in arbitrarily complex formal derivations over the p-Datalog clauses in a CID. More specifically, combinations of inference rules in $\mathcal{I}_{\mathcal{D}}$ are used to model, as described below, complex tasks that seamlessly and reliably compose query answering and knowledge discovery steps.

CIDS as Classical Database Systems In addition to the model-theoretic and fixpoint semantics that extend and adapt those in [17], a novel operational semantics for CIDS has been defined in terms of a database calculus and algebra such as underpin classical query processing engines. The benefits of this technical move is to ensure that CID systems admit of an implementation strategy in which they emerge as classical database systems, unlike, e.g., the many approaches to deductive databases on the one hand, and proposals such as [7,16,18,19], on the other.

The combined inference tasks that CID systems evaluate are combinations of inference rules in $\mathcal{I}_{\mathcal{D}}$. Now, every such rule is a comprehension expression, in the Fegaras-Maier monoid calculus [12] over subsets of $\mathcal{L}_{\mathcal{D}}$, i.e., over collections of p-Datalog clause. It follows that every CID task that is expressible in a given CID system can be compiled into a monoid calculus expression. This, in turn, can be translated (using the results in [12]) into an equivalent, optimizable monoid algebra, and then into a concrete, physical counterpart of the latter, e.g., [22].

CID Inference Rules as Monoid Comprehensions Both deductive and inductive CID inference rules are expressed as monoid comprehensions. The monoid calculus is based on the abstract-algebraic notion of a *monoid*, whose operations generalize query-language operations over scalar, nested, aggregate (e.g., derived using count, sum, etc.), and parametric-collection values (e.g., sets, bags and lists). A *monoid comprehension* over a monoid $\oplus$ is an expression of the form $\oplus\{e \,\|\, w_1, w_2, \ldots, w_n\}$, $n > 0$. The merge function $\oplus$ is called the *accumulator*, the expression e the *head*, and the sequence $w_1, w_2, \ldots, w_n$ the *body*, of the comprehension. Each term w_i is called a *qualifier*, and is either a *generator* of the form $v \leftarrow D$, where v is a *range variable* and D is a *generator domain* (i.e., an expression that denotes a collection or a nested comprehension expression); a *filter* p, where p is a predicate over some range variable in scope; or a *binding* of the form $v \equiv \alpha$, where v is a new range variable and α is an expression (i.e., binding is a notational convenience with textual replacement semantics). All monoid comprehensions can be reduced to an expression of the form $\oplus\{e(V_1, V_2, \ldots, V_n) \,\|\, V_1 \leftarrow D_1, \ldots, V_n \leftarrow D_n, p(V_1, V_2, \ldots, V_n)\}$. This canonical expression denotes a result R as follows:

```
R := 𝒵⊕                                          //where, e.g., 𝒵⊕ is ∅ if ⊕ is ∪
for each V1 in D1 :
    ...
    for each Vn in Dn :
        if p(V1, V2, ..., Vn)  then
            R := R ⊕ 𝒰⊕(e(V1, V2, ..., Vn))  //where, e.g., 𝒰⊕(e) is {e} if ⊕ is ∪
return R
```

Collections of p-Datalog clauses (e.g., EDB, IDB, $\mathcal{L}_{\mathcal{D}}$) are, of course, monoids. An inner monoid comprehension can be referred to anywhere that a variable of the same monoid type might occur in the outer monoid comprehension, viz., in the head, in the domain of a generator or of a binding, and in a filter. Crucially for this paper, comprehension nesting can denote the binding of the consequences derived by one CID inference rule to an input premisses to another one into arbitrarily complex topologies. So, comprehension nesting underpins the compositionality property of CID inference rules expressed as monoid comprehension.

