A Data Mining Support Environment and its Application on Insurance Data

M. Staudt, J.-U. Kietz, U. Reimer
Swiss Life, Information Systems Research (CH/IFuE), CH-8022 Zurich, Switzerland
{Martin.Staudt,Uwe.Kietz,Ulrich.Reimer}@swisslife.ch

Abstract

Huge masses of digital data about products, customers and competitors have become available for companies in the services sector. In order to exploit its inherent (and often hidden) knowledge for improving business processes the application of data mining technology is the only way for reaching good and efficient results, as opposed to purely manual and interactive data exploration. This paper reports on a project initiated at Swiss Life for mining its data resources from the life insurance business. Based on the Data Warehouse MASY collecting all relevant data from the OLTP systems for the processing of private life insurance contracts, a Data Mining environment is set up which integrates a palette of tools for automatic data analysis, in particular machine learning approaches. Special emphasis lies on establishing comfortable data preprocessing support for normalised relational databases and on the management of meta data.

Introduction

Although the amount of digital data available in most companies is growing fast due to the rapid technical progress of hardware and data recording technology, the lots of valuable information hidden in the data is barely exploited. Instead of huge collections of low-level data we need abstract and high-level information that is tailored to the user's (mostly management people) needs and can be directly applied for improving the decision making processes, for detecting new trends and elaborating suited strategies etc. Unfortunately, the data collections from which to derive this information have a chaotic structure, are often erroneous, of doubtful quality and only partially integrated. In order to bridge the gap between both sides, i.e. to find a reasonable way for turning data into information, we need (efficient) algorithms that can perform parts of the necessary transformations automatically. There will always remain manual steps in this data analysis and information gathering task, like the selection of data subsets and the evaluation of the resulting hypotheses. However, the automatic processing should cover all those parts that can not be handled properly by humans due to the size of transformation input and output.

Knowledge discovery in databases (KDD) aims at the automatic detection of *implicit*, previously *unknown* and potentially *useful* patterns in the data. One prerequisite for employing automatic analysis tools is a consolidated and homogenized set of data as input. Data Warehouses provide exactly this, thus forming the ideal first steps in setting up a KDD process.

In order to explore how Data Mining tools can complement its Data Warehouse, Swiss Life set up a data mining project which is concerned with the design and implementation of the Data Mining environment ADLER[1]. In particular, ADLER aims at enabling end users to execute mining tasks as independently as possible from data mining experts' support and at making a broad range of data mining applications possible. It turned out that the most crucial factor for a successful application of mining algorithms is a comprehensive support for preprocessing which takes into account meta data about the data sources and the algorithms. Although the data warehouse already provides a consolidated and homogeneous data collection as input for analysis tasks we have to deal with certain task- and algorithm-specific data preparation steps which includes further mining-tool-specific data cleaning as well as restructuring, recoding and transformation of the multi-relational data.

The rest of this paper is organized as follows: Section 2 gives some information about the Data Warehouse MASY and the components of the Data Mining environment. Section 3 summarizes the mining technology employed, i.e. the algorithms and their integration. Section 4 describes the meta data management task within the mining environment and in particular explains at an example from our concrete Data Warehouse schema how preprocessing operations can be supported. Section 5 presents several applications for ADLER.

[1] Analysis, Data Mining and Learning Environment of Rentenanstalt/Swiss Life

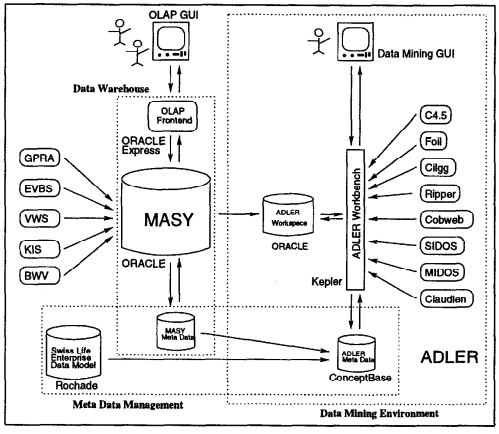

Figure 1: Data Mining Environment ADLER

Data Warehouse and Data Mining Environment

The masses of digital data available at Swiss Life – not only data from insurance contracts but also from external sources, such as information about the socio-demographic structure and the purchasing power of the population in the various parts of the country – led to the development of the central integrated Data Warehouse MASY (Fritz 1996).

