Editors

Zbigniew Suraj
Chair of Computer Science
University of Rzeszow
Rzeszow
Poland

Ludwik Czaja
Institute of Informatics
University of Warsaw
and Vistula University
Warsaw
Poland

Preface

This two-volume book contains the papers selected for presentation at the Concurrency, Specification and Programming (CS&P) Workshop. It is taking place from 28th to 30th September 2015 in Rzeszow, the biggest city in southeastern Poland.

CS&P provides an international forum for exchanging scientific, research, and technological achievements in concurrency, programming, artificial intelligence, and related fields. In particular, major areas selected for CS&P 2015 include mathematical models of concurrency, data mining and applications, fuzzy computing, logic and probability in theory of computing, rough and granular computing, unconventional computing models. In addition, three plenary keynote talks were delivered.

The Workshop was initiated in the mid-1970s by computer scientists and mathematicians from Warsaw and Humboldt Universities, as Polish-German annual meetings. The first meeting in this series was named 1st Symposium on Mathematical Foundations of Computer Science and it took place in Warsaw from 12th to 19th September 1976. These meetings have been suspended for some years in the eighties until the beginning of ninetieths, and reactivated in 1992. Since then, the Workshop bears the name CS&P, when the first meeting after the break came into effect in Berlin. Now, it is being organized every even year by the Humboldt University of Berlin and every odd year by the University of Warsaw.

It should be mentioned that the CS&P meetings, initially purely bilateral, since 1992 have developed into events attended by participants from a number of various countries beside Poland and Germany. In 2003 the University of Information Technology and Management in Rzeszow, in 2004, the Fraunhofer Institut FIRST in Berlin, and in 2015, the University of Rzeszow joined the organizers as full members of the Committee and financial contributors. The present CS&P 2015 meeting will be hosting participants from the following countries: Canada, Germany, India, Italy, Poland, Russia, Saudi Arabia, Slovakia, Ukraine.

The CS&P 2015 is the twenty-fourth meeting after the break. It received 53 submissions that were carefully reviewed by Program Committee members or external reviewers. After a reviewing process, 49 papers were accepted for presentation at the workshop and publication in the CS&P 2015 proceedings. This book also contains three extended abstracts by the plenary keynote speakers.

It is truly a pleasure to thank all those people who contributed to preparation of this book. In particular, we would like to express our appreciation for the work of the CS&P 2015 Program Committee members and external reviewers who helped to assure the high standards of accepted papers. We would like to thank all the authors of CS&P 2015, without whose high-quality contributions it would not have been possible to organize the workshop. We are grateful to the Organizing Committee members for their involvement in all the organizational matters related to the CS&P 2015 as well as the creation and maintenance of the conference website. We wish to express our thanks to Mikhail Moshkov, Andrzej Skowron and Louchka Popova-Zeugmann for accepting to be plenary speakers at CS&P 2015. We greatly appreciate the financial support received
from the University of Rzeszow, the University of Warsaw, and the Vistula University in Warsaw.

We hope that the CS&P 2015 workshop proceedings will serve as a valuable reference for researchers and developers in the field.

September 2015

Zbigniew Suraj
Ludwik Czaja
Organization

CS&P 2015 was organized by the Chair of Computer Science, the University of Rzeszow, Rzeszow, Poland, in cooperation with the Institute of Mathematics and the Institute of Informatics, the University of Warsaw, Warsaw, Poland, the Vistula University, Warsaw, Poland, the Warsaw Center of Mathematics and Computer Science, Warsaw, Poland, and the Institute of Informatics, the Humboldt University, Berlin, Germany.

CS&P 2015 Conference Committees

Program Committee

- Hans-Dieter Burkhard, Humboldt University, Berlin, Germany
- Ludwik Czaja, University of Warsaw, Warsaw, Poland and Vistula University, Warsaw, Poland
- Monika Heiner, Brandenburg University, Cottbus, Germany
- Anna Gomolinska, University of Bialystok, Bialystok, Poland
- Magdalena Kacprzak, Bialystok University of Technology, Bialystok, Poland
- Hung Son Nguyen, University of Warsaw, Warsaw, Poland
- Wojciech Penczek, Institute of Computer Science, PAS, Warsaw, Poland and University of Podlasie, Siedlce, Poland
- Lech Polkowski, University of Warmia and Mazury, Olsztyn, Poland
- Louchka Popova-Zeugmann, Humboldt University, Berlin, Germany
- Holger Schlingloff, Fraunhofer FIRST and Humboldt University, Berlin, Germany
- Serhat Seker, Vistula University, Warsaw, Poland
- Andrzej Skowron, University of Warsaw, Warsaw, Poland and Systems Research Institute, PAS, Warsaw, Poland
- Zbigniew Suraj, University of Rzeszow, Rzeszow, Poland
- Marcin Szczuka, University of Warsaw, Warsaw, Poland
- Matthias Werner, Technical University, Chemnitz, Germany
- Karsten Wolf, University of Rostock, Rostock, Germany

Organizing Committee

- Aneta Derkacz
- Katarzyna Garwol
- Piotr Grochowalski
- Piotr Lasek
- Lukasz Maciura
- Wieslaw Paja
- Krzysztof Pancerz
- Zbigniew Suraj
- Piotr Wasilewski
External Reviewers

- Michal Knapik
- Artur Meski
- Roman Redziejowski
- Anna Sawicka
- Bartłomiej Starosta
- Maciej Szreter
- Dominik Slezak
- Piotr Chrzastowski-Wachtel
# Table of Contents

Detecting Hazardous Events from Sequential Data with Multilayer Architectures .......................... 1  
*Karol Kurach, Krzysztof Pawlowski*

Instance-Level Constraints in Density-Based Clustering ....................................................... 11  
*Piotr Lasek*

Controlling Petri Net Behavior Using Time Constraints ..................................................... 19  
*Irina A. Lomazova, Louchka Popova-Zeugmann*

Evaluation of Decision Table Decomposition Using Dynamic Programming Classifiers .............. 34  
*Mieczyslaw Mankowski, Tadeusz Luba, Cezary Jankowski*

ExpTime Tableaux with Global Caching for Graded Propositional Dynamic Logic ...................... 44  
*Lech T. Polkowski*

Estimation and Feature Selection by Application of Knowledge Mined from Decision Rules Models .......................................................... 57  
*Wieslaw Paja, Krzysztof Pancerz*

Information Systems and Soft Sets ......................................................................................... 69  
*Zofia Machnicka, Marek Palasinski*

An Approach for Resolving Conflicts in Automatic Medical Objects Classification ............... 73  
*Przemyslaw W. Pardel, Jan G. Bazan, Jacek Zarychta, Stanislawa Bazan-Socha*

An Afficient Equivalence-Checking Algorithm for a Model of Programs with Commutative and Absorptive Statements .................................................. 85  
*Vladislav Podymov*