Given this technical background, the monoid calculus can be used to express CID inference rules, and their composition. For example, the monoid comprehension (a) in Figure 1 characterizes the *elementary production* [8] deductive inference rule upon which the Datalog *bottom-up naive evaluation* procedure is defined. The comprehension scans *IDB* and *EDB* for candidate premisses. These are filtered to ensure that they have a most general unifier (with $\triangledown$ denoting that the *mgu* function failed to find one). Those premisses that satisfy the filter justify the ejection of the corresponding consequence, which is constructed by applying the substitution θ to the head of the first premiss. The major difference from a Datalog setting is the use of validity combinators to compute the validity assessment that annotates the consequence from those asserted to the premisses (in the last line in *eep*). Then, the CID inference rule query_answer, in (a), simply applies *eep* in a particular way, viz., taking as premiss the query (represented as a singleton) and elements from the set of all *eep*-inferrable consequences of

$$
\begin{array}{l}
\mathit{eep(IDB,EDB)} \equiv \\
\uplus\{\ \mathit{substitute}((h \leftarrow\!\dashv V \vdash), \theta)\ \rrbracket \\
\quad (h \leftarrow\!\dashv V_0 \vdash l_1, \ldots, l_n) \leftarrow \mathit{IDB}, \\
\quad (r_1 \leftarrow\!\dashv V_1 \vdash) \leftarrow \mathit{EDB}, \\
\quad \ldots, \\
\quad (r_n \leftarrow\!\dashv V_n \vdash) \leftarrow \mathit{EDB}, \\
\quad \theta \equiv \mathit{mgu}([l_1, \ldots, l_n], [r_1, \ldots, r_n]), \\
\quad \theta \neq \nabla, \\
\quad V \equiv f^{\leftarrow}(V_0, f^{\wedge}(V_1, \ldots, V_n)) \\
\} \\
\mathit{query\_answer}(Q, \langle \mathit{IDB}, \mathit{EDB} \rangle) \equiv \\
\quad \Phi(\mathit{eep}(\{Q\}, (\mathit{EDB} \cup \mathit{eep}(\mathit{IDB}, \mathit{EDB}))))
\end{array}
$$

(a) Bottom-Up Naive Evaluation [8]

$$
\begin{array}{l}
\mathit{association\_rules}(L, O, \langle B, D \rangle, M) \equiv \\
\cup\{\ (h(Y,F) \leftarrow\!\dashv V \vdash l_1, l_2, \ldots, l_n)\ \rrbracket \\
\quad (h(Y,F) \leftarrow l_1, l_2, \ldots, l_n)) \quad \leftarrow L, \\
\quad Q_b \equiv (q(Y) \leftarrow l_1, l_2, \ldots, l_n), \\
\quad X_b \equiv \mathit{query\_answers}(Q_b, \langle B, O \cup D \rangle), \\
\quad Q_h \equiv (q(Y) \leftarrow l_1, l_2, \ldots, l_n, h(Y,F)), \\
\quad X_h \equiv \mathit{query\_answers}(Q_h, \langle B, O \cup D \rangle), \\
\quad |X_h| \geq M, \\
\quad V \equiv \frac{|X_h|}{|X_b|} \\
\}
\end{array}
$$

(b) Association Rules [1,10]

Fig. 1. Examples of CID Inference Rules

$IDB \cup EDB$. Φ in (a) (see the details in [5]) enforces the uniqueness constraint on alternative derivations [17].

Similarly, the monoid comprehension (b) in Figure 1 builds upon (a) to characterize *association* analysis [1]. It chooses a candidate rule from the language bias L. A language bias L denotes a set of Datalog clauses and, if not an enumeration, can be an expression in one of several generative formalisms, e.g., [9,21]. The generated rules are range-restricted in such way that the constant terms in the head of each rule allows co-occurrence to be defined in terms of query containment, addressing the representational issues raised in [10], at least for a class of rule patterns as expressive as conjunctive queries. Then, the candidate rule, gives rises to two queries Q_h and Q_b, that are posed, using the *query_answers* comprehension defined in (a), against the background data $O \cup D$ and knowledge B. This results in the sets X_h and X_b of *eep*-inferrable consequences generated by augmenting B with the candidate rule. A candidate rule r is considered to have matched the requirements for ejection if its Herbrand base has a cardinality above the support threshold M. Q_h is contained in Q_b, by definition, and the cardinality of X_h and X_b determines the significance of a candidate rule r. The validity assessment for the ejected consequence r is computed, in the last line of *association_rules*, as the *confidence measure* of r [1]. For CID inference rules that capture classification and hierarchical clustering, see [6].

