EverMiner Prototype Using LISp-Miner Control Language

Milan Šimůnek and Jan Rauch

Faculty of Informatics and Statistics, University of Economics, Prague
nám W. Churchilla 4, 130 67 Prague 3, Czech Republic
{simunek,rauch}@vse.cz

Abstract. The goal of the *EverMiner* project is to run automatic data mining process starting with several items of initial domain knowledge and leading to new knowledge being inferred. A formal description of items of domain knowledge as well as of all particular steps of the process is used. The *EverMiner* project is based on the *LISp-Miner* software system which involves several data mining tools. There are experiments with the proposed approach realized by manual chaining of tools of the *LISp-Miner*. The paper describes experiences with the *LISp-Miner Control Language* which allows to transform a formal description of data mining process into an executable program.

1 Introduction

The *EverMiner* project is introduced in [8,14]. Its idea is to automate data mining process with help of several items of initial domain knowledge. All items of domain knowledge are formalized as well as particular steps of the process [7]. GUHA procedures [1,3,10] are used as core analytical tools. Input of each GUHA procedure consists of an analysed data and several parameters defining a large set of relevant patterns, output is a set of relevant patterns true in the analysed data.

The *EverMiner* project is based on the *LISp-Miner* system [12,9] which involves several GUHA procedures as well as modules for data preprocessing and dealing with items of domain knowledge. There are several experiments with the proposed approach realized by manual chaining of the procedure mining for generalized association rules and modules of *LISp-Miner* system [11]. Process of data mining with association rules is described by FOFRADAR – a formal frame for data mining with association rules [7] based on observational calculi. The goal of this paper is to describe experiences with a scripting language *LISp-Miner Control Language* (LMCL) which allows to transform a formal description of data mining process into an executable program.

Main features of the *EverMiner* project are outlined in Section 2. An example of a simple and manually implemented concept is described in Section 3. An application of LMCL to run an enhancement of this example is introduced in Section 4. The example concerns medical data. However, the goal of this example is not to get new medical knowledge, but to introduce the LMCL language.

T. Andreasen et al. (Eds.): ISMIS 2014, LNAI 8502, pp. 113–122, 2014.

There are various approaches to describe and automate data mining process [2,5,6,15]. Their detailed comparison with the presented approach is out of the scope of this paper and is left as a further work. Let us mention that they do not use the formalization of a data mining process based on observational calculi.

2 EverMiner Principles

The concept of the *EverMiner* is based on two loops – an outer loop (see the left part of Fig. 1) and an inner loop (see the right part of Fig. 1) in the phase (4) of answering analytical questions formulated in step (3) in the outer loop. A domain expert (1) is necessary to supervise the automated process. But his role is limited mainly to approval or disapproval of newly inferred knowledge. He or she doesn't intervene directly into the outer and inner loops so they could be really automated. Domain knowledge (2) contains both the initial domain knowledge prepared by the domain expert and the newly inferred domain knowledge which is clearly flagged as *data specific* and needs an approval from the domain expert before it could become part of accepted domain knowledge. Nevertheless, newly inferred knowledge could be used immediately to formulate new analytical questions so the whole automated process could carry on even if the expert is busy and not available. He or she could return back later and approve or disapprove the whole bunch of newly inferred knowledge afterwards.