Betweenness, Lukasiewicz Rough Inclusions, Euclidean Representations in Information Systems, Hyper-granules, Conflict Resolution .................................. 97  
*Lech T. Polkowski*

Fuzzy Systems of Logical Inference and Their Applications .............................................. 111  
*Oleksandr Provotar*

Enhanced Error Correction Algorithm for RBF Neural Networks ..................................... 120  
*Pawel Rozycki, Janusz Kolbusz*

The Minimization Method of Boolean Functions in Polynomial Set-theoretical Format ............ 130  
*Bohdan Rytsar*
Detecting Hazardous Events from Sequential Data with Multilayer Architectures

Karol Kurach and Krzysztof Pawlowski

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{kkurach,kpawlowski236}@gmail.com

Abstract. Multivariate time series data play an important role in many domains, including real-time monitoring systems. In this paper, we focus on multilayer neural architectures that are capable of learning high level representations from raw data. This includes our previous solution based on Recurrent Neural Networks with Long Short-Term Memory (LSTM) cells. We build upon this work and present improved methods that aim to achieve higher prediction quality and better generalization to other similar tasks. We apply new deep neural architectures, minimize feature engineering and explore different ways of model selection. In particular, our focus on architectures includes networks with attention mechanism and convolutional networks. We tackle overfitting challenges in a presence of concept drift.

Key words: Time Series Forecasting, Recurrent Neural Networks, Ensemble Methods, Deep Learning

1 Introduction

Predicting future level of methane concentration in a coal mine is an important task, on which depends an efficiency of mining operation[12]. We focus on this supervised learning classification problem, which was also the topic of the IJCRS’15 Data Challenge: Mining Data from Coal Mines competition[6]. As a part of an earlier solution[8] we developed methods based on Deep Neural Networks, that achieved competitive results. However, a few opportunities for improvement were left. In this paper, we extend our previous work and develop improved methods. In particular, the main contributions of this paper are the following enhancements:

1. Minimal feature engineering that allows better generalization and easier application to other domains and tasks,
2. More advanced neural architectures to allow models with even greater learning capacity and higher prediction quality,
3. Explore different model selection methods in attempt to reduce overfitting, and in general investigate what techniques work best for problems with concept drift and highly correlated data at the same time.

* Both authors contributed equally.
The rest of this paper is organized as follows. In Section 2 we describe the problem. In Section 3 we describe our previous, “baseline” solution which is based on Deep Neural Networks. In Section 4 we present the improved methods. Finally, Section 5 concludes the paper and proposes the future work.

2 Problem Statement

We start this section describing the data used in the competition. Then, we document the evaluation procedure, including the target measure to be optimized. Finally, we review the most important challenges relevant to the task.

2.1 Data

The goal of IJCRS’15 Data Challenge competition was to predict dangerous level of methane concentration in coal mines based on the readings of 28 sensors. It is an example of supervised learning classification task. The data is split into training and test set, where the training set contains 51700 records and the test set contains 5076 records.

Each record is a collection of 28 time series – corresponding to 28 sensors that are installed in the coal mine. The sensors record data such as level of methane concentration, temperature, pressure, electric current of the equipment etc. Each of the time series contains 600 readings, taken every second, for a total of 10 minutes of the same time period for each sensor. That gives a total of 16800 features per record. The time periods described in the training data set overlap and are given in a chronological order. For the test data, however, the time periods do not overlap and are given in random order.

For each record in the training set, three labels are given. The test set is missing the labels – it is the goal of the competition to predict those values. Each label instance can be either normal or warning. Those levels signify the amount of methane concentration, as recorded by the three known sensors, named MM263, MM264 and MM256. The second-by-second readings of those sensors are described in time series mentioned in the previous paragraph. The predictions are to be made about the methane level in the future - that is during the period between three and six minutes after the end of the training (time series) period. If the level of methane concentration reaches or exceeds 1.0, then the corresponding label should be warning. Otherwise, it should be normal.

2.2 Evaluation

The submissions consist of three predictions of label values, made for each of 5076 records in the test set. Each prediction is a number – a higher value denotes a higher likelihood that the true label value is warning. The score is defined as a mean of area under the ROC curve, averaged over the three labels.

Participants may submit their predictions during the course of the competition. Until the finish of the competition, the participants are aware only of the score computed over preliminary test set – a subset of the whole test set that contains approximately 20% of the records. This subset is picked at random by the organizers and is fixed for all competitors but it is not revealed to the participants which of the test records belong to
it. The participants may choose a single final solution, possibly taking into the account the scores obtained on the preliminary test set. However, the final score is computed over the final test set – remaining approximately 80% of the test data. This score is revealed only after the end of the competition and is used to calculate the final standings – the team with the highest score is declared the winner.

2.3 Challenges

We describe the two main challenges one needs to overcome when tackling this problem.

**Overlapping training periods.** Almost all adjacent training records overlap by 9 out of the total 10 minutes recorded in the time series. It clearly violates the assumption of i.i.d. that underpins the theoretical justification of many learning algorithms. In addition, due to the overlap, a classical cross-validation approach may result in splits very „similar” data across different folds and in turn yield over-optimistic estimates of the model performances.

**Concept drift.** Training and test data come from different time periods. The records in the training set are sorted by time, so it’s easy to notice that there are very significant trends in the data that change along with the time. With test data samples taken at times belonging to a different interval than training samples, one can expect a severe concept drift - and indeed exploratory tests showed that classifier performance degrades on the test set, as compared to the same classifier’s performance when it is evaluated on the interval of training data that was not used for its learning.

3 Baseline Model

In this section we present a brief summary of our solution to IJCRS’15 competition[8]. Our method consist of the two main parts: the Recurrent Neural Network with Long Short-Term Memory cells (which we refer to as "LSTM" later in the paper) and Deep Feedforward Neural Network ("DFNN"). Intuitively, the LSTM processes the whole input sequence, taking the order of sensor readings into account, while DFNN operates on the last 30 readings. Finally, we ensemble the predictions to improve the performance.

3.1 Long Short-Term Memory Model

Recurrent Neural Network (RNN) is a type of artificial neural network in which dependencies between nodes form a directed cycle. This kind of network is particularly suited for modeling sequential data, where the length of the input is not fixed or can be very long.
Long Short-Term Memory is an RNN architecture designed to be better at storing and accessing information than standard RNN [5]. LSTM block contains memory cells that can remember a value for an arbitrary length of time and use it when needed. It also has a special *forget gate* that can erase the content of the memory when it is no longer useful. All the described components are built from differentiable functions and trained during back-propagation step. The connections in LSTM cells vary slightly between implementations. The variant that we used is presented in Figure 1.

![LSTM block diagram](image)

**Fig. 1.** A graphical representation of LSTM memory cells used in [16] and in our solution. The value $h_{t-1}^l$ represents the hidden state at the previous time step (same layer). The value $h_{t}^{l-1}$ represent the hidden state at the same time step (layer below).