CID systems exhibit both *closure*, as a result of the uniform representation language, and *compositionality*, as a result of the uniform representation of tasks as compositions of inference rules in a formalism that exhibits compositionality. As a result of the uniform representation for CID inference rules, CID systems are characterized by *seamless evaluation*, insofar as arbitrary compositions of deductive and inductive CID inference rules are evaluated by one single database engine, thereby minimizing overheads for the system and impedance for end users due to context switches. Database engines based on the monoid calculus and algebra have proved their merit and one of the authors has been involved in developing several of these, most recently a distributed query engine for the Grid [2]. Finally, the *extensibility* path of CID systems is simpler than most, insofar as new deductive and inductive CID inference rules need only be cast as monoid comprehensions as exemplified in 1.

The next section shows how these characteristics of CID systems make a significant difference in a challenging, real-world application context.

3 A Case Study

The Application Context In open economies, banks hold foreign currency stocks resulting from their customers' trade on imports and exports. This gives rise to an interbank market where currency surpluses and deficits are traded. This market acts to balance currency supply and demand and provides banks with profit opportunities when prices fluctuate and is regulated by each country's *bank supervising authority* (BSA). BSAs set limits and margins which banks must abide by so that risks are managed, as stipulated in international agreements.However, in their attempt to maximize profits, banks often resort to speculative practices that can lead to severe losses if the market shifts suddenly. Since, in the limit, such practices may undermine the economic stability of entire countries and even propagate to linked economies, BSAs must elicit the knowledge that will justify intervention, coercion and, ultimately, revision of regulations.

BSAs have a team of inspectors to analyse the data on banks and on their transactions throughout the day. Their goal is to shortlist those banks, if any, whose pratices merit closer scrutiny. However, these practices are, by definition, hard to spot, because they often exploit loopholes in the regulations and typically arise by composition of several, individually-legitimate transactions. They tend, therefore, to lie hidden in the huge volume of interbank trade. Worse still, banks roughly know (or are adept in predicting) where inspection is likely to focus. Provided there is the intention to do so, they are often able to cover signals that could be more easily identified.

From the viewpoint of the inspectors, the target is clearly an elusive, and moving, one: they must learn quickly, and then learn again and again, which practices are being deployed that may merit intervention. Thus, in trying to keep abreast with constantly changing practices, BSA inspectors would benefit from automated, flexible, expressive support for composing knowledge discovery with query answering steps. The ability to do so could underpin an on-demand, exploratory mode of interaction with the data that best serves the aim of eliciting new (or refreshing stale) knowledge about potentially risky practices.

The problem faced by BSAs is that eliciting this knowledge in this particular way requires combining knowledge discovery and query answering steps seamlessly (i.e., without incurring costs and delays that would defeat the purpose of swift response) and reliably (i.e., in a principled, justifiable manner as behoves the BSA's crucial role in promoting economic stability). Unfortunately, if one takes a representative range of knowledge discovery techniques, one finds that state-of-the-art support for combining knowledge discovery and query answering steps is neither seamless (insofar as existing tools are independent, totally decoupled, from one another and hence offer no compositionality unless glueware is specifically developed for that purpose) nor reliable (insofar as there is no principled way to automate the piping of the outputs of one tool into another that considers the validity assessments of the original and intermediate outcomes) across that range.

Alternative Task Models Table 1 contrasts two task models. To the left, one based on the best support that can be offered by decoupled tools (granting that specific products may offer specifically integrated support for specific combinations of knowledge discovery and query answering steps, e.g., a classification model can be constructed and then invoked in a prediction join in Microsoft SQL Server). To the right, the alternative offered by CID systems, which can seamlessly and reliably combine knowledge discovery and query answering steps.