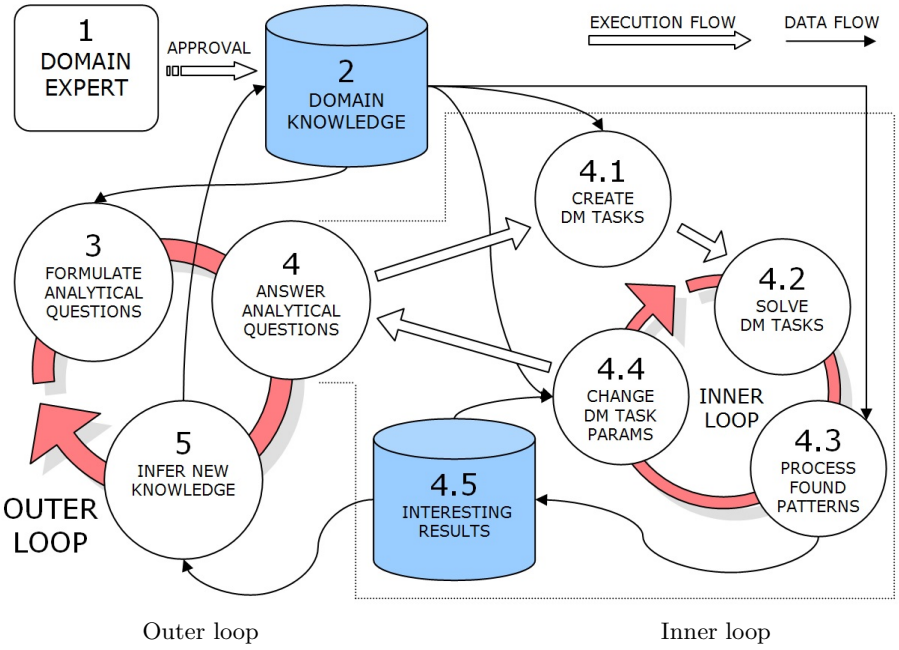


Fig. 1. *EverMiner* seen as two loops

Domain knowledge is formalized in a way both understandable to domain experts and suitable to be used in an automated data mining process. The outer loop starts with currently available domain knowledge (2). There are several ways to formulate reasonable analytical questions based on the current knowledge in step (3). Shortly, we can try (a) to verify that domain knowledge is valid in analysed data, (b) to found new patterns not yet covered by current knowledge or (c) to found exceptions to known patterns.

By answering analytical questions in step (4) we mean to create and solve several data mining tasks, results of which can contribute to a solution of the analytical question formulated in step (3). We plan to start with data mining tasks which can be solved by the GUHA procedures implemented in the *LISp-Miner*. An example of formulation of analytical questions and application of the GUHA procedure *4ft-Miner* dealing with association rules to solve one of the formulated analytical questions is in Section 3. Usually, several applications of GUHA procedures are necessary to solve one analytical question.

3 Mechanical Turk Proof of Concept

3.1 Data Set STULONG

As demo data we use the data matrix *Entry* belonging to the dataset STULONG concerning *Longitudinal Study of Atherosclerosis Risk Factors*[1].The data matrix concerns 1.417 male patients, each row describes one patient. The data matrix has 64 columns corresponding to particular attributes of patients. We use four groups of attributes – *Personal*, *Diet*, *Measures*, and *Examinations*.

Group *Personal* has three attributes: *Marital_Status* (4 categories i.e. possible values), *Education* (4 categories), and *Responsibility* in a job (4 categories). Group *Diet* has two attributes: *Beer* (3 categories) and *Coffee* (3 categories). Group *Measures* has only one attribute *BMI* (i.e. Body Mass Index) with 17 categories – < 21, $\langle 21;22)$, $\langle 22;23)$, $\ldots$, $\langle 35;36)$, ≥ 36. Group *Examinations* has three attributes: *Diastolic* blood pressure (7 categories), *Systolic* blood pressure (9 categories) and *Cholesterol* in mg% (10 categories). Fig. 4 presents the selected attributes and categories.

3.2 Domain Knowledge and Analytical Questions

Three types of domain knowledge related to the STULONG data are managed by the *LISp-Miner* system [11]: *groups of attributes*, *information on particular attributes* and a *simple influence between attributes*. There are 11 basic groups of attributes defined at `http://euromise.vse.cz/challenge2004/data/entry/` and four additional groups of attributes defined in the previous section. Groups of attributes are used to define reasonable analytical questions.

Information on particular attributes include information on types of attributes (nominal/ordinal/cardinal). An example of a simple influence between attributes

[1] see `http://euromise.vse.cz/challenge2004`.

is an SI-formula *BMI* $\uparrow\uparrow$ *Diastolic* saying that if BMI of a patient increases then patient's diastolic blood pressure increases too [11].