**Architecture and training**  The network that we used was unfolded to 60 time steps and trained using back-propagation through time[14]. The sensor values go through the hidden layer, which in this case consist solely of LSTM cells. At time step $t \in 1, \ldots, 60$, the input for RNN are 28 average sensor values from seconds $((t-1) \ast 10, t \ast 10)$.

After processing the whole sequence, the last network’s hidden state encodes all sensor averages in the same order in which they were seen. On top of this we build a standard supervised classifier (Multi-Layer Perceptron in this case) that predicts the binary outcome. The *warning* class is assigned a value of 1.0 and *normal* class is assigned a value of 0.0. The loss function used in the final model was Mean Squared Error. It performed better than Binary Cross Entropy loss which is typically used for binary classification.
The training is done using standard Stochastic Gradient Descent. To improve the convergence of this algorithm, the data was normalized to mean 0 and variance 1. Also, the training data is randomly shuffled after every training epoch. We initialize all the parameters by sampling from uniform distribution. To avoid exploding gradient problem, the gradients are scaled globally during training, so that their norm is not greater than 1% of parameters’ norm. All models were trained using Torch[3] on a machine with a GPU card.

3.2 DFNN Model

Deep feedforward neural network (DFNN) is an artificial neural network with multiple layers of hidden neurons. One notable difference between DFNN and LSTM network (described in Section 3.1) is that DFNN architecture does not contain recursive connections – instead, every neuron of the previous layer is connected with every neuron of the next layer. We train the DFNN model with a backpropagation algorithm[13] that uses stochastic gradient descent (with mini-batch and momentum) as an optimization procedure to minimize the root mean squared error between numeric predictions and the target values. To avoid overfitting to the training set we use two regularization[4] methods: ad-hoc early stopping[9] and dropout[10].

Feature engineering For DFNN model we apply the following preprocessing steps:

1. scale the readings (separately for each sensor) to mean 0 and standard deviation 1
2. transform the values with \( x \rightarrow \log(1 + x) \) function,
3. compute mean and standard deviation for every sensor, taken over the last 30 readings (30-second period),
4. keep the last 20 readings for the sensor that corresponds to the target label,
5. discard all the original features.

Such preprocessing reduces the number of features from 16800 (28 * 600) to just 76 (28 * 2 + 20).

Training and parameter tuning For each target label we train a different DFNN model and tune its parameters independently. We perform model selection to optimize the performance on two sets. Initially, we use the validation set created from 20% of the original data and train on the remaining 80%. For the final submission, we use the preliminary test set. See Subsection 3.3 for the discussion of model selection challenges.

3.3 Ensemble

The Baseline Model is an ensemble of two submodels - LSTM described in Subsection 3.1 and DFNN described in Subsection 3.2. More precisely, the ensemble procedure computes rank for each of the submodels (independently) and then, for each record it takes the arithmetical average of the corresponding ranks as a final prediction. Table 1 illustrates the scores that particular models achieve on the preliminary test set.
<table>
<thead>
<tr>
<th>AUC score</th>
<th>label</th>
<th>MM263</th>
<th>MM264</th>
<th>MM256</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.9599</td>
<td>0.9560</td>
<td>0.9605</td>
<td></td>
</tr>
<tr>
<td>DFFN</td>
<td>-</td>
<td>0.9773</td>
<td>0.9602</td>
<td></td>
</tr>
<tr>
<td>ensemble</td>
<td>-</td>
<td>0.9722</td>
<td>0.9683</td>
<td></td>
</tr>
</tbody>
</table>

As the Baseline Model we combine the best-performing methods for each target label. That is, for label *MM263* we use LSTM, for label *MM264* we use DFNN and for label *MM256* we use the ensemble of LSTM and DFNN.

**subsection Results and challenges**

Baseline Model achieves the final score of 0.94 – recall that this score is computed on the final test set and revealed only after the end of competition. While the score on the final test set is good, we noticed that Baseline Model achieves a much better score of 0.9685 on the preliminary test set.

Such decrease in performance is probably caused by overfitting and rather easy to explain: we performed model selection based on the preliminary test scores. Therefore, it is not surprising that the very best model, as judged by its performance on the preliminary test set, does not achieve a similar performance on the final test set.

The cross-validation procedure is a standard Machine Learning methodology to deal with a danger of overfitting. We did not use it for optimizing the Baseline Model, because we observed a significant concept drift between the training and test set (as stated in Subsection 2.3). Our hypothesis was that the data more similar to the final test set would give more useful estimates of the model’s final performance.

During the contest, we did not test this hypothesis and the obvious challenge is to verify, and possibly to refute it. We address this challenge in Subsection 4.3.

**4 Improved Methods**

We improve the algorithm described in Section 3, particularly to address the challenges described in Subsection 3.3. To that end, we:

1. **Minimize feature engineering** for better generalization to other time series tasks and to make the improved model easier to apply,
2. **Introduce new architectures** to increase model’s capacity for learning,
3. **Improve model selection** to reduce the overfitting effect.

The rest of this section describes these improvements in detail.

**4.1 Minimize Feature Engineering**

Recall from Section 3 that some components of our Baseline Model required a significant feature engineering, particularly as described in Subsection 3.2. Such approach, while effective in practice, makes model less generalizable as the feature engineering
steps depend on the problem at hand. If one could reduce this process to minimum, it would be much easier to apply the methods to other multivariate time series problems. That is what we aim for. To that end, we limit feature engineering only to the following two operations:

- **Data normalization**, in regards to mean and standard deviation. This is a standard Machine Learning procedure, and as such it should be applicable to almost any problem. Without data normalization, and thus with data at different scales, it could be difficult to control the optimization procedure and the regularization.
- **Downsampling the data**. That is, replacing groups of adjacent values in the time series with their average. We do not optimize the downsampling granularity for the best possible score, it is only set to fit in the memory and decrease computing time.

### 4.2 New Architectures

In Section 3 we described a solution that was an ensemble of Recurrent Neural Network and Deep Feedforward Neural Network. We investigate how we could improve on this architecture and train a powerful model without the need of ensembling techniques. To this end, we propose several modifications to our baseline network. This includes changes to the LSTM training procedure, adding attention mechanism and applying convolutional layers.

**LSTM improvements** In the RNN network described in the Section 3.1, the target binary value is predicted from $h_n$ (the last hidden state). This architecture unfortunately has some drawbacks: there is only one signal at the end of the sequence, which can be distorted during long back-propagation. As a result, the model will not learn much from the data at the begin of the sequence. To address this problem, we modify the architecture to predict the target label at every time step (from every $h_i$).

Another improvement is related to the fact that the network had only one hidden layer. An easy modification is to train the network with $l \geq 2$ vertical layers. Such a network has a capacity to express several different transitions happening at one time step.