Table 1 both describes and comments on the alternative task models, thus hinting at usability benefits in deploying CID systems when compared with decoupled tools for query answering and knowledge discovery. The remainder of this section discusses in detail one instance of what inspector might use a CID system to explore and how they can go about it. The problem is real and is known in the relevant literature as *the tandem strategy* since it involves co-ordinated action between different banks.

Characterizing Cases of Tandem Strategy The task for a BSA inspector is to shortlist banks for inspection. The case study describes how CIDs would help identify candidates for inspection due to evidence of a tandem strategy. Insofar as a goal of the paper is to argue for the usability benefits of CID systems, the description below is cast in the form of an inspector's possible train of thought.

The inspector starts from the belief that transactions between medium-sized banks with similar business profile are often similar. Violation of this assumption raises the suspicion of collusion between the banks involved. The inspector is after sophisticated speculative practice in which (often volatile) close partnerships are formed between several banks.

The task model is the one to the right in Table 1. The CID involved stores data on banks and their assets, their buying and selling limits, their exposure to risk and their position (i.e., their balance) for a given time frame in days. It also stores data on transactions (e.g., the banks involved, the currency in question, the value, the contracted date, the time frame for clearance). Transactions that only differ in their id are grouped into a entry in a transaction group relation and annotated with a count of the number of tuples that share those attribute values. The extent of such relations would contain p-clauses such as:

bank(b1, commercial, gru, international, 2000000000) $\leftarrow\!\dashv 1.0 \vdash$
risk_exposure(b1, 30, 70000000) $\leftarrow\!\dashv 0.9 \vdash$
balance(b1, 30, 40000000, bought) $\leftarrow\!\dashv 1.0 \vdash$
transaction(b1, b3, usd, low, 20031107, 30, 410000) $\leftarrow\!\dashv 1.0 \vdash$

limits(Bank, 1000000, 10000000) $\leftarrow\!\dashv 1.0 \vdash$ bank(Bank, _, _, _, Assets), Assets> 1000000000

Decoupled Tools	CID Systems
1. Inspectors have at their disposal DBMSs and data mining workbenches. 2. Throughout the day, they make use prepared interfaces to the tools in (1) that they have commissioned the IT staff to implement so that they can retrieve, clean and transform the data they wish to analyse. 3. They may, e.g., execute prepared queries over the DBMSs. For example, they may seek for specific indicators, e.g., banks effecting unusually high-valued transactions, or operating too close to their legal limit. 4. They may also execute mining scripts that, again, the IT staff has prepared to help them search for promising models of the data in (2). 5. Steps (3) and (4) above may lead an inspector to request the IT staff to integrate a promising model so that it can be invoked from the queries in (3) and to expect this model to be deployed with minimal delay. 6. Inspectors constantly make notes about their explorations, since they will need to explain the knowledge they have elicited and provide the data that supports the former. Such explanations will perforce rely on the various tasks on various tools in (1) that will need to be structured, somehow, so as to aid understanding. 7. At close of play, the inspectors draw a shortlist of any targets for inspection they have succeeded in identifying.	1. Inspectors have at their disposal a CID system that is configured with the necessary inference rules and populated with the same data as in (2) to the left and, in addition, domain knowledge (e.g., regulations, best practices) in the form of logical rules. 2. Throughout the day, inspectors rely only on this CID system. Because combined inference tasks are declarative, expressive and capable of being composed with closure, and because the interaction model supports on-demand explorations, each inspector can him or herself devise and evaluate CID tasks without resorting to IT staff. 3. They may, e.g., interactively explore the data either making no assumptions about it or starting from accumulated background knowledge. They can compose querying and discovering steps under the assurance that closure is guaranteed by uniform representations, and compositionality, by a principled treatment of validity assessments. 4. The declarative nature of CID tasks and the uniform clausal representation mean that a simple log of tasks executed and results obtained comes close to being self-explanatory and may stand in lieu of ad-hoc notes. 5. At close of play, the inspectors draw a shortlist of any targets for inspection by simply consolidating their explorations into one combined CID task that, upon execution, derives that shortlist.

