Foundations and Discovery of Operational Definitions

Jan M. Żytkow and Zbigniew W. Raś

Computer Science Dept. University of North Carolina, Charlotte, N.C. 28223
e-mail: zytkow@uncc.edu & ras@uncc.edu
also Institute of Computer Science, Polish Academy of Sciences

Abstract. Empirical equations and rules are important classes of regularities that can be discovered in databases. We concentrate on their role as definitions of attribute values. Such definitions can be used in many ways in a single database and for transfer of knowledge between databases. We analyze *quests* for definitions of an attribute in a given database. A quest triggers a discovery mechanism that specializes in searching recursively a system of databases and returns a set of partial definitions. We introduce the notion of shared operational semantics founded on an equation-based and rule-based system of partial definitions. It gives necessary foundations for designing local query answering systems in a distributed knowledge system (DKS).

1 Shared semantics for distributed autonomous DBs

In many fields, such as medical, manufacturing, banking, military and educational, similar databases are kept at many sites. Each database stores information about local events and uses attributes suitable for locally collected information, but since the local situations are similar, the majority of attributes are compatible among databases. Yet, an attribute may be missing in one database, while it occurs in many others. For instance, different military units may apply the same battery of personality tests, but some of these tests may be not used in one unit or another.

Missing attributes lead to problems. A recruiter new at a given unit may query a local database S_1 to find candidates who match a desired description, only to realize that one component a_1 of that description is missing in S_1 so that the query cannot be answered. The same query would work in other databases but the recruiter is interested in identifying suitable candidates in S_1.

1.1 System architecture

Operational semantics introduced in [15] provides definitions of missing attributes through search for definitions in many databases. Figure 1 shows the architecture of a distributed knowledge system. Discovery Layer for each database is initially formed from rules and equations extracted from that database. They define some of the attributes by other attributes in the same database and are

discovered by an automated process. They are used for knowledge exchange between databases and jointly form an integrated semantics for a distributed knowledge system defines the meaning of queries. Query answering system QAS uses definitions extracted at other databases and/or available in the local discovery layer to answer queries which otherwise would not be reachable.

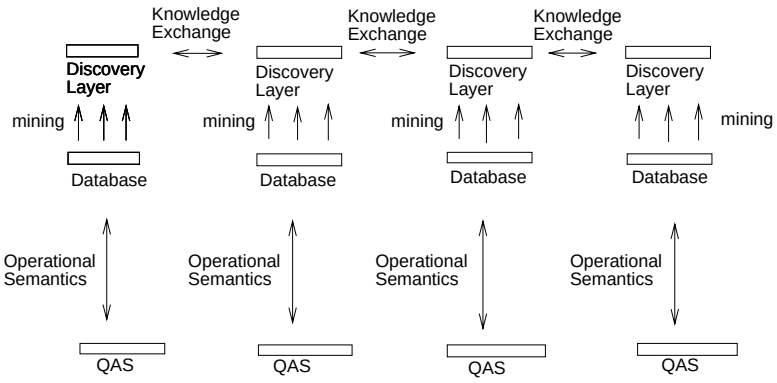

Fig. 1. Distributed Knowledge System

1.2 Links to previous research

QAS is a natural knowledge-discovery-based extension of the query answering system for a system of databases, presented in [12], [13]. In [12], [13] we used rules discovered in one database to define values of missing attributes in other databases. The search for rules can use many strategies which find rules describing decision attributes in terms of classification attributes. It has been used in conjunction with such systems like $LERS$ (developed by J. Grzymala-Busse [3]) or $AQ15$ (developed by R. Michalski and his collaborators [8]).

The task of integrating established database systems can be complicated not only by the differences between the sets of attributes but also by differences in structure and semantics of data. The notion of an intermediate model, proposed by Wiederhold [7], is very useful in dealing with such a problem, because it describes the database content at a relatively high abstract level, sufficient for a homogeneous representation of all databases. In our paper a discovery layer can be seen as an application of the ideas of an intermediate model for a distributed DB system.

1.3 Operational definition

Definitions that are used to compute attribute values of objects are often called operational definitions. They are common in science, where values of each at-

tribute are determined in many ways, depending on different applications. Operational semantics has been introduced by Bridgman [1] and developed by Carnap [2] and many others, including semantics of coherent sets of operational definitions developed by Zytkow [16] and applied in robotic experiments [18].