MASY comprises data from currently four OLTP systems: contract data (about 700,000 contracts, some 500,000 clients) plus externally available data collections. Some of the data sources are shown in Figure 1. The basic insurance contract data e.g. stems from the systems EVBS and VWS while GPRA contains (personal) data about all Swiss Life clients and partners. BWV is a publicly available catalogue of all (3 Million) Swiss households. Whereas the OLTP systems only contain the actual data, MASY keeps the history of changes, too. MASY is implemented on top of ORACLE and follows a ROLAP warehouse architecture, i.e. employs on top of the relational structures a multidimensional OLAP-frontend. The database itself has both a normalized relational scheme gained from integrating the schemas of all source systems, and a derived (redundant) denormalized Galaxy schema to efficiently support multi-dimensional access for some predefined tasks. The normalized scheme contains around 20 GB of data, distributed over approximately 30 tables and 600 attributes. Figure 2 shows an excerpt of this schema with the 10 main relations.

Based on the homogenized data in MASY, the Data Mining environment ADLER offers tools and a methodology for supporting information extraction from the relational structures. Compared to OLAP tools which mainly concentrate on interactive exploration, abstraction and aggregation of data (which also leads to new information) this extraction takes place semi-automatically. Interpreting the mining results and initiating certain preprocessing on the input data (as discussed below) remains a manual task. Figure 1 shows the overall architecture of ADLER and its relationship to MASY and the OLTP sources respectively.

Data Mining Technology

Since there is not *the* distinguished data mining approach suited for handling all kinds of analysis tasks ADLER integrates a broad palette of approaches, including classical statistical ones as well as techniques from the areas of Machine Learning and Neural Nets.

Figure 1 shows some algorithms that are available. With respect to their output we can distinguish between the following categories of algorithms:

1. detection of associations, rules and constraints: a common application of these techniques are e.g. market basket analyses.
2. identification of decision criteria: with decision trees as one possible output format we can support tasks like credit assignments.
3. discovery of profiles, prototypes and segmentations: for example, classes of customers with similar properties can be grouped together and handled in a uniform way.

Another categorization concerns the kind of input data allowed:

a. sets of attribute-value pairs describing properties of certain data objects represented in one single relation (*attribute-based approaches*) or
b. input tuples from different relations and background knowledge, e.g. Datalog programs (*relational approaches*).

The latter category in particular allows to include additional background knowledge and arbitrary combinations of different classes of data objects. While statistical algorithms as well as most Machine Learning and commercially available Data Mining algorithms are attribute-based, Inductive Logic Programming approaches fall into the second category of relational approaches (Kietz 1996).

Following the two orthogonal categorizations above the algorithms in Figure 1 can be classified as follows:

Tool	O	I	Reference
SIDOS/ Explora	1	a	(Hoschka & Klösgen 1991)
MIDOS	1	b	(Wrobel 1997)
Claudien	1	b	(De Raedt & Bruynooghe 1993)
C4.5	2	a	(Quinlan 1993)
Ripper	2	a	(Cohen 1995)
Cilgg	2	b	(Kietz 1996)
Foil	2	b	(Quinlan & Cameron-Jones 1993)
Cobweb	3	a	(Fisher 1987)