**Attention mechanism** One of the recently introduced improvements for LSTMs is the attention mechanism[2]. The general idea behind this modification is based on the observation that LSTMs process long sequences with a limited memory. After processing the whole input data, the LSTM needs to encode all of it in the internal memory cells. For longer sequences, it is impossible to do without some information loss. The attention mechanism solves this problem by allowing the network to automatically search for parts of the input that are relevant. There are two main types of attention: hard attention[1], which is based on Reinforcement Learning techniques and soft attention[15] that is differentiable and trained by back-propagation. In our work, we focus on the latter and add a similar mechanism to the network.
Convolutional networks Deep Neural Networks achieve state of the art performance in image recognition[7]. The best performing models are built using convolutional and subsampling layers[11]. The weight sharing in convolution allows the network to significantly reduce the number of parameters to train, and as a result detect features irrespective of their position in the image. Inspired by 2D convolutions used for images, we train a 1D convolutions for the time series data. This can be compared to training moving window "patterns" in the sequential data, and searching for them at the test time. One of the main advantages of this approach is simplicity of training, compared to more complex recurrent networks.

4.3 Better Model Selection

Recall from Subsection 3.3 that Baseline Model experiences quite heavy drop in score when it is applied to the final test set, from the results it achieved on the preliminary test set. It is caused by fact, that we performed model selection with preliminary test set – a classical example of overfitting. We believe that, given heavy distribution drift, such choice had some justification. Yet it could be beneficial to compare different schemes of model selection, which we describe below.

Standard k-fold cross-validation is probably one of the most common Machine Learning method. However, in our problem the data overlap by 9 out of 10 minutes, which makes adjacent record very similar. If each record is assigned to a random fold, as standard k-fold cross validation requires, the procedure could favor models that are overfit to the training data.

Deterministic cross-validation One improvement to cross-validation is by partitioning the records in such a way, that the number of overlapping instances assigned to different folds is minimized. To that end a procedure similar to k-fold cross-validation can be used. Instead of assigning records randomly to the folds, one sorts all the records chronologically and then divides them into equal, continuous, folds. For example, in 5-fold procedure, the first fold contains the fifth of the earliest records, the second fold contains the second fifth of the earliest records and so on.

Repeated deterministic cross-validation Another improvement to deterministic k-fold cross-validation could be to perform it on a fixed number of continuous intervals, repeated and together covering the whole set. That makes records belonging to the same fold more diverse and still does not assign too many overlapping records to different folds.

The partitioning of methods mentioned above are illustrated in Table 2.
Table 2. Illustration of cross-validation partitionings for 20 records (5 folds). Records are sorted chronologically. Different letters denote assignment to different folds

| c-v method                  | record number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|-----------------------------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| standard (randomized) 5-fold|               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |
| deterministic 5-fold        |               | A | A | A | B | B | B | C | C | C | D | D | D | D | E | E | E | E | E | E | E | E | E |
| 2-repeated deterministic 5-fold |             | A | A | B | B | C | C | D | D | D | D | E | E | E | E | E | E | E | E | E | E | E | E |

Rolling validation  In all methods described so far, future data can be used to train model that is then validated on past instances. Such inversion of chronology is not always desirable, especially if significant trends exist. In practical real-time application one could never predict the present based on the knowledge learned „from the future”. To implement this constraint, and possibly increase accuracy of estimations, one can partition the records in the same way as for deterministic cross-validation. However, only folds earlier (chronologically) than validation fold are allowed to take part in training the model. As the estimate of the measure, one simply takes average of the results from all \((k - 1)\) the runs. Table 3 demonstrates the partitioning.

Table 3. Illustration of rolling validation partitioning (5 folds). Records are sorted chronologically. „T” denotes that record is used for training; „V” denotes that record is used for validation; „-” denotes that record is not used at all

| run  | record number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|------|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| run 1 |               | T | T | T | T | T | V | V | V | V | - | - | - | - | - | - | - | - | - | - | - | - | - |
| run 2 |               | T | T | T | T | T | T | T | T | T | V | V | V | V | - | - | - | - | - | - | - | - | - | - |
| run 3 |               | T | T | T | T | T | T | T | T | T | T | T | T | V | V | V | V | - | - | - | - | - | - | - |
| run 4 |               | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | V | V | V | V | V | V | V | V |

5 Conclusion

The IJCRS’2015 Data Challenge competition was an opportunity to develop and test methods of solving multivariate time series problem that features a concept drift. Our first, baseline solution based on Deep Neural Networks achieved a competitive score of 0.94. We build upon it to further enhance the performance and generalization. The contribution of this paper is improved methods of approaching such tasks. In particular, we apply new deep neural architectures, minimize feature engineering and explore different ways of model selection. The immediate future work is to obtain precise results of the above methods and analyze them. The next step would be to evaluate how the described methods generalize to different datasets.
References

Instance-Level Constraints in Density-Based Clustering

Piotr Lasek
Chair of Computer Science,
University of Rzeszow
ul. Prof. Pigonia 1,
35-310 Rzeszow, Poland
lasek@ur.edu.pl

Abstract. Clustering data into meaningful groups is one of most important tasks of both artificial intelligence and data mining. In general, clustering methods are considered unsupervised. However, in recent years, so-named constraints become more popular as means of incorporating additional knowledge into clustering algorithms. Over the last years, a number of clustering algorithms employing different types of constraints have been proposed. In this paper we focus on instance level constraints such as must-link and cannot-link and present theoretical considerations on employing them in a well know density-based clustering algorithm – DBSCAN. Additionally we present a modified version of the algorithm using the discussed type of constraints.

Key words: data mining, data clustering, supervised clustering, clustering with constraints, instance-level constraints

1 Introduction

Clustering is one of well-known and often used data mining methods. Its goal is to assign data objects (or points) to different clusters so that objects that were assigned to the same clusters are more similar to each other than to objects assigned to another clusters [9].

Clustering algorithms can operate on different types of data sources such as databases, graphs, text, multimedia, or on any other datasets containing objects that could be described by a set of features or relationships [2]. Performing a clustering task over a dataset can lead to discovering unknown yet interesting and useful patterns or trends in the dataset. Since clustering algorithms do not require any external knowledge to be run (except simple algorithm’s parameters like k in the k-Means algorithm), the process of clustering, in opposite to classification, is also often referred to as an unsupervised learning. However, there has always been a natural need to incorporate already collected knowledge into algorithms to make them better both in terms of efficiency and quality of results. This need led to the emergence of a new branch of clustering algorithms based on so-named constraints. Constraint-based clustering algorithms employ the fact, that in many applications, the domain knowledge (e.g. in the form of labeled objects) is already known or could be easily specified by domain experts. Moreover, in some cases such knowledge can be automatically detected. Initially, researchers focused on algorithms that incorporated pairwise constraints on cluster membership or
learned distance metrics. Subsequent research was related to algorithms that used many different kinds of domain knowledge [5].

The major contribution of this work is the offering of a method of using instance-level constraints in the DBSCAN algorithm.