Table 1. Task Models for BSA Inspectors: Contrasting Quality of Support

Note the validity assessments that annotate each p-clause. In particular, the validity assessment of 0.9 in the second fact above asserts that the computed risk exposure measure is thought to be less than certain. Recall that every close is annotated with same triple of validity combinators, viz. $\langle min, min, max \rangle$.

Note also that while all other relations are extensionally defined in the EDB, limits/3 is a rule-defined view in the IDB. Thus, CIDs can use rules to capture simple norms and regulations, such as exemplified above. As an another example, the inspector can explicitly represent the operating assumption with the following clause:

```
supervision_target(Bank)←
  mid_sized(Bank),
  mid_sized(Partner_Bank),
  similar_banks(Bank, Partner_Bank),
  transaction_group(Transaction1, Bank, Partner_Bank, 20030510, _, _, _, _, Count1),
  transaction_group(Transaction2, Bank, Partner_Bank, 20030510, _, _, _, _, Count2),
  dissimilar_transactions(Transaction1, Transaction2).
```

The extent of this rule contains mid-sized banks that have engaged in dissimilar transaction with similar partners. The inspector might then formalize the intuition about what is a mid-sized bank as follows:

mid_sized(Bank)$\leftarrow\!\!\dashv 1.0 \vdash$bank(Bank,_,_,_,_,Assets),Assets $>$ 10000000,Assets $<$ 100000000

So far, all the specified information can be obtained via query answering, i.e., deductively. However, the concepts of *similar_banks* and *dissimilar_transactions* appeal, in this case, to knowledge that is sufficiently complex and volatile to merit being induced on the fly, instead of axiomatically asserted once and for all. Thus, the inspector plans for these relations to be defined in terms of inducible ones. These inducible relations can be taken to be those resulting from a clustering step, in the sense that banks that are placed in the same cluster are taken thereby to be similar, and likewise for transactions. This means that, provided the inspector is able to induce clusters on banks and on transactions (i.e., bank_cluster/2 and transaction_cluster/2 resp., below), s/he can capture bank similarity and transaction dissimilarity in the obvious way, as shown to the left and right below, respectively:

similar_banks(Bank1, Bank2)$\leftarrow\!\!\dashv 1.0 \vdash$
  bank_cluster(Bank1, Cluster1),
  bank_cluster(Bank2, Cluster2),
  Cluster1 $=$ Cluster2.

dissimilar_transactions(Transct1, Transct2)$\leftarrow\!\!\dashv 1.0 \vdash$
  transaction_cluster(Transct1, Cluster1),
  transaction_cluster(Transct2, Cluster2),
  Cluster1 $\neq$ Cluster2.

Thus, the computational implementation of this investigative approach can be seen to depend on deductive and inductive steps. Indeed, the case above is an instantiation of a hypothetico-deductive cycle not unlike those that empirical science often relies upon, with the induction of the clusters being followed by their use in deducing similarity relations over a particular body of gathered evidence.

Specifying CID Tasks CID systems, in this case study, have a significant, positive usability impact because they support the seamless combination of inductive, e.g., clustering, steps and classical querying, i.e., deductive, steps. Once the inspector has come up with a problem-solving strategy, specifying it as a CID task will lead to an executable request to identify banks that are supervision