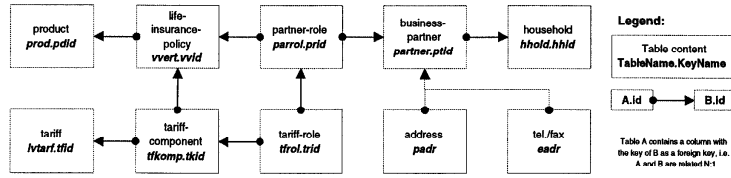

Figure 2: MASY schema excerpt

Offering a great variety of different mining methods requires an open framework that allows the integration of quite different types of algorithms and a high extensibility. The *heart* of the ADLER-Workbench is the Data Mining tool KEPLER (Wrobel *et al.* 1996) which relies on the idea of "Plug-Ins": Its tool description language and a basic API enable the building of wrappers around a given implemented mining algorithm and its inclusion into the applicable set of tools. From a tool description KEPLER can generate input masks for the source data and for setting parameters. The type of the mining results (e.g. rule or decision tree) is also given in the tool description. A result handling specification determines whether the tool itself presents the result, KEPLER does it, or some external display tool should be started.

All the mentioned mining algorithms are based on pure main-memory processing. We assume that the main memory in today's high-end workstations (more than 1 GB) should be sufficient to hold the data needed to obtain optimal results with hypothesis spaces that are still small enough to be searchable in reasonable time. Therefore, employing specific disk-based methods does not seem meaningful to us.

However, according to our experience it is absolutely necessary to do all the preprocessing (including data selection, sampling, recoding, etc.) on the data base directly and not in main memory because the amount of raw input data to be dealt with is in realistic applications by far too large (in our case about 11 GB). As the data warehouse is read-only we have added a database as a cache between the warehouse and ADLER. All preprocessing is done on this cache database. The result of preprocessing is loaded into main memory for the mining task proper. This approach is in contrast to other Data Mining environments which provide preprocessing only for data in main memory, and certainly not for multi-relational data as ADLER does.

KEPLER also offers a number of (main memory based) preprocessing operations and input/output interfaces. For the integration with MASY these facilities were extended by building an interface to the cache database mentioned above on which the preprocessing operations are executed and intermediate data is buffered.

Meta Data Management and Preprocessing Support

We now turn to the management of meta data within ADLER. After explaining its basic role we describe the model that was implemented to capture the meta data relevant for mining with ADLER. We also give an example for the support given by ADLER during the preprocessing and transformation phases.

The Role of Meta Data

Meta data plays an important role for the integration task in distributed information systems, both at design and run time. Figure 1 shows three different types of meta data important in our context:

- The Swiss Life IT department administrates an overall enterprise data model that describes all relevant objects, agents and systems in the company. The actual implementation platform is Rochade.
- The MASY Data Warehouse meta data is mainly stored in certain ORACLE tables. Parts of it have also beeen ported to the Rochade repository. Typically, such meta data contains information about the integration and transformation process (technical meta data) and the warehouse contents itself, e.g. different (materialized) views and levels of abstraction that are available for the end users (business meta data).
- Among the meta data relevant for Data Mining we can distinguish between the following categories:
  - background knowledge, like product models or household data
  - knowledge about data selection and transformations needed to properly feed each mining algorithm with data for a certain task
  - application methodology that gives criteria as to which tools should be used in which way to solve which problems.

The ADLER Meta Model

Instead of handling meta data of single applications (like classical data dictionaries) within the operational database system – a very common way for Data Warehouse solutions – it is much more appropriate to manage it separately from the data and combine it with other kinds of meta data. Consequently, we introduce