The paper is divided into six sections. In Section 1 we have given a brief introduction to clustering with constraints. Next, in Section 2, we shortly describe most known types of constraints and focus on instance-level constraints such as must-link and cannot-link constraints. Section 3 shortly presents reference works in the field of constrained clustering – especially those which are related to density-based clustering. In Section 4, the DBSCAN algorithm is reminded. Further, in Section 5 we present the proposed method. Conclusions are drawn and further research is briefly commented and discussed in Section 6.

2 Instance-Level Constraints

In constrained clustering algorithms background knowledge can be incorporated into algorithms by means of different types of so-named constraints. Over the years, different methods of using constraints in clustering algorithms have been developed [5]. Constraint-based methods proposed so far employ techniques such as modifying the clustering objective function including a penalty for satisfying specified constraints [6], clustering using conditional distributions in an auxiliary space, enforcing all constraints to be satisfied during clustering process [13] or determining clusters and constraints based on neighborhoods derived from already available labeled examples [1].

Several types of constraints are known. For example, instance constraints describing relations between objects, distance constraints such as inter-cluster δ-constraints as well as intra-cluster ϵ-constraints [2]. Nevertheless, the hard instance-level constraints seem to be most useful since the incorporation of just few constraints of this type can increase clustering accuracy as well as decrease runtime. In [12] authors introduced two kinds of instance-level constraints, namely: the must-link and cannot-link constraints. These constraints are simple but they have interesting properties. For example must-link constraints are symmetrical, reflexive and transitive, similarly to an equivalence relation: if two points, say \( p_0 \) and \( p_1 \), are in a must-link relationship (or, in other words, are connected by a must-link constraint), then these points should be assigned to the same cluster \( c \) by a clustering algorithm. On the other hand, if two points, say \( r_0 \) and \( r_1 \), are in a cannot-link relationship (or are separated by a cannot-link constraint), then these points must not be assigned to the same cluster \( c \).

3 Related Works

As mentioned before, in constrained clustering background knowledge can be incorporated into algorithms by means of different types of constraints. Through the years, different methods of using constraints in clustering algorithms have been developed [5]. Constraint-based methods proposed so far employ techniques such as modifying the clustering objective function including a penalty for satisfying specified constraints [6], clustering using conditional distributions in an auxiliary space, enforcing all constraints
to be satisfied during clustering process [13] or determining clusters and constraints based on neighborhoods derived from already available labelled examples [1]. On the other hand, in the distance based methods, the distance measure is designed so that it satisfies given constraints [10, 4]. It is worth mentioning that among constrained algorithms proposed so far, there are only a few constraint-based representatives from the interesting for us group of density based algorithms, namely: C-DBSCAN [11], DBCOM [7] or DBCluC [14].

C-DBSCAN is an example of a density-based algorithm using instance-level constraints. In C-DBSCAN [4] two types of constraints are supported, namely must-link and cannot-link constraints. In the first step, the algorithm partitions the dataset using the KD-Tree [3]. Next, under cannot-link constraints, so-named local clusters are created by means of density-reachable points. In case there is a cannot-link constraint between points in the same leaf node of the KD-Tree, then each point in a currently traversed leaf is labelled as a singleton local cluster. Otherwise, every point \( p \) from a leaf is checked, whether \( p \) is a core point or not. If it is not, in other words, the number of points in an \( \epsilon \)-neighborhood of \( p \) is less than the value of \( \text{MinPts} \), then \( p \) is labelled as a NOISE and is ignored. If it is, then all points that are density-reachable from \( p \) are assigned to the same local cluster. Step 3a of the algorithm is designed for enforcing must-link constraints. If points connected by must-link constraints were assigned to different local clusters, then such clusters are merged into, so-named, a core local cluster. Step 3b was designed for further merging of the local clusters in order to enforce cannot-link constraints that have not been satisfied yet. Cluster merging is done by means of hierarchical agglomerative clustering with single linkage. For each pair of candidate clusters to be merged, it is checked whether they contain points that could be found in a set of cannot-link constraints. If they are involved in a cannot-link constraint then the clusters cannot be merged. The steps of the algorithm are stopped if the number of clusters does not change any more.

DBCluC [14] which was also based on the known DBSCAN algorithm [8] employs an obstacle modelling approach for density-based clustering of large two-dimensional datasets. By means of modelling of obstacles which improves the efficiency of clustering it is also capable of detecting clusters of arbitrary shape and is not sensitive to the order of points in a dataset as well as to the obstacle constraints and noise. The efficiency of clustering is leveraged by a reduction of polygons modelling the obstacles - the algorithm simply removes unnecessary edges from the polygons making the clustering faster in terms of a number of constraints to be analysed.

The DBCOM algorithm [7] pre-processes an input dataset so that it considers the presence of physical obstacles by modelling them - similarly to DBCluC. It can detect clusters of arbitrary shapes and size and is also considered to be insensitive to noise as well as an order of points in a dataset. The obstacles are modelled by representing them as simple polygons and it uses a polygon edge reduction algorithm so that the number of edges used to test visibility between points in space could be reduced in order to improve the efficiency of DBCOM so the results reported by authors of the algorithm confirm that it can perform polygon reduction faster than DBCluC. The algorithm comprises of three steps: first, it reduces the obstacles by employing the above mentioned
Table 1. The notations related to instance-level constraints used in this paper and auxiliary variables and notations used in pseudo-code of the algorithm

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C = )</td>
<td>The set of pairs of points that are in a \textit{must-link} relation.</td>
</tr>
<tr>
<td>( c = (p_0, p_1) )</td>
<td>Two points ( p_0 ) and ( p_1 ) are in a \textit{must-link} relation (must be assigned to the same resulting cluster).</td>
</tr>
<tr>
<td>( C = (p) )</td>
<td>The set of points which are in a \textit{must-link} relation with point ( p ).</td>
</tr>
<tr>
<td>( C \neq )</td>
<td>The set of pairs of points that are in a \textit{cannot-link} relation.</td>
</tr>
<tr>
<td>( c \neq (r_0, r_1) )</td>
<td>Two points ( r_0 ) and ( r_1 ) are in a \textit{cannot-link} relation (must not be assigned to the same resulting cluster).</td>
</tr>
<tr>
<td>( C \neq (r) )</td>
<td>The set of points which are in a \textit{cannot-link} relation with point ( r ).</td>
</tr>
<tr>
<td>( ClusterId )</td>
<td>The auxiliary integer variable used for storing currently-created cluster’s identifier.</td>
</tr>
<tr>
<td>( p.ClusterId )</td>
<td>By using such a notation we refer to a ( ClusterId ) related to point ( p ).</td>
</tr>
<tr>
<td>( p.ndf )</td>
<td>Such a notation is used to refer to a value of the NDF factor associated with point ( p ).</td>
</tr>
<tr>
<td>( R_d, R_t )</td>
<td>The auxiliary variables for storing deferred points.</td>
</tr>
<tr>
<td>( DPSet )</td>
<td>The variable for storing dense points. It is used for in an iterative process of assigning points to clusters.</td>
</tr>
</tbody>
</table>

edge reduction method, then performs the clustering and finally applies hierarchical clustering on formed clusters.