Operational semantics can be applied to databases. Many computational mechanisms can be used to define values of an attribute. We call them operational definitions because each is a mechanism by which the values of a defined attribute can be computed. Many are partial definitions, as they apply to subsets of records that match the "if" part of a definition. In 1989-1990, Ras et al. [6], [14] introduced a mechanism which first seeks and then applies as definitions rules in the form "If Boolean-expression(x) then a(x)=w" which are partial definitions of attribute a applicable to all objects x that satisfy Boolean-expression(x).

The 49*er* system can find knowledge in many forms, including equations, that can be used to define one attribute by other attributes in a relational table. We conducted experiment, using this mechanism in addition to rule-based definitions [4]. The growing interest in KDD will make the discovery of operational definitions increasingly popular. Recently, Prodromidis & Stolfo [10] argued that attribute definitions are a useful target for discovery in databases.

1.4 Shared semantics in action: query answering

Many query-answering situations can benefit from the following generic scenario. A query q is issued at database S_1, but it is "unreachable" in S_1 because it uses an attribute a which is missing in S_1. A request for a definition of a is issued to other sites in the distributed autonomous database system. The request specifies attributes $a_1, ..., a_n$ available at S_1. When attribute a and a subset $\{a_{i_1}, ..., a_{i_k}\}$ of $\{a_1, ..., a_n\}$ are available in another database S_2, a discovery mechanism is invoked to search for operational definitions at S_2, by which values of a can be computed from values of some of $a_{i_1}, ..., a_{i_k}$. If discovered, such a definition is returned to the discovery layer over S_1 and used to compute the unknown values of a that occur in query q.

The same mechanism can apply if attribute a is available at S_1, but some values of a are missing. In that case, the discovery mechanism can be applied at S_1, if the number of the available values of a is sufficiently large.

1.5 Other applications

Functional dependencies in the form of equations are a succinct, convenient form of knowledge, useful in many ways. The equation $a = f(a_1, a_2, ..., a_m)$ can be directly used to predict values $a(x)$ of a for object x by substituting the values of $a_1(x), a_2(x), ..., a_m(x)$ if all are available. If some are not directly available, they may be predicted by other operational definitions.

When we suspect that some values of a may be wrong, an equation imported from another database may be used to verify them. An equation acquired at the same database may be used, too. For instance, patterns discovered in clean data can be applied to distinguish wrong values in the raw data.

Equations that are generated at different sites can be used to cross-check the consistency of knowledge and data coming from different databases. If the values of a that are computed by two independent equations are approximately equal, this confirms consistency of both definitions.

All equations by which values of a can be computed expand the understanding of a. Attribute understanding is often initially inadequate when we receive a new dataset for the purpose of data mining. We may know the domain of values of a, but we do not understand a's detailed meaning, so it is difficult to apply background knowledge and the knowledge discovered about a. In such cases, an equation that relates a poorly understood attribute a with attributes of known meaning, explains some of the meaning of a.

2 Recursive search for equations

Let us present in algorithmic details a recursive discovery mechanism that supports global query answering. When an attribute a is needed but unreachable in database S_1, a request for a definition of a is issued to other sites in the distributed database system. The request specifies the attributes $a_1, ..., a_n$ available in S_1, because only those attributes can be included in a definition useful at S_1.

In this section we present a recursive algorithm that searches for equations and we analyze an application os this algorithm. But that algorithm can be used to search for rules. In section 3 we will present an example of recursive search for rules.

When the attribute a and a subset $\{a_{i_1}, ..., a_{i_k}\}$ of $\{a_1, ..., a_n\}$ are available in another database S_2, $49er$'s discovery mechanism is invoked to search S_2 for equations by which values of a can be computed from values of some of $a_{i_1}, ..., a_{i_k}$. If discovered, such equations are returned to S_1 and can be used in numerous ways.

In [15] we considered a computational mechanism that searches at each database individually for equations suitable in a role of definitions of a. But there are numerous situations when this mechanism must be expanded and applied recursively.

2.1 Non-overlapping attribute sets

First, there may be no database which contains a and any of $\{a_1, ..., a_n\}$. This can be illustrated with the following example of simple relation schemas, one relation per database:

$S_1(a_1, a_2, ..., a_n)$; definition of attribute a is sought
$S_2(a, b_1, ..., b_k)$
$S_3(b_1, a_2, a_3)$

Suppose that an equation $a = f(b_1)$ has been discovered in S_2. It cannot be used in S_1, because b_1 is unavailable. But S_3 includes b_1 and some of $\{a_1, ..., a_n\}$. An equation $b_1 = f_1(a_2, a_3)$ may be discovered that defines b_1 in terms a_2 and

a_3. That equation can be substituted into $a = f(b_1)$ leading to equation $a = f(f_1(a_2, a_3))$ that can be applied in S_1.