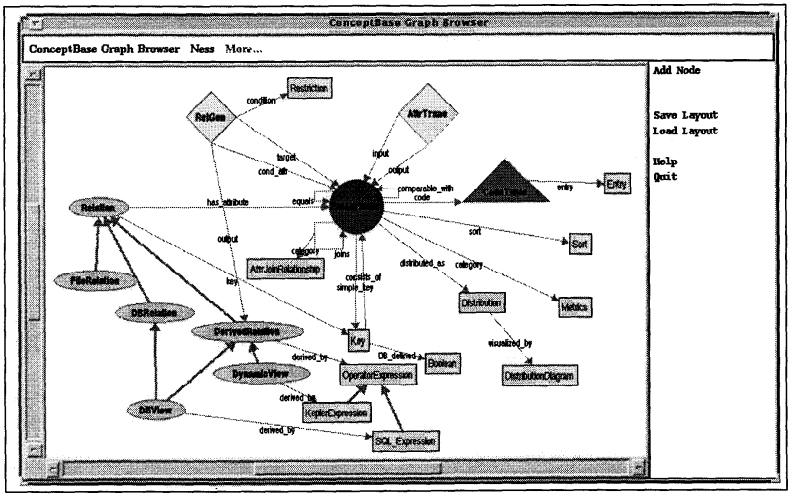

Figure 3: Meta Model of ADLER: Data Representation and Preprocessing

an independent meta data repository with a powerful representation formalism that plays a prominent role throughout the whole KDD process. ADLER employs the deductive and object-oriented meta data management system ConceptBase (Jarke *et al.* 1995) and its knowledge representation language Telos to describe the mining meta data, relevant excerpts of the MASY meta data and the links to the enterprise meta data stored in Rochade.

In the following, we concentrate on meta data describing the MASY schema and certain transformation classes available for the preprocessing phase. Typical preprocessing operations needed to produce or transform attributes are:

Recoding and Discretisation: Most Data Mining tools (including KEPLER) assume a correspondence between the data type of an attribute (e.g. integer, string) and its metric (e.g. nominal, ordinal, scalar). This is usually not the case in a data warehouse so that recoding or even discretisation is needed depending on the capabilities of the Data Mining tool and the intended metric for the Data Mining task.

Scaling: Scaling is needed if the scales of two attributes to be compared are different. Without proper scaling similarity measures, as used for clustering or regression, will produce garbage.

Abstraction and grouping: Nominal attributes with too many different values are useless or at least difficult for Data Mining tools without attribute value grouping (e.g. the profession code in MASY has more than 14000 different values). Such attributes have to be abstracted to be of any use.

Aggregation and construction: Constructed attributes can greatly improve data mining results (Rendell & Seshu 1990). Aggregation is most often needed to incorporate the summary of attributes from a 1:N related table into a single table, e.g. the total amount of benefits from different policies of a single household.

Figure 3 shows the graphical representation of an O-Telos object collection managed by ConceptBase. The displayed objects constitute parts of the meta schema used in ADLER to control the preprocessing and mining activities and to describe the source data and their relationships. We leave out the above mentioned explicit connection to the warehouse sources and start with relations available for the mining process. Apart from database relations (`DBRelation`) we consider also relations extracted from flat text files available as additional background knowledge not integrated in the warehouse (mainly for experimental purposes). The model also provides so-called dynamic views that do not exist in the database but are created on the fly during the KDD process, and it provides database views defined on top of base relations by SQL expressions. All types of relations have columns, namely attributes (instances of `RelAttribute`).

Attribute values are very often discrete and possible values of an attribute can be organized in code tables, particularly for reasons of efficiency. An entry in such a table relates a short code word with the actually intended value. Our database currently employs about 250 code tables with the number of entries ranging from 2 to several hundreds. Discrete attributes can be either nominal or ordinal, a further category comprises car-

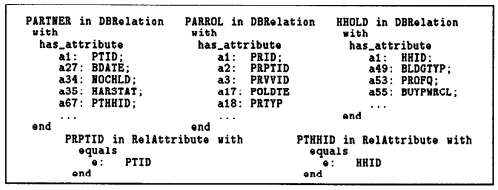

Figure 4: Relation and attribute specification

dinal data which is not only ordered but also allows to measure distances between values. These categories are covered by **Metrics**. Furthermore, from a semantic point of view attributes can be assigned to sorts, like money or years. The data type of an attribute (like **NUMBER**, **STRING**) is also recorded in the meta data base but not shown in Figure 3 as well as the allocation of relations in databases, the owner, and other kinds of user information.

One or more attributes usually constitute the **key** of a relation, i.e. their value(s) uniquely identifies a tuple. Single attribute keys are called **simple_key**, like a client identifier (see table **PARTNER** below).