4 Density-Based Clustering

The \textit{DBSCAN} algorithm is a well known density-based clustering algorithm. Below we remind the key definitions related to the \textit{DBSCAN} algorithm which will be used in the sequel. Notations and their descriptions are given in Table 1.

\textbf{Definition 1} (\textit{\( \epsilon \)-neighborhood, or \( \epsilon NN(p) \) of point \( p \)}. \( \epsilon \)-neighborhood of point \( p \) is the set of all points \( q \) in dataset \( D \) that are distant from \( p \) by no more than \( \epsilon \); that is, \( \epsilon NN(p) = \{q \in D | dist(p, q) \leq \epsilon \} \), where \( dist \) is a distance function.

\textbf{Definition 2} (\textit{directly density-reachable points}). Point \( p \) is directly density reachable (Figure 1) from point \( q \) with respect to \( \epsilon \) and \textit{MinPts} if the following two conditions are satisfied:

a) \( p \in \epsilon NN(q) \)
**Fig. 1.** $p_0$ is directly density-reachable from core point $p_1$; $p_0$ is density-reachable from $p_2$ ($\text{MinPts} = 6$).

**Fig. 2.** Both $p_0$ and $p_5$ are density-reachable from core point $p_2$, so $p_0$ and $p_5$ belong to $C(p_2)$ ($\text{MinPts} = 6$).

1) $q$ is a core point.

**Definition 3 (density-reachable points).** Point $p$ is density-reachable from a point $q$ with respect to epsilon and $\text{MinPts}$) if there is a sequence of points $p_1, \ldots, p_n$ such that $p_1 = q$, $p_n = p$ and $p_i + 1$ and is directly density-reachable from $p_i$, $i = 1 \ldots n - 1$. (Figure 2)

**Definition 4 (a border point).** Point $p$ is a border point if it is not a core point and is density-reachable from a core point.

**Definition 5 (cluster).** A cluster is a non-empty set of points in $D$ which are density-reachable from a same core point.

**Definition 6 (noise).** Noise is the set of all points in $D$ that are not density-reachable from any core point.

Firstly, the algorithm generates a label for the first cluster to be found. Next, the points in $D$ are read. The value of the $\text{ClusterId}$ attribute of the first point read is equal to UNCLASSIFIED. While the algorithm analyzes point after point, it may happen that the $\text{ClusterId}$ attributes of some points may change before these points are actually analyzed. Such a case may occur when a point is density-reachable from a core point examined earlier. Such density-reachable points will be assigned to the cluster of a core point and will not be analyzed later. If a currently analyzed point $p$ has not been classified yet (the value of its $\text{ClusterId}$ attribute is equal to UNCLASSIFIED), then the $\text{ExpandCluster}$ function is called for this point. If $p$ is a core point, then all points in $C(p)$ are assigned by the $\text{ExpandCluster}$ function to the cluster with a label equal to the currently created cluster’s label. Next, a new cluster label is generated by $\text{DBSCAN}$. Otherwise, if $p$ is not a core point, the attribute $\text{ClusterId}$ of point $p$ is set to NOISE, which means that point $p$ will be tentatively treated as noise. After analyzing all points in $D$, each point’s attribute $\text{ClusterId}$ stores a respective cluster label or its value is equal to NOISE. In other words, $D$ contains only points which have been assigned to particular clusters or are noise.
5 Clustering with Instance Constraints

In this section we present how we adapted instance constraints in DBSCAN.

In our proposal, we introduce another type of points, namely the deferred points (Definition 7). Below we present the definition of deferred point as well as modified definitions of cluster and noise - Definition 9 and Definition 10, respectively.

Definition 7 (deferred point). A point $p$ is deferred if it is involved in a cannot-link relationship with any other point or it belongs to a $\epsilon$–neighborhood $\epsilon\text{NN}(q)$, where $q$ is any point involved in a cannot-link relationship.

Definition 8 (parent point). A parent point of a given point $p$ is the point $q$ which is involved in a cannot-link relationship with any other point and $p$ is located withing $q$’s $\epsilon\text{NN}(q)$.

Definition 9 (cluster). A cluster is a maximal non-empty subset of $D$ such that:

– for two non-deferred points $p$ and $q$ in the cluster, $p$ and $q$ are neighborhood-based density-reachable from a local core point with respect to $k$, and if $p$ belongs to cluster $C$ and $q$ is also neighborhood-based density connected with $p$ with respect to $k$, then $q$ belongs to $C$;
– a deferred point $p$ is assigned to a cluster $c$ if the nearest neighbour of $p$ belongs to $c$, otherwise $p$ is considered as a noise point.

Definition 10 (noise). The noise is the set of all points in $D$ that:

– are not density-reachable from any core point or
– being a deferred point had two or more neighbours located at the same distance and thus could not be unambiguously assigned to a cluster.

Our method works so that the DBSCAN algorithm (Figure 3) in the phase of preparation omits all points which are involved in any cannot-link relationship and marks them as DEFERRED. Then, it adds those points to an auxiliary list called $R_d$ which will be later used in the main loop of the algorithm and the AssignDeferredPoints function.

Then the algorithm iterates through all UNCLASSIFIED points from $D$ except those which were added to $R_d$. For all of those points it calls the ExpandCluster function (Figure 5) and passes all necessary parameters. The main modifications in this function concern how points involved in must-link relationships are processed. Basically, if such a point is found and it is a core point or belongs to a neighbourhood of a points which is a core point, then it is assigned to seeds or curSeeds lists depending on which part of the ExpandCluster function is being executed.

The last part of the algorithm is to process DEFERRED points. This is done by means of the AssignDeferredPoints function (Figure 4). This function performs so that for each point $q$ from $R_d$ (a list of points which were marked as DEFERRED in the main algorithm method) it determines what would be the nearest cluster of that point ($g_p$). Additionally it analyzes cannot-link points connected from the currently processed area (an $\epsilon$–neighbourhood of $p$) so that it checks whether the parent point of $q (p)$ (which nearest cluster is $g_p$) and in is a cannot-link relationship with another parent point ($p_{\neq}$) was assigned to the same cluster ($g_{p_{\neq}}$) If so, then the point $q$ is marked as NOISE. Otherwise, $q$ is assigned to cluster $g_p$. 
Algorithm DBSCAN(D, k, C=, C̸ =)
1. Rd = ∅
2. label all points in D as UNCLASSIFIED;
3. ClusterId = label of a first cluster;
4. for each point q involved in any constraint from C= do
   5. label q and points in ϵNN(q) as DELETED;
   6. endfor;
7. add all DELETED points to Rd;
8. for each point p in set D, Rd do
   9. if (p.ClusterId = UNCLASSIFIED) then
     10. if ExpandCluster(D, p, ClusterId, ϵ, MinPts) then
         11. ClusterId = NextId(ClusterId);
         12. endif;
     13. endif;
     14. endfor;
9. AssignDefferedPoints(D, p, ClId, MinPts, ϵ, C=, C̸ =);
10. endfor;

Fig. 3. The DBSCAN algorithm.