We outline how these items of knowledge can be used to define reasonable analytical questions, see outer loop in Fig. 1. We use groups of attributes *Personal*, *Diet*, *Measures*, and *Examinations* and SI-formula *BMI* $\uparrow\uparrow$ *Diastolic*. An example of an analytical question is: *In the data matrix* Entry*, are there any interesting relations between combinations of attributes from* Personal*,* Diet*, and* Measures *on one side and attributes from* Examinations *on the other side?* We denote this analytical question as $\mathcal{AQ}_1$, symbolically

$\mathcal{AQ}_1$: [*Entry*: *Personal*, *Diet*, *Measures* $\approx^?$ *Examinations*].

This question can be enhanced: *In the data matrix* Entry*, are there any interesting relations between combinations of attributes from* Personal*,* Diet*, and* Measures *on one side and attributes from* Examinations *on the other side? However, we are not interested in consequences of the known fact that if BMI increases, then diastolic blood pressure increases too.* Symbolically we can write

$\mathcal{AQ}_{1E}$: [*Entry*: *BMI* $\uparrow\uparrow$ *Diastolic* $\not\rightarrow$ *Personal*, *Diet*, *Measures* $\approx^?$ *Examinations*].

3.3 EverMiner Analytical Questions

Similarly, additional analytical questions can be

$\mathcal{AQ}_2$: [*Entry*: *Diet*, *Measures*, *Examinations* $\approx^?$ *Personal*]

$\mathcal{AQ}_3$: [*Entry*: *Measures*, *Examinations*, *Personal* $\approx^?$ *Diet*]

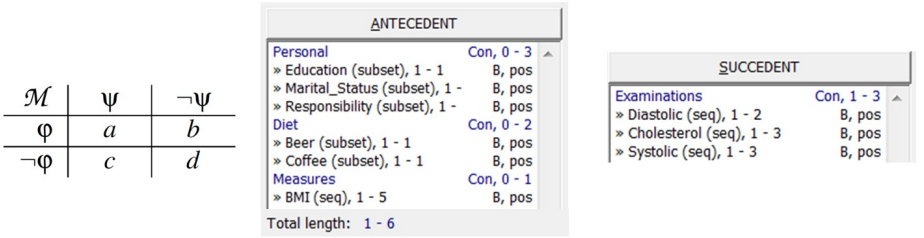
$\mathcal{AQ}_4$: [*Entry*: *Examinations*, *Personal*, *Diet* $\approx^?$ *Measures*].

We assume here that the only item of knowledge of the type *simple influence between attributes* is *BMI* $\uparrow\uparrow$ *Diastolic*. It makes no sense to use this item in the introduced additional analytical questions. We assume here that the analytical questions $\mathcal{AQ}_1$, $\mathcal{AQ}_2$, $\mathcal{AQ}_3$, $\mathcal{AQ}_4$ are the only analytical questions to be solved. However, actually hundreds or thousands of similar analytical questions can be formulated using available groups of attributes and SI-formulas.

3.4 Solving a GUHA Task – An Example

The *EverMiner* project is based on applications of GUHA procedures implemented in the *LISp-Miner* system, there are nine GUHA procedures mining for various types of patterns [9,12]. Each analytical question formulated in the outer loop, see Fig. 1 is transformed into several data mining tasks. We outline how the GUHA procedure *4ft-Miner* can be used to solve the analytical question $\mathcal{AQ}_1$: [*Entry*: *Personal*, *Diet*, *BMI* $\approx^?$ *Examinations*].

The procedure *4ft-Miner* deals with association rules $\varphi \approx \psi$ where φ and ψ are Boolean attributes derived from columns of an analysed data matrix. Boolean attribute φ is called *antecedent* and ψ is called *succedent*. The symbol $\approx$ is a *4ft-quantifier*, it corresponds to a criterion concerning quadruples $\langle a, b, c, d\rangle$ of non-negative integers a, b, c, d such that $a + b + c + d > 0$. The association rule $\varphi \approx \psi$ is true in the data matrix $\mathcal{M}$ if the criterion corresponding to $\approx$ is satisfied



$\mathcal{M}$	ψ	$\neg\psi$
φ	a	b
$\neg\varphi$	c	d

$4ft(\varphi, \psi, \mathcal{M})$ Relevant antecedents Relevant succedents

Fig. 2. $4ft(\varphi, \psi, \mathcal{M})$ and definitions of sets of relevant antecedents and succedents

for a contingency table $4ft(\varphi, \psi, \mathcal{M})$ of φ and ψ in data matrix $\mathcal{M}$, see Fig. 2. Here a is the number of rows of $\mathcal{M}$ satisfying both φ and ψ, b is the number of rows of $\mathcal{M}$ satisfying φ and not satisfying ψ, etc.