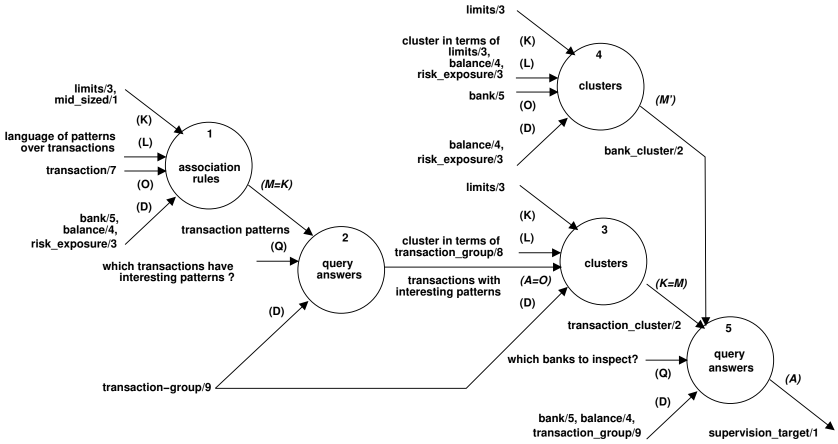

Fig. 2. Shortlisting Banks Suspected of Engaging in Tandem Strategies

targets because of suspicions that they are engaging in a tandem strategy. This task can be formulated and submitted for a CID system (e.g., the proof-of-concept prototype developed by the authors and described in [5]) to execute.

A refinement would be to try and filter the transaction according to some criterion. One option is to refine the hypothesis definition directly, but this may be too blunt in cases, such as the one in hand, where the knowledge is volatile. So, instead, the inspector decides to only cluster transactions that are, in a sense to be defined, unusual. One way to characterize what is a unusual transaction is to use association mining to find interesting co-occurrence patterns. Then, the inspector would only need to cluster transactions that instantiate such patterns.

The inspector first tries to induce a set of association rules to check whether they define an appropriate selection criterion. Satisfied with that, the inspector can then proceed to refine the task by inserting an association analysis step before clustering the transactions thus filtering out those that less likely to be unusual. This task is a complete specification of the inductive-deductive argument for shortlisting a bank on the grounds that it is suspected of engaging in tandem strategies. Figure 2 depicts the complete CID task in graphical form. Nodes denote inferential rules (and hence, steps in CID tasks). Directed edges denote inputs and outputs. Edges are labelled with the p-clause collections they denote. When the output of one step is used as input of another the the two edges are unified, as indicated by their labels. This is possible because of closure and leads to compositionality, since such graphs can be arbitrarily complex. Every occurrence of edge unification is manifested as variable sharing between nested comprehensions in the textual form of a CID task. Note that a bracketed letter indicates which kind of p-clause collections instantiate the corresponding edge. Thus, K, Q, L, and M are p-rule collections (with Q a singleton) and stand for background knowledge, query, language bias and model, respectively. D, A, and O are p-fact collections and stand for data, answers and observations (i.e., examples, or training instances), respectively.

Evaluating CID Tasks As explained above, a CID task such as depicted in graphical form in 2 can be expressed as a monoid comprehension. Two of the CID inference rules involved (viz., *query_answer* to a monoid calculus expression that nests the definition of the CID inference rules presented in Section 2 and in [6]. The translation of this expression into an equivalent expression in the monoid algebra is performed exactly as described in [12]. Then, the resulting logical plan is optimized into an execution plan consisting of operators in a physical algebra and then evaluated using an iterator-based paradigm [13]. The execution of a physical plan retrieves stored p-Datalog clauses (including the lazy generation of p-Datalog clauses specified in language biases); propagates clauses upwards with the validity assessments arising from intermediate operations; and finally, outputs the set of p-Datalog clauses produced by the root operator as the final answer. A prototype implementation of a CID engine was built, and is described in [5] was build, that supports exactly the evaluation strategy for CID tasks. The execution of the above task in the prototype implementation has proved the CID task in Figure 2 effective in detecting tandem strategies in data sets known to contain concrete historical examples of such phenomena. For instance, a p-fact such as `supervision_target(b2)`$\leftarrow\!\!\dashv 0.7 \vdash$ might be part of the task outcome. Note that the validity assigned to such p-facts embodies the principled treatment of the validity asserted of the axiomatic data, and computed for the induced association and clustering models.