Function AssignDefferedPoints(D, Rd, ClId)
1. for each point q ∈ Rd do
2. p ← GetParent(q);
3. gp ← NearestCluster(p);
4. gq ← NearestCluster(q);
5. if gq ̸= gp̸ = then
6. mark q as NOISE;
7. else if
8. assign point q to gq;
9. endif;
10. remove q from Rd;
11. end if;
12. end for;

Fig. 4. Assigning deferred points to clusters.

Fig. 5. The ExpandCluster function.

Function ExpandCluster(D, p, ClId, MinPts, ϵ, C=, C̸ =)
1. seeds = Neighborhood(D, p, ϵ);
2. p.ClusterId = UNCLASSIFIED;
3. return FALSE;
4. while (seeds > 0 do
5. curPoint = first point in seeds;
6. curSeeds = Neighborhood(D, curPoint , ϵ);
7. if |curSeeds| ≥ MinPts then
8. for each point q in curSeeds do
9. add ClId(q) to seeds;
10. end for;
11. end if;
12. end while;
13. if curPoint ̸= ClId then
14. append q to seeds;
15. else if curPoint = UNCLASSIFIED then
16. q.ClusterId = ClId;
17. end if;
18. end if;
19. delete curPoint from seeds;
20. end if;
21. end for;
22. end while;
23. return TRUE;

Fig. 6. The pseudo-code of the DBSCAN algorithm using instance constraints.

6 Conclusions and Further Research

Must-link and cannot-link constraints are supposed to work so that points connected by must-link constraints are assigned to the same clusters and points which are in cannot-link relationships cannot be assigned to the same clusters. Obviously such constraints may lead to many problems, not to mention the fact that some constraints may be contradictory. Our approach slightly loosens both must-link and cannot-link constraints by prioritizing them. The intuition is that must-link constraint are more important than cannot-link constraints which gives us the advantage when trying to fulfill all constraint. We try to fulfill all must-link constraints first (assuming of course that all of them are valid) treating them as more important than cannot-link constraints. When processing cannot-link constraints, points which are contradictory (in terms of fulfilling both must-link and cannot-link constraints are intuitively marked as noise.

In this paper we presented a method of adapting the DBSCAN algorithm to work with instance constraints. Our future works will focus on analyzing the complexity of the proposed method as well as on testing its performance compared to other constrained clustering algorithms.
References

Controlling Petri Net Behavior Using Time Constraints

Irina A. Lomazova1* and Louchka Popova-Zeugmann2

1 National Research University Higher School of Economics (HSE), Moscow, 101000, Russia
   ilomazova@hse.ru
2 Department of Computer Science, Humboldt University Berlin, Germany
   popova@informatik.hu-berlin.de

Abstract. In this paper we study how it is possible to control Petri net behavior using time constrains. Controlling here means forcing a process to behave in a stable way by associating time intervals to transitions and hence transforming a classic Petri net into a Time Petri net. For Petri net models stability is often ensured by liveness and boundedness. These properties are crucial in many application areas, e.g. workflow modeling, embedded systems design, and bioinformatics. This paper deals with the problem of transforming a given live, but unbounded Petri net into a live and bounded one by adding time constraints. We specify necessary conditions for the solvability of this problem and present an algorithm for adding time intervals to net transitions in such a way that the resulting net becomes bounded while staying live.

1 Introduction

Distributed systems range in almost all areas: from technical systems to biological systems or to systems of business processes. Although such systems are very different in their subject matter they all have common properties, such as reiteration of all subprocesses or returning to some initialization in the system, or containing finitely or infinitely many different states etc. Petri nets are widely used for modeling and analysis of distributed systems. The first two properties concern the liveness of the model, the second two are the subject of boundedness studies. In most of the practical systems the infiniteness of all reachable states is an undesired property.

A typical example is a business process model, represented by a workflow net — a special kind of a Petri net. The essential property for workflow nets is soundness, also called proper termination [1]. Soundness is intensively studied in the literature [1, 2, 7, 11, 13, 16]. Checking soundness of workflow nets can be reduced to checking liveness and boundedness for the extended net obtained by connecting the source place with the sink place through a new transition in the initial workflow net. Thus ensuring liveness and boundedness of a model can be applied for asserting soundness of workflow nets. In biological systems liveness and boundedness ensure system stability [8, 9]. In embedded systems scheduling is often necessary due to the resource limitations [10].

In practice it may often happen that a given live Petri net is not bounded. Then it would be helpful to “repair” the model by adding priorities or time to transitions so, that

* This study was supported by the National Research University Higher School of Economics’ Academic Fund.
the net becomes bounded staying live. In other words, the question is whether we can "repair" the model with the help of priority or time constrains. In [5] J. Desel proposed an approach for a brute-force-scheduling to ensure bounded behavior, employing transitions of a given subset infinitely often. Here we study when and how time constraints can ensure boundedness of a given live Petri net, retaining its liveness. In contrast to brute-force approach, time constraints allow local and more flexible control — not just forcing one 'good' execution.

In [12] we have considered a way to add priorities in order to transform a live and unbounded model, represented by a Petri nets, into a live and bounded one. For this reason we have defined a sub-tree of the reachability tree, the so-called "spine tree". The spine tree contains all minimal cyclic runs together with prefixes leading to the cycles. In this paper we use the spine-based coverability tree, derived from the spine tree, in order to add time intervals to the transitions in such a way that the resulting Time Petri net is live and bounded. Of course, this is not always possible.

Priority scheduling is an appropriate solution for workflow systems. For biological systems it is not so good, since there is no mechanism to assign priorities to events in biological systems. Scheduling biological systems with the help of time constraints would provide a much more natural solution [18].

In this work we show how to associate time constrains to a given live and unbounded Petri net in such a way that the resulting time-dependent net is live and bounded. Thereby we want to fully preserve the structure of the given net. For that we use the spine tree and the spine-coverability tree, introduced in [12] and compute a parametrical state space for a Time Petri net, for which the underlying timeless Petri net is the given one.

The paper is organized as follows. In Section 2 we first give a more detailed motivation for this work and then recall some basic definitions in the theory of Petri nets. Section 3 presents the main contributions of the paper: a sufficient condition for transforming a live and unbounded Petri net into a live and bounded one by adding time intervals and an algorithm for computing these intervals, as well as, an example illustrating the algorithm. Section 5 contains some conclusions.

2 Preliminaries

2.1 Motivation

Liveness in bounded Petri nets is considered in numerous works. In [19] Ridder and Lautenbach considered the relationship between liveness in bounded nets and T-invariants. For marked graph Petri nets characterization of liveness and boundedness in terms of reachability graphs was done in [3]. Schedulability analysis of Petri nets, aimed at ensuring infinitely repeated firing sequences within a bounded stated space, was studied in [10].