Input of the *4ft-Miner* consists of an analysed data matrix, definitions of sets of relevant antecedents and succedents and of a 4ft-quantifier $\approx$. Output is a set of all rules $\varphi \approx \psi$ true in the analysed data matrix where φ is a relevant antecedent and ψ is a relevant succedent.

A definition of a set of relevant antecedents used to solve the analytical question $\mathcal{AQ}_1$ is in Fig. 2. An antecedent is a conjunction $\varphi_P \wedge \varphi_D \wedge \varphi_B$ where φ_P is a Boolean characteristics of the group *Personal*, similarly for φ_D, φ_B and the groups *Diet*, *BMI* respectively. The attribute φ_P is a conjunction of 0 - 3 Boolean attributes created from attributes *Education*, *Marital_Status*, and *Responsibility*. Four Boolean attributes *Education*(*basic*), ..., *Education*(*university*) are automatically created from the attribute *Education*, similarly for *Marital_Status*, and *Responsibility*. The attributes φ_D are created analogously. The attributes φ_B are in a form $BMI(\alpha)$ where α is a sequence of 1–5 consecutive categories of BMI (see expression `BMI(seq), 1 - 5` in the definition of relevant antecedents in Fig. 2). This way 75 Boolean characteristics of BMI are defined, $BMI(< 21, \langle 21; 22), \langle 22; 23))$ i.e. $BMI(< 23)$ being an example. The expressions `B, pos` means that all Boolean attributes are equally important [10].

The set of relevant succedents is defined similarly. Sequences of categories are again used, see right part of Fig. 2. We used the 4ft-quantifier $\Rightarrow_{p,B}$ of founded implication defined by the criterion $\frac{a}{a+b} \geq p \wedge\ a \geq B$. The rule $\varphi \Rightarrow_{p,B} \psi$ means that at least $100p$ per cent of rows of $\mathcal{M}$ satisfying φ satisfy also ψ and that there are at least B rows of $\mathcal{M}$ satisfying both φ and ψ. There are about 20 additional 4ft-quantifiers implemented in the *4ft-Miner*.

We started with the quantifier $\Rightarrow_{0.95,50}$. More than $4.5 * 10^6$ of association rules were verified in 150 sec. (PC with 4GB RAM and Intel i5-3320M processor at 2.6 GHz) and no true rule was found. After about 10 automatically computed modifications of parameters we get 98 true rule for 4ft-quantifier $\Rightarrow_{0.8,30}$. We used the way introduced in [11] to filter out 32 consequences of $BMI \uparrow\uparrow Diastolic$. This is based on transforming the SI-formula to a set of its atomic consequences i.e. all suitable simple rules $BMI(\alpha) \Rightarrow_{p,B} Diastolic(\beta)$ where α, β are subsets of

possible values of *BMI*, *Diastolic* respectively. Then all their consequences are filtered out, deduction rules of the logic of association rules [9] are used.

The remaining rules are used to create a set of interesting results of the application of the GUHA procedure *4ft-Miner* with the 4ft-quantifier $\Rightarrow_{0.8,30}$. Several (10 – 20) strongest rules can be considered as examples of results. Another example of results is an assertion that among found rules 10 are consequences of (yet not considered) item of knowledge $BMI \uparrow\uparrow Systolic$.

4 Applying the LISp-Miner Control Language

Although theoretically proved valid, a manual implementation of the *EverMiner* steps is not feasible. The number of tasks to be solved could easily grow above hundreds and even thousands. Therefore an automated approach is necessary and here the *LISp-Miner Control Language* (LMCL) steps in. LMCL is a scripting language based on Lua and its syntax [4]. The main purpose of LMCL is to provide programmable means to automate all the main phases of data mining i.e. to import data, to pre-process them, to formulate reasonable analytical tasks, to process those tasks and finally to digest found patterns and to report only the interesting ones to the user or to infer directly new items of domain knowledge. In this sense, the language is a necessary prerequisite for automation of data mining process. But, it could serve other purposes too [13].