4 Related Work

Some extensions of SQL, and even new SQL-like languages, with constructs that denote mining algorithms have been proposed [14,15,20]. Since these approaches involve hard-wired knowledge discovery functionalities, they deliver modest results in terms of closure, compositionality and seamlessness.

The use of inductive inference in deductive databases to derive information in intensional form was first suggested in [7]. However, to the best of our knowledge, the technical development of this idea was never pursued further. In another approach, [19] hints of the idea of combining data and knowledge stocks into a class of information management tools called *inductive databases*. Again, the published details make it difficult to estimate the nature of the task of, and the effort involved in, formalizing and implementing this class of systems. The proposal in [16] was the first to encompass several data mining techniques, as CID systems do. [16] provides a constraint-based algebra to combine the results derived by different invocations of data mining algorithms, however, this comes short of a principled approach to the composition of deductive and inductive steps, as provided in CID systems. Finally, the proposal in [18] instantiates some ideas in [19]. It uses version spaces to induce strings over boolean constraints that capture simple forms of knowledge. In contrast, CIDS use a fully-fledged database calculus and algebra, and operate on a more expressive knowledge representation, viz., Datalog clauses. In general, the lack of a reconciliation of the inductive and deductive methods into a unified foundation has led all of [7,19,16,18] to lack both flexibility and a clear route for incorporation into mainstream database technology.

More closely related to CIDS is previous work by the authors on applying specific combinations of query answering and knowledge discovery techniques to specific problem areas. For example, an extension of inductive relations of [7] was implemented as a prototype knowledge management platform [4] and later used to characterize web-services substitutivity [3]. However, neither [4] nor [3] have a database-centred operational semantics and, hence, no clear path to scalability. Moreover, unlike these, CID systems represent in the underlying logic the distinction between outcomes with different validity assessments. The user-defined assimilation policies of [4] and [3] only allow limited closure.

The capabilities that CIDS make available are not matched by commercial database and data-mining systems. Modulo user-interfacing facilities, the state-of-the-art for commercial database systems is to equate data mining algorithms to user-defined functions invokable in SQL queries. IBM Intelligent Miner allows classification and clustering models coded in XML to be deployed and applied to row sets. Microsoft SQL Server with OLE DB DM provides a comparable mechanism, viz., *prediction join*, to build and apply classification models. In SPSS Clementine, one can specify compositions of data preparation, mining, and visualization steps, but no more than one mining step. In all the above, compositionality and closure are very restricted and evaluation is far from seamless across the two inference modes involved. Levels of coupling are often so loose that significant engineering effort is required for interoperation to be possible.

5 Conclusions

The main contribution of this paper is detailed argument, based on a realistic case study, for the significant value added by the use of CID systems in lieu of the decoupled tools that, bundled in suites and workbenches or not, represent the state-of-the-art in combining query answering and knowledge discovery steps. The paper has shown that CID systems exhibit a collection of properties that make them a good candidate for supporting the combination of inductive and deductive inference that is an essential ingredients of scientific activity. Crucially, CID systems have a database-centred operational semantics, i.e., they rely on optimizing execution plans for scalability. They have better chances, therefore, of avoiding the pitfalls inherent in approaches based on heuristic search. This principled, well-founded reconciliation of inductive and deductive inferential processes contributes to the removal of one of the central obstacles to improving the level of automated support for scientific discovery in data-rich and knowledge-poor areas such as genomics. As a result of this step forwards, the way is open for the engineering of industrial-strength CID systems whose impact may be to attenuate the gap between very steep growth rates for data stocks and very flat ones for knowledge stocks. It is thus a concrete hope that CID systems may become important tools in the broad are of e-science. In particular, the way is wide open for the deployment of CID system on computational grids as they share the same semantics and the same architectural patterns as high-level Grid middleware such as [2], which the authors have helped design and implement.