Particularly interesting for the restructuring of base data by generating new derived relations are relationships between attributes. In the case of basic relationships, e.g. given by foreign keys, we assume the **equals** relationship between the involved attributes. Typically, joins between relations do not involve arbitrary attributes but only those where it makes sense. The model allows to state such attribute pairs by specifying **joins** links, possibly together with additional information concerning the join result (e.g. 1:N or N:M matching values). Links between attributes labeled **comparable_with** give additional hints for possible comparisons during the mining phase. More complicated relationships between attributes result from applying attribute transformation operators (instances of **AttrTrans**) to an attribute (as **input**) to obtain another (new) attribute (**output**). The application of such operators will be explained by the example in the section below. Operators to generate derived relations (**RelGen**) specify **target** attributes of the (new) relation, a **condition** part posing certain restrictions both on these attributes as well as on additional ones (**cond_attr**). From all involved attributes the associated relations needed for executing the necessary joins and for performing the selections can be determined automatically.

In order to give an idea how the meta schema explained above can be used in ADLER at runtime we now show an example stemming from the preparation of our first analyses of the data available in MASY. For the example we use three relations from Fig. 2:

- **PARTNER**: client data relation with key attribute **PTID** for the client number and further attributes, e.g. birthdate (**BDATE**), marital status (**MARSTAT**), number of children (**NOCHLD**);
- **PARROL**: relates insurance policies (**VVERT**) with partners (**PARTNER**) wrt. to their role in the contract (e.g. insured person or policy holder **PRTYP**) and certain other information (e.g., the contract date **POLDTE**) with foreign keys **PRPTID** to **PARTNER** and **PRVVID** to **VVERT**;
- **HHOLD**: household data, mainly general data (e.g. type of flats or houses (**BLDGTYP**), typical professional qualification (**PROFQ**) and buying power class (**BUYPWRCL**) of inhabitants) available for so called cells of 30 - 50 households in Switzerland, with key **HHID**.

The records in **PARTNER** are partly linked to households by the (foreign key) attribute **PTHHID**. Thus we can obtain further information about the client's environment via the household he is living in.

In Telos notation the meta data repository would contain this information as shown in Figure 4. Note, that we *instantiate* the classes in Figure 3. The **has_attribute** values of the instances of **DBRelation** simply name the relevant attributes (in the Telos syntax each value listed has an identifier of its own, given in front of the value followed by a colon). The foreign key relationships mentioned above are modelled as instances of the **equals** relationship from **RelAttribute** to itself.

In order to construct a relation **NPARTNER** for a certain mining task which restricts the client set to those people who made a new contract since the beginning of last year, we perform the transformation steps given below. We are particularly interested in the age and the (weighted) buying power class distribution and want to include the **BLDGTYP** and **PROFQ** attributes from **HHOLD**.

1. The birthday attribute **BDATE** of **PARTNER** has a specific date default for missing values. This default value influences the age distribution, so it should be substituted with **NULL**. The result is an attribute column **BDATE'**.
2. From **BDATE'** and the current date the age **AGE** of each client can be obtained.
3. The buying power class of a client should be weighted with his marital status and the number of children. We first code the values of the marital status as numerical values (new attribute **MARSTAT'**) under the assumption that single persons can afford more things (value **1**) while the value of married people is neutral (**0**) and the value of divorced and separated persons is negative (**-1**).
4. The new weighted buying power class **WBUYPWRCL** results from adding this value to the old class value and further reducing it by one for every two children.
5. Relation **NPARTNER** is defined as consisting of the attributes constructed during the previous steps and those of **HHOLD** mentioned above. In addition to the join between **PARTNER** and **HHOLD** and the attribute transformations a join with **PARROL** is required in order to eliminate clients without "new" contracts.

The join conditions in both cases are derived implicitly from the **equals** links in the model. The selection condition concerning the contract date **POLDTE** has to be given explicitly.