4.1 EverMiner Stulong Simple Algorithm

The *EverMiner Stulong Simple* demo presented here is really a simplified version of the *EverMiner* concept, in line with the proof of concept described in Section 3. Its main purpose is to proof that the LMCL is able to automate data mining process and to solve many data mining tasks in parallel with speed that would be never possible to achieve through standard user interface.

Few user-defined parameters guide the whole process which is fully automated. Apart from the rather technical ones (e.g. a connection string to database with analysed data), there is one piece of domain knowledge in form of groups of attributes and association of attributes with them. The second important input parameters are the minimal and maximal number of patterns to mine. There are several ways how to reduce (or enlarge) task search space to influence the number of found patterns, but they are out of scope of this paper. Just a very simple heuristic has been implemented for now. Nevertheless, it is successfully exploited in the step (4) of Fig. 1 to ensure the number of found patterns is within given range, see Fig. 3.

After a desired number of hypotheses was reached for each task (or a specified maximal limit of iterations was reached), a summary of task results could be prepared in form of an analytical report. An example of such a report (manually shortened) is in Fig. 4.

There is no space to discuss an automated data preprocessing in this paper (an example is in [13]). The first example where LMCL simplifies and speeds-up the data mining process is in the analytical task formulation phase. There

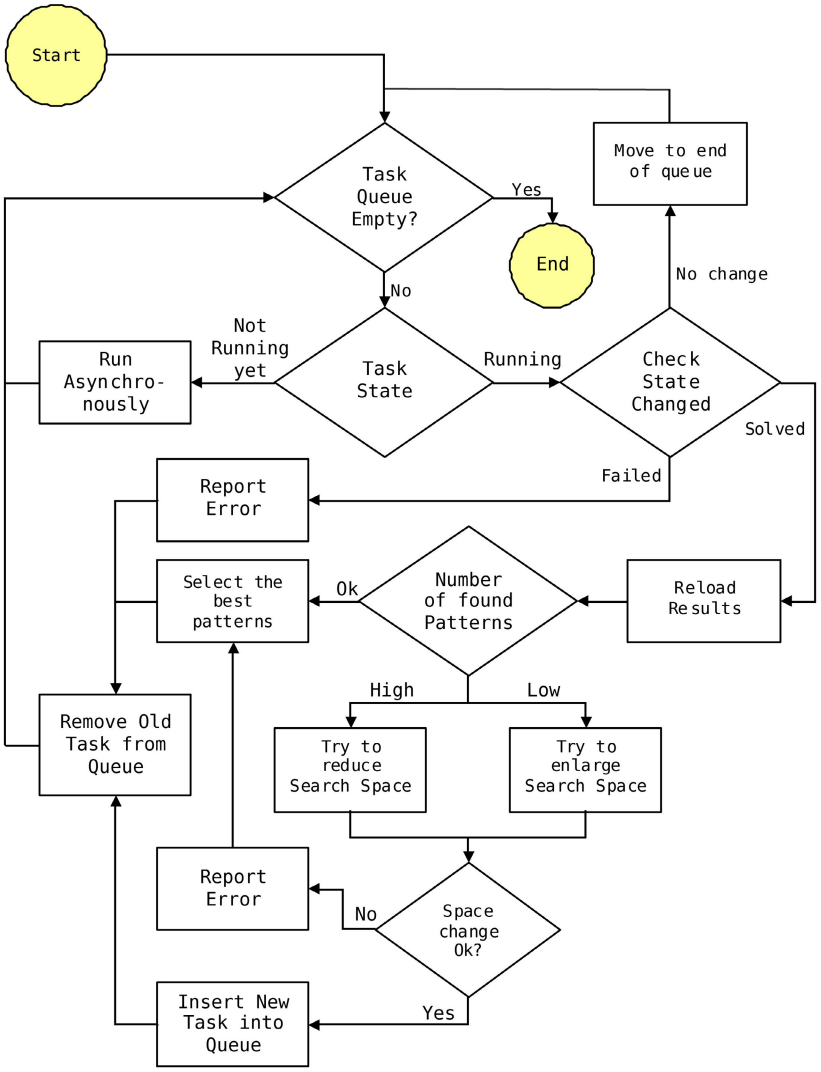


Fig. 3. Asynchronous parallel task processing algorithm description

is a data mining task automatically created for each analytical question. Task parameters are set to pre-defined initial values so the fine-tuning could turn in any direction based on number of hypotheses found. There is a simple heuristic implemented to take a special care of ordinal and cardinal values and to set coefficients of type *sequence* up to length of one third of number categories in corresponding attribute, see an example of LMCL syntax in Fig. 5.