2.2 Search for a sufficient fit

Second, there may be a database S_4 that includes a and some of $\{a_1, ..., a_n\}$. But no equation that defines a through any of $\{a_1, ..., a_n\}$ has a fit sufficient to play the role of a definition. In this situation, the search for a definition can be expanded. Perhaps an equation is discovered that has a sufficient fit to play the role of a definition, but in addition to some of $\{a_1, ..., a_n\}$ it uses b_1, unavailable in S_1. We already discussed the steps appropriate for this situation.

2.3 Empirical contents in a set of definitions

There is a more systematic reason why the search for equations should continue, even if it has been successful. Equations that are used to compute missing values are empirical generalizations. Although they may be reliable, we cannot trust them unconditionally, and it is a good practice to seek their further verification, especially if they are applied to the expanded range of values of a. The verification may come from additional knowledge that can be used as alternative definitions. Ras et al. [12], [13] used rules coming from various sites and verified their consistency.

Multiple equations give a chance for cross-verification, as their predictions can be compared. Each consistent prediction provides extra justification for the system of definitions, while each inconsistency calls for further empirical analysis of data and definition improvements.

2.4 Recursive discovery algorithm

The following algorithm can be used to search recursively for an attribute definition:

Algorithm: Find definitions of attribute a that are applicable in DB

```
For each database X
    if a is available in X then
        seek definition of a in X; keep them on list def(a,X)
for each X and each definition DEF in def(a,X)
    if all attributes that define a are available in DEF,
        then add DEF to list of definitions of a
        else for each attribute b missing in DB
            find definitions of attribute b that are applicable in DB
```

2.5 An example of recursive search for a definition

Consider the following database schemas

$S_1(a_1, a_2, a_3)$,
$S_2(a, a_1, b)$,
$S_3(a, b, a_3)$,
$S4(a2, b)$

also illustrated in Figure 2. Discovery layer is assigned to each of these four databases and contains definitions of attributes and/or attribute values extracted from them. Definition of attribute a is sought.

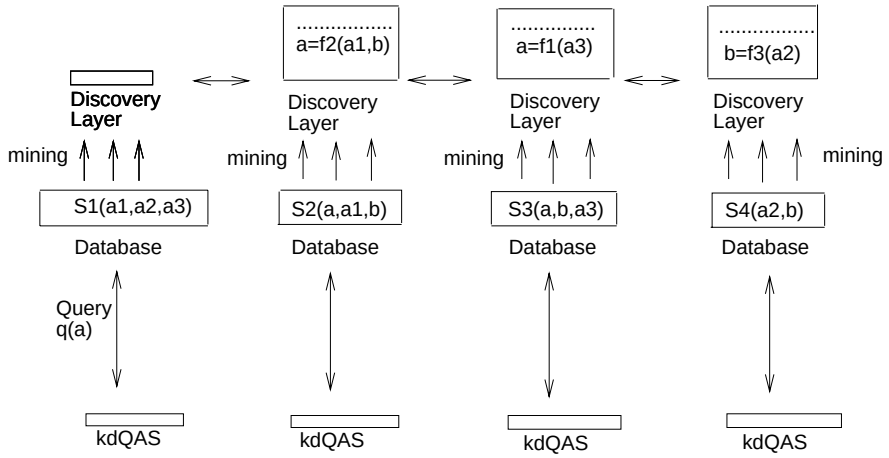

Fig. 2. Search for equations in support of query answering; an example

The recursive search for a definition follows these steps:

1. S_1 sends a request: define a, use a_1, a_2, a_3
2. S_2 and S_3 try to answer.

3a. Situation 1: definition found in S_3: $a = f_1(a_3)$

3b. Situation 2: no definition found, so the search is expanded to additional attributes: define a, use a_1, a_2, a_3 and any other parameters available.

4. S_2 tries to answer.

5a. Situation 3: definition not found so the search halts.

5b. Situation 4: definition found in S_2: $a = f_2(a_1, b)$

6. A new quest is issued: define b, use a_1, a_2, a_3
7. S_3 and S_4 try to answer.

8a. Situation 5: definition not found so the search halts.

8b. Situation 6: definition found in S_4: $b = f_3(a_2)$

9. Equation in 8b is substituted into equation in 5b: $a = f_2(a_1, f_3(a_2))$
10. The search halts.

3 Query answering system based on reducts

In this section we recall the notion of a reduct and show how it can be used to improve the query answering process in distributed autonomous database systems ($DADS$). We assume that information stored in all databases is consistent.