Preprocessing Support: An Example

Each of the above steps is realized by instances of **AttrTrans** (Steps 1–4) and **RelGen** (Step 5). Instances of **AttrTrans** belong to different types of transformation classes (specializations of **AttrTrans**) with suiting parameters. Instances of **RelGen** can be understood as generalized multi-join operators which are applied to attributes only, but with a given selection condition and an implicit derivation of the participating relations. The attribute transformations concern either one attribute (**NullIntr**, **AttrDecode**, **AttrComp** for replacing certain values by **NULL**, performing general value replacements or arbitrary computations) or several attributes (**AttrComb** for arbitrary value combinations).

Step 1: build **BDATE'**

```
NullBDATE in NullIntr with
  input
    i:  BDATE
  output
    o:  BDATE'
  value
    v1: "10000101";
    v2: "99999999"
end
```

Step 2: build **AGE**

```
CompAGE in AttrComp with
  input
    i: BDATE'
  output
    o: AGE
  comp_expression
    c: ":o = substr(today,4,1) - substr(:i,4,1)"
end
```

The **value** attribute (defined for **NullIntr**) specifies the date values (digits of year, month and day) that were 'misused' for marking missing values and should be replaced by **NULL**. The **comp_expression** value in **CompAGE** allows the derivation of the output value **:o** for attribute **o** of **CompAGE** from the input value **:i** of **i** by subtracting the year component of **:i** from the system date **today**.

Step 3: build **MARSTAT'**

```
DecMARSTAT in DecodeAttr with
  input
    i:      MARSTAT
  output
    o:      MARSTAT'
  from
    from1: "single";
    from2: "married";
    from3: "separated";
    from4: "divorced"
  to
    to1:    1;
    to2:    0;
    to3:   -1;
    to4:   -1
end
```

Step 4: build **WBUYPWRCL**

```
CombWBUYPWRCL in AttrComb with
  input
    i1: MARSTAT';
    i2: BUYPWRCL;
    i3: NNOCHLD
  output
    o:  WBUYPWRCL
  comp_expression
    c:  ":o = :i2 + :i1 - (:i3 div 2)"
end
```

Each **from** value in **DecMARSTAT** is replaced by its corresponding **to** value.

Step 5: generate **NPARTNER**

```
GenNPARTNER in RelGen with
  output
    o:  NPARTNER
  target
    t1: PTID;
    t2: AGE;
    t3: WBUYPWRCL;
    t4: BLDGTYP;
    t5: PROFQ
  cond_attr
    a1: POLDTE
  condition
    c: " :a1 > '19960101'"
end
```

Besides the advantage of having a documentation of executed transformations, another motivation for modelling the preprocessing steps is to enable the user to specify the required transformations in a stepwise fashion, while the actual generation of a specified relation is done automatically by ADLER so that the user does not need to have any knowledge of SQL. The automatic generation builds upon the concretely modelled operations like those shown above. The operation classes (like **NullIntr**) are associated via mapping rules to SQL code.

Even for the small number of attributes in our example the resulting (SQL) expressions become very complex and their direct manual specification is error-prone and laborious. Of course, there should also be GUI support for protecting the user from having to type the still cryptic Telos specifications. Furthermore, analyses and therefore also the necessary preprocessing steps are usually reexecuted from time to time on different database states. Storing the transformation sequences speeds up the whole process and facilitates its automation.

Meta Data for Mining Activities

While our plans for recording transformation and preprocessing steps are relatively clear, the considerations of how to represent the actual mining activities are more vague. As a first step it is necessary to model the mining algorithms wrt. different input requirements, parameters and results. Each mining session consists of executing algorithms (corresponding to instantiating the mining model) in order to pursue a certain mining goal. The type of the algorithm and the available data sources obviously relate with the mining goal. We think it is useful to record the complete path from the first transformation in the preprocessing phase to the successfully

generated result (e.g., a decision tree) which satisfies the intended goals. In future sessions (in particular on new database states) it can be reconstructed how the whole analysis process took place. Furthermore, we intend to boil down our mining experience for the various kinds of tasks into instructions and generalized examples of how to best approach similar minings tasks. This forms a very superficial mining methodology only but should still help new users to get more rapidly acquainted with data mining in ADLER. In addition, the preprocessing and mining activities do not constitute a sequential but cyclic process where it is useful even during the same mining session to have the possibility to go back to previous steps in order to reexecute them after having made some modifications.