EverMiner Stulong Simple Analytical Report

Data Preprocessing

Diet
Beer: no, 1 or 2 mugs, 3 or more mugs

Examinations
Diastolic: <50;70), <70;80), <80;90), <90;100), <100;110), <110;120), <120;150)

Measures
BMI: <21, <21;22), <22;23), <23;24), <24;25), <25;26), <26;27), <27;28), <28;29), <29;30), <30;31), <31;32), <32;33), <33;34), >=34

Personal
Education: basic, apprentice, full secondary, university
Marital_Status: married, divorced, single, widower
...

Tasks

Diet, Measures, Personal -> Examinations (08)
Task finished succesfully and an acceptable number of patterns has been found
Number of iterations: 8
Found patterns: 20
The most interesting ones:

- **Beer**(*no*) & **Education**(*apprentice*) & **Marital_Status**(*married*) >÷< **Diastolic**(*<80;100)*)
- **BMI**(*<24*) & **Education**(*university*) >÷< **Diastolic**(*<70;90)*)
- **BMI**(*<21;24)*) & **Education**(*university*) >÷< **Systolic**(*<110;140)*)

Fig. 4. Example of an automatically created analytical report (shortened)

```
for j, attribute in ipairs( attributeArray) do

   -- add literal for each attribute
   ftliteralSetting= lm.tasks.settings.FTLiteralSetting({
      pFTPartialCedentSetting= ftpartialCedentSetting,
      pAttribute= attribute
   });

   if ( (attribute.getDataCharacterTypeCode() == lm.codes.DataCharacterType.Ordinal) or
        (attribute.getDataCharacterTypeCode() == lm.codes.DataCharacterType.Cardinal)) then
   -- sequence up to 1/3 of number of categories

       ftliteralSetting.setCoefficientTypeCode( lm.codes.CoefficientType.Sequence);
       ftliteralSetting.MaxLen= math.max( math.floor( attribute.CategoryCount / 3), 1);
   else
   -- default (subset 1-1)
   end;
end;
```

Fig. 5. LMCL syntax example

The most important phase of the *EverMiner Stulong Simple* example is an asynchronous parallel processing of data mining tasks till the desired number of hypotheses is reached, see Fig. 3. It implements the *EverMiner* inner cycle using a task queue. The queue is initially filled with analytical tasks constructed in the step (4.1) of Fig. 1. The inner loop is processed till the queue is empty. The first step pops the top-most task in the queue and checks its state. If the task is not computed yet, it starts an asynchronous generation and verification of patterns by calling the *LISp-Miner ProcPooler* module for background computation of data mining tasks on multiple processor cores of a hosting computer. (Alternatively, the *LM GridPooler* module could be used to utilize distributed grid of computers.)

If the task has already been started, a query is made for the task state update (possibly made meanwhile by the *LM ProcPooler* module in another thread). If the state has not changed yet, the task is moved to the end of the queue and the first step is repeated for another task. If the task has finished with some error, its state is changed to *failed* and the task is removed from further processing. If the task state has been solved successfully, found patterns are loaded and the execution forks based on the number of found patterns. If it is within the defined acceptable range, the *most interesting patterns* (in this simplified version just the *first three*) are marked as *final results* to be included in the analytical report. If the number of found patters is outside the given range, the concerned task settings are changed to enlarge (respectively to reduce) the solution-space searched. Changes to the task settings are limited for now to a change of parameters p, B of the 4ft-quantifier $\Rightarrow_{p,B}$, see Section 3.4. In the above mentioned application 28 runs of the *4ft-Miner* procedure was used which required 2 minutes and 50 seconds.