Let us assume that $S = S(A)$ is a database schema and $S(X, A)$ represents its view. Each $a \in A$ is interpreted here as a function $a : X \longrightarrow Dom(a)$, where $Dom(a)$ is a domain of a. For simplicity reason we assume that $Dom(a), Dom(b)$ are disjoint for any $a, b \in A$ such that $a \neq b$.

Let $B \subset A$. We say that $x, y \in X$ are indiscernible by B in S, denoted $[x \approx_B y]$, if $(\forall a \in B)[a(x) = a(y)]$.

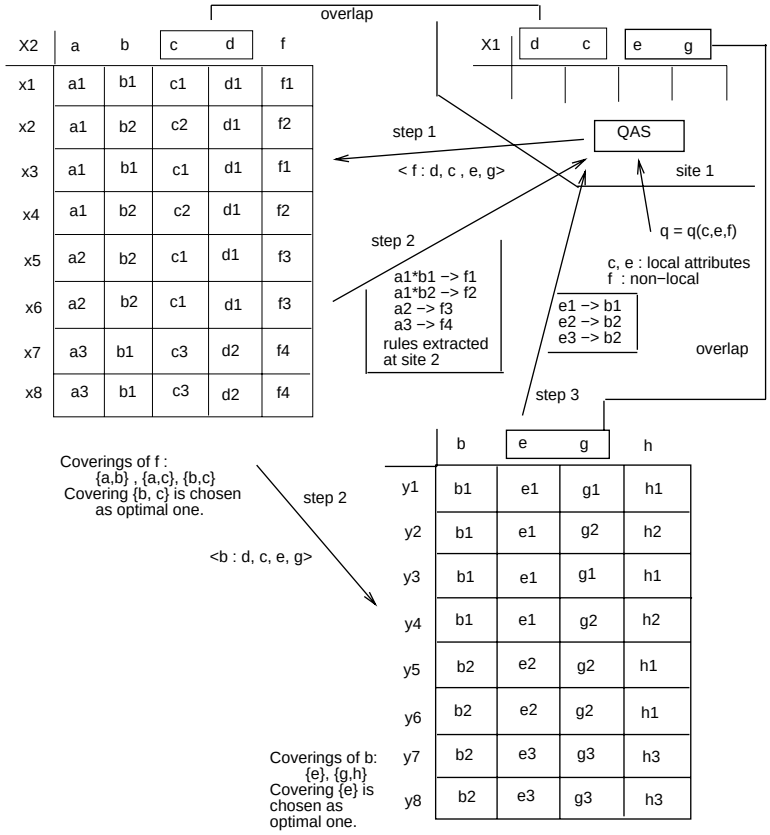

Fig. 3. Process of resolving a query by QAS in $DADS$

Now, assume that both B_1, B_2 are subsets of A. We say that B_1 depends

on B_2 if $\approx_{B_2} \subset \approx_{B_1}$. Also, we say that B_2 is a reduct of B_1 (B_1-reduct) if B_1 depends on B_2 and B_2 is minimal. If B is a singleton set ($B = \{f\}$) then instead of B-reduct we say f-reduct.

Example. Assume the following scenario:

- $S_1 = (X_1, \{c, d, e, g\})$, $S_2 = (X_2, \{a, b, c, d, f\})$, $S_3 = (X_3, \{b, e, g, h\})$ are views of database schemas S_1, S_2, S_3, respectively.
- User submits a query $q = q(c, e, f)$ to the query answering system QAS associated with database S_1,
- Databases S_1, S_2, S_3 form a distributed autonomous database system $DADS$.

Attribute f is non-local for a database S_1 so the query answering system associated with S_1 has to contact other sites of $DAKS$ requesting a definition of f in terms of $\{d, c, e, g\}$. Such a request is denoted by $< f : d, c, e, g >$. Assume that the database S_2 is contacted. The definition of f, extracted from S_2, involves only attributes $\{d, c, e, g\} \cap \{a, b, c, d, f\} = \{c, d\}$. There are three f-reducts (coverings of f) in S_2. They are: $\{a, b\}, \{a, c\}, \{b, c\}$. The optimal f-reduct is the one which has minimal number of elements outside $\{c, d\}$; in our case $\{a, c\}$ and $\{b, c\}$. Let us assume that $\{b, c\}$ is chosen as an optimal f-reduct in S_2.
Then, the definition of f in terms of attributes $\{b, c\}$ may be extracted from S_2 and the query answering system of S_2 will contact other sites of $DADS$ requesting a definition of b (which is non-local for S_1) in terms of attributes $\{d, c, e, g\}$. If a definition of b is found, then it is sent to QAS of S_1. Figure 4 shows the process of resolving query q in the example above.