Acknowledgement Marcelo A. T. Aragão gratefully acknowledges the support from the Department of Computer Science and the University of Manchester and from the Central Bank of Brazil.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. VLDB'94*, pages 487–499. Morgan Kaufmann, 1994.
2. M. Alpdemir, A. Mukherjee, A. Gounaris, N. Paton, P. Watson, A. Fernandes, and D. Fitzgerald. OGSA-DQP: Service-based distributed querying on the grid. In E. B. et al., editor, *Proc. EDBT 2004*, volume 2992 of *LNCS*, pages 858–861. Springer, 2004.
3. M. A. T. Aragão and A. A. A. Fernandes. Characterizing web service substitutivity with combined deductive and inductive engines. In T. Yakhno, editor, *Proc. ADVIS'02*, volume 2457 of *LNCS*, pages 244–254. Springer, Oct. 2002.
4. M. A. T. Aragão and A. A. A. Fernandes. Inductive-deductive databases for knowledge management. In N. Matta and R. Dieng, editors, *Proc. ECAI KM&OM'02*, pages 11–19, July 2002.
5. M. A. T. Aragão and A. A. A. Fernandes. Combining query answering and knowledge discovery. Technical report, University of Machester, 2003. Available from `http://www.cs.man.ac.uk/~alvaro/`.
6. M. A. T. Aragão and A. A. A. Fernandes. Logic-based integration of query answering and knowledge discovery. In H. Christiansen and M.-S. Hacid, editors, *Proc. 6th Flexible Query Answering Systems (FQAS'2004)*, LNCS, pages 68–83, Lyon, France, July 2004. Springer.
7. F. Bergadano. Inductive database relations. *IEEE TKDE*, 5(6):969–971, 1993.
8. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* 1990. ISBN: 0387517286.
9. L. Dehaspe and L. D. Raedt. Dlab: A declarative language bias formalism. In Springer, editor, *Proc. ISMIS'96*, volume 1079 of *LNCS*, pages 613–622, 1996.
10. L. Dehaspe and H. Toivonen. Discovery of relational association rules. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 189–212. Springer, Sept. 2001.
11. S. Džeroski and N. Lavrač, editors. *Relational Data Mining.* Springer, Berlin, Sept. 2001. ISBN: 3540422897.
12. L. Fegaras and D. Maier. Optimizing object queries using an effective calculus. *ACM TODS*, 25(4):457–516, 2000.
13. G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, June 1993.
14. J. Han, Y. Fu, W. Wang, J. Chiang, O. R. Zaïane, and K. Koperski. DBMiner: interactive mining of multiple-level knowledge in relational databases. In H. V. Jagadish and I. S. Mumick, editors, *Proc. SIGMOD'96*, pages 550–550, Montreal, Canada, June 1996. ACM Press.

15. T. Imielinski and A. Virmani. MSQL: A query language for database mining. *DMKD*, 3(4):373–408, Apr. 1999.
16. T. Johnson, L. V. S. Lakshmanan, and R. T. Ng. The 3W model and algebra for unified data mining. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proc. 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 21–32, Cairo, Egypt, Sept. 2000. M. Kaufmann.
17. L. V. S. Lakshmanan and N. Shiri-Varnaamkhaasti. A parametric approach to deductive databases with uncertainty. *IEEE TKDE*, 13(4):554–570, 2001.
18. S. D. Lee and L. de Raedt. An algebra for inductive query evaluation. In *Proc. ICDM'03*, 2003.
19. H. Mannila. Inductive databases and condensed representations for data mining. In J. Maluszynski, editor, *Proc. ILP'97*, volume 13, pages 21–30. MIT Press, Oct. 1997.
20. R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proc. VLDB'96*, pages 122–133, Mumbai (Bombay), India, 3–6 Sept. 1996. Morgan Kaufmann.
21. F. C. Pereira and D. H. Warren. Definite clause grammars for natural language analysis. *Artificial Intelligence*, 13:231–278, 1980.
22. S. Sampaio, N. W. Paton, J. Smith, and P. Watson. Validated cost models for parallel OQL query processing. In *Proc. OOIS'02*, LNCS, pages 60–75, 2002.