4.2 LMCL Performance

LMCL is implemented by the *LM Exec* module of the *LISp-Miner* system using Lua script interpreter library of version 5.2. The used Lua interpreter is really lightweight and proved to be fast, so far tested with up to medium-sized scripts (thousands of code-lines). Script parsing and execution overhead costs are insignificant compared to data mining task solution times or to data transfers from database. Performance of LMCL scripts therefore depends solely on ability of the *LISp-Miner* system modules to compute data mining tasks. It has been proved already (see e.g. [10]) that the algorithms and optimizations techniques implemented in the *LISp-Miner* system lead to solution times linearly dependant on number of rows (objects) in analysed data.

5 Conclusions

We have demonstrated the first experience with the *LISp-Miner Control Language* which allows to transform a formal description of a data mining process into an executable program. It was shown that it is possible to use this language to automate data mining processes with association rules. In the next steps we assume to use theoretical results [7] and considerations [8,14] to enhance the described experiments by deeper application of additional formalized items of domain knowledge and other types of patterns.

Also, a deeper comparison of the presented approach with additional approaches, see e.g. [2,5,6,15], is necessary.

References

1. Hájek, P., Havránek, T.: Mechanising Hypothesis Formation - Mathematical Foundations for a General Theory. Springer, Heidelberg (1978)
2. Hájek, P., Havránek, T.: GUHA 80: An Application of Artificial Intelligence to Data Analysis. Computers and Artificial Intelligence 1, 107–134 (1982)

3. Hájek, P., Holeňa, M., Rauch, J.: The GUHA method and its meaning for data mining. J. Comput. Syst. Sci. 76, 34–48 (2010)
4. Ierusalimschy, R., Figueiredo, L.H., de Celes, W.: Lua an extensible extension language. Software: Practice & Experience 26, 635–652 (1996)
5. Mansingh, G., Osei-Bryson, K.-M., Reichgelt, H.: Using ontologies to facilitate post-processing of association rules by domain experts. Information Sciences 181, 419–434 (2011)
6. Phillips, J., Buchanan, B.G.: Ontology guided knowledge discovery in databases. In: Proc. First International Conference on Knowledge Capture, pp. 123–130. ACM, Victoria (2001)
7. Rauch, J.: Formalizing Data Mining with Association Rules. In: Proceedings of 2012 IEEE International Conference on Granular Computing (GRC 2012), pp. 406–411. IEEE Computer Society, Los Alamitos (2012)
8. Rauch, J.: EverMiner: consideration on knowledge driven permanent data mining process. International Journal of Data Mining, Modelling and Management 4(3), 224–243 (2012)
9. Rauch, J. (ed.): Observational Calculi and Association Rules. SCI, vol. 469. Springer, Berlin (2013)
10. Rauch, J., Šimůnek, M.: An Alternative Approach to Mining Association Rules. In: Lin, T.Y., Liau, C.-J., Ohsuga, S., Hu, X., Tsumoto, S. (eds.) Foundations of Data Mining and knowledge Discovery. SCI, vol. 6, pp. 211–231. Springer, Heidelberg (2005)
11. Rauch, J., Šimůnek, M.: Applying Domain Knowledge in AssociationRules Mining Process - First Experience. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS (LNAI), vol. 6804, pp. 113–122. Springer, Heidelberg (2011)
12. Šimůnek, M.: Academic KDD Project LISp-Miner. In: Abraham, A., Franke, K., Köppen, M. (eds.) Intelligent Systems Design and Applications. ASC, vol. 23, pp. 263–272. Springer, Tulsa (2003)
13. Šimůnek, M.: LISp-Miner Control Language – description of scripting language implementation. Submitted for publication in Journal of System Integration, `http://www.si-journal.org` ISSN: 1804-2724
14. Šimůnek, M., Rauch, J.: EverMiner – Towards Fully Automated KDD Process. In: Funatsu, K., Hasegava, K. (eds.) New Fundamental Technologies in Data Mining, pp. 221–240. InTech, Rijeka (2011)
15. Sharma, S., Osei-Bryson, K.-M.: Toward an integrated knowledge discovery and data mining process model. The Knowledge Engineering Review 25 49–67 (2010)
16. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. Information Systems 29, 293–313 (2004)