4 Conclusion

The discovery layer at each site is formed from partial definitions either extracted from that site or imported from other sites. All these partial definitions (if consistent) define a local operational semantics and the meaning of queries seen by the local query answering system. Discovery processes update discovery layers associated with all databases in $DADS$. They do it in real-time whenever a local query cannot be answered with the help of local operational semantics. As a result of operational definitions discovery, local semantics is augmented with a relevant selection of global knowledge in $DADS$.

References

1. Bridgman, P.W., *The Logic of Modern Physics*, The Macmillan Company, 1927
2. Carnap, R., "Testability and Meaning", in *Philosophy of Science, 3*, 1936
3. Grzymala-Busse, J., "LERS - A system for learning from examples based on rough sets", in *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, Slowinski, R. (ed), Kluwer Academic Publishers, 1992, 3-18

4. Klopotek, M., Michalewicz, M., Michalewicz, Z., Ras, Z., Wierzchon, S., Zytkow, J., "Discovering knowledge in distributed databases", in *Proc. of 6th International Workshop on Intelligent Information Systems*, 1997, 128-138
5. Kryszkiewicz, M., Rybinski, H., "Reducing information systems with uncertain attributes", in *ISMIS'96 Proceedings*, LNCS/LNAI, Springer, Vol. 1079, 1996, 285-294
6. Maitan, J., Ras, Z., Zemankova, M., "Query handling and learning in a distributed intelligent system", in *Methodologies for Intelligent Systems, IV*, (Ed. Z.W. Ras), North Holland, 1989, 118-127
7. Maluf, D., Wiederhold, G., "Abstraction of representation for interoperation", in *Proceedings of Tenth International Symposium on Methodologies for Intelligent Systems*, LNCS/LNAI, Springer-Verlag, No. 1325, 1997, 441-455
8. Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N. "The multipurpose incremental learning system AQ15 and its testing application to three medical domains." in *Proceedings of the Fifth National Conference on Artificial Intelligence*, Morgan Kaufmann, 1986, 1041-1045
9. Pawlak, Z., "Rough classification", in *International Journal of Man-Machine Studies*, Vol. 20, 1984, 469-483
10. Prodromidis, A.L. & Stolfo, S., "Mining databases with different schemas: Integrating incompatible classifiers", in *Proceedings of The Fourth Intern. Conf. onn Knowledge Discovery and Data Mining*, AAAI Press, 1998, 314-318
11. Ras, Z., "Query answering in DAKS based on reducts", in *Proceedings of FQAS'2000*, Advances in Soft Computing, Physica-Verlag, 2000, will appear
12. Ras, Z., "Resolving queries through cooperation in multi-agent systems", in *Rough Sets and Data Mining* (Eds. T.Y. Lin, N. Cercone), Kluwer Academic Publishers, 1997, 239-258
13. Ras, Z., Joshi, S., "Query approximate answering system for an incomplete DKBS", in *Fundamenta Informaticae Journal*, IOS Press, Vol. 30, No. 3/4, 1997, 313-324
14. Ras, Z., Zemankova, M, "Intelligent query processing in distributed information systems", in *Intelligent Systems: State of the Art and Future Directions*, Z.W. Ras, M. Zemankova (Eds), Ellis Horwood Series in Artificial Intelligence, London, England, November, 1990, 357-370
15. Ras, Z., Żytkow, J.,"Discovery of equations to augment the shared operational semantics in distributed autonomous BD System", in *PAKDD'99 Proceedings*, LNCS/LNAI, No. 1574, Springer-Verlag, 1999, 453-463
16. Żytkow, J., "An interpretation of a concept in science by a set of operational procedures", in *Polish Essays in the Philosophy of the Natural Sciences*, Krajewski W. ed. Boston Studies in the Philosophy of Science, Vol.68, Reidel 1982, p.169-185.
17. Żytkow, J. & Zembowicz, R., "Database exploration in search of regularities", in *Journal of Intelligent Information Systems*, No. 2, 39-81
18. Żytkow, J.M., Zhu, J., and Zembowicz R., "Operational definition refinement: a discovery process", in *Proceedings of the Tenth National Conference on Artificial Intelligence*, The AAAI Press, 1992, p.76-81.

This article was processed using the LaTeX macro package with LLNCS style