Applications and Outlook

For Swiss Life, Data Mining has a high potential to support marketing initiatives that preserve and extend the market share of the company. In order to experiment with different mining algorithms a number of concrete applications were identified and selected:

Potential Clients: One might think that the wealthy inhabitants of a certain geographical area are the most promising candidates for acquiring new contracts, but this is usually not the case. An interesting data mining task is therefore to find out what the typical profiles of Swiss Life customers are with respect to the various insurance products. An insurance agent can then use these profiles to determine from a publicly available register of households those non-customers of his geographic area which are quite possibly interested in a certain kind of life insurance.

Customer Losses: One way to reach lower cancellation rates for insurance contracts is via preventive measures directed to customers that are endangered through personal circumstances or better offers from competitors. By mining the data about previous cancellations, using as background knowledge unemployment statistics for specific regions and questionnaire data, we expect to obtain classification rules that identify customers that may be about to terminate their insurance contracts.

Other mining tasks concern the identification of differences between the typical Swiss Life customers and those of the competitors, and the segmentation of all persons in the MASY warehouse into the so called *RAD 2000 Target Groups* based on a set of fuzzy and overlapping group definitions developed by the Swiss Life marketing department some years ago.

A subset of our data was anonymized and made available on the Web for further experiments[2].

[2]It can be obtained from http://research.swisslife.ch/kdd-sisyphus/.

References

Cohen, W. W. 1995. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference (ML95).*

De Raedt, L., and Bruynooghe, M. 1993. A theory of clausal discovery. In Muggleton, S. H., ed., *The Third International Workshop on Inductive Logic Programming.*

Fisher, D. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2(2):139 – 172.

Fritz, M. 1996. The employment of a data warehouse and OLAP at Swiss Life (in German). Master's thesis, University of Konstanz.

Hoschka, P., and Klösgen, W. 1991. A support system for interpreting statistical data. In Piatetsky-Shapiro, and Frawley., eds., *Knowledge Discovery in Databases.*

Jarke, M.; Gallersdoerfer, R.; Jeusfeld, M.; Staudt, M.; and Eherer, S. 1995. ConceptBase - a deductive object base for meta data management. *Journal of Intelligent Information Systems* 4(2):167–192. see also http://www-i5.informatik.rwth-aachen.de/CBdoc.

Kietz, J.-U. 1996. *Inductive Analysis of Relational Data (in German).* Ph.D. Dissertation, Technical University Berlin.

Quinlan, R., and Cameron-Jones, R. M. 1993. Foil: A midterm report. In Brazdil, P., ed., *Proceedings of the Sixth European Conference on Machine Leaning (ECML-93),* 3–20. LNAI 667,Springer.

Quinlan, J. R. 1993. *C4.5: Programms for Machine Learning.* Morgan Kaufmann.

Rendell, L., and Seshu, R. 1990. Learning hard concepts through constructive induction: framework and rationale. *Computational Intelligence* 6:247 – 270.

Wrobel, S.; Wettschereck, D.; Sommer, E.; and Emde, W. 1996. Extensibility in data mining systems. In *Proc. of the 2nd Int. Conf. On Knowledge Discovery and Data Mining.* AAAI Press.

Wrobel, S. 1997. An algorithm for multi-relational discovery of subgroups. In Komorowski, J., and Zytkow, J., eds., *Principles of Data Mining and Knowledge Discovery: First European Symposium (PKDD'97),* 78–87. Berlin, New York: Springer Verlag.

Acknowledgements: We would like to thank the KEPLER team at GMD St. Augustin and Dialogis GmbH for their collaboration and support.[3]

[3]KEPLER is commercially available from Dialogis GmbH (http://www.dialogis.de).