In [9] and [8] M. Heiner considered the problem of transforming live and unbounded Petri nets into live and bounded nets by adding time durations to transitions. It is shown in these works, that when a Petri net is covered by T-invariants (i.e. each transition enters into at least one T-invariant with a non-zero component), T-invariants can be used for
computing time durations for transitions, making the net bounded. In other words, this method allows to transform a live and unbounded Petri net, covered by T-invariants, into a live and bounded Timed Petri net [18] with the same structure. Unfortunately this method does not always work, as it was shown in [9].

Furthermore, as it is shown in [12], a possibility to transform a live and unbounded net into a bounded one can depend not only on T-invariants, but on initial markings as well. So, the algorithm for making a live Petri net also unbounded with the help of transition priorities, which we presented in [12], essentially takes into account initial states. This is the case also for the algorithm assigning time intervals to the transitions and represented in this article.

Live and unbounded Petri nets were considered also in [6]. The notion of weak boundedness was introduced there. A Petri net \( \mathcal{N} \) is called weakly bounded, iff it is unbounded, but for every reachable marking \( m \) in \( \mathcal{N} \) a bounded run is enabled in \( m \), i.e. from every reachable marking we can find a way to continue the execution in such a way, that the number of tokens in each place will be not greater that some fixed value. The distinction between bounded, weakly bounded and not weakly bounded Petri nets is very important for applications. However, till now, there is no algorithm for distinguishing weakly bounded and not weakly bounded Petri nets. There is a reason to believe that these notions are connected with a possibility to transform an unbounded Petri net into a bounded one by adding some time, or priority constraints.

2.2 Basics

Let \( \mathbb{N} \) denote the set of natural numbers (including zero) and let \( \mathbb{Q}_0^+ \) be the set of all non-negative rational numbers including zero. All notions and notations used here are generally known and can be found in [4].

Let \( P \) and \( T \) be disjoint sets of places and transitions with \( P \cup T \neq \emptyset \) and let \( F \subseteq (P \times T) \cup (T \times P) \rightarrow \mathbb{N} \) be a flow relation. Then \( \mathcal{N} = (P, T, F) \) is a (unmarked) Petri net. A marking in a Petri net is a function \( m : P \rightarrow \mathbb{N} \), mapping each place to some natural number (possibly zero). A (marked) Petri net \( (\mathcal{N}, m_0) \) is an unmarked Petri net \( \mathcal{N} \) with its initial marking \( m_0 \). Further we call marked Petri nets just Petri nets and use vector notation for marking by fixing some ordering of places in a Petri net.

Pictorially, \( P \)-elements are represented by circles, \( T \)-elements by boxes, and the flow relation \( F \) by directed arcs. Places may carry tokens represented by filled circles. A current marking \( m \) is designated by putting \( m(p) \) tokens into each place \( p \in P \).

For a transition \( t \in T \) an arc \((x, t)\) is called an input arc, and an arc \((t, x)\) – an output arc.

A transition \( t \in T \) is enabled in a marking \( m \) iff \( \forall p \in P \ m(p) \geq F(p, t) \). An enabled transition \( t \) may fire yielding a new marking \( m' \), such that \( m'(p) = m(p) - F(p, t) \) for each \( p \in P \) (denoted \( m \xrightarrow{t} m' \), or just \( m \rightarrow m' \)). Then we say that a marking \( m' \) is directly reachable from a marking \( m \).

A marking \( m \) is called dead iff it enables no transition.

A run in \( \mathcal{N} \) is a finite or infinite sequence of firings \( m_1 \xrightarrow{t_1} m_2 \xrightarrow{t_2} \ldots \). An initial run in \((\mathcal{N}, m_0)\) is a run, starting from the initial marking \( m_0 \). A cyclic run is a finite
run starting and ending at the same marking. A maximal run is either infinite, or ends with a dead marking.

We say that a marking \( m \) is reachable from a marking \( m' \) in \( N \) iff there is a run \( m' = m_1 \rightarrow m_2 \rightarrow \cdots \rightarrow m_n = m \); \( m \) is reachable in \( (N, m_0) \) iff \( m \) is reachable from the initial marking. By \( R(N, m) \) we denote the set of all markings reachable in \( N \) from the marking \( m \). A run \( \sigma \) in \( (N, m) \) is called feasible iff \( \sigma \) starts from a reachable marking.

A T-invariant in a Petri net with \( n \) transition \( t_1, \ldots, t_n \) is an \( n \)-dimensional vector \( \alpha = (\alpha_1, \ldots, \alpha_n) \) with \( \alpha_i \in \text{Nat} \) such that after firing of every transition sequence containing exactly \( \alpha_i \) occurrences of each transition \( t_i \) in an arbitrary marking \( m \) (if possible) leads to the same marking \( m \).

A reachability graph of a Petri net \((N, m_0)\) presents detailed information about the net behavior. It is a labeled directed graph, where vertices are reachable markings in \((N, m_0)\), and an arc labeled by a transition \( t \) leads from a vertex \( v \), corresponding to a marking \( m \), to a vertex \( v' \), corresponding to a marking \( m' \) iff \( m \rightarrow t \in N \) in \( N \).

A reachability graph may be also represented in the form of a reachability tree, which can be defined in a constructive form. We start from the initial marking as a root. If for a current leaf \( v \) labeled with a marking \( m \), there is already a node \( v' \neq v \) lying on the path from the root to \( v \) and labeled with the same marking \( m \), we notify \( v \) to be a leaf in the reachability tree. If not, nodes directly reachable from \( m \) and the corresponding arcs are added. Note, that in a reachability tree run cycles are represented by finite paths from nodes to leaves.

A place \( p \) in a Petri net is called bounded iff for every reachable marking the number of tokens residing in \( p \) does not exceed some fixed bound \( \kappa \in \text{Nat} \). A marked Petri net is bounded iff all its places are bounded.

It is easy to see, that a Petri net \((N, m_0)\) is bounded iff its reachability set \( R(N, m_0) \), and hence its reachability graph, are finite.

A marking \( m' \) covers a marking \( m \) (denoted \( m' \geq m \)) iff for each place \( p \in P \), \( m'(p) \geq m(p) \). The relation \( \geq \) is a partial ordering on markings in \( N \). By the firing rule for Petri net, if a sequence of transitions is enabled in a marking \( m \), and \( m' \geq m \), then this sequence of transitions is also enabled also in \( m' \). A marking \( m' \) strictly covers a marking \( m \) (denoted \( m' > m \)) iff \( m' \geq m \) and \( m \neq m' \) redundantly.

For an unbounded Petri net, a coverability tree gives a partial information about the net behavior. It uses the notion of a generalized marking, where the special symbol \( \omega \) designates an arbitrary number of tokens in a place. Formally, a generalized marking is a mapping \( m : P \rightarrow \text{Nat} \cup \{\omega\} \). A coverability tree is defined constructively. It is started from the initial marking and is successively constructed as a reachability tree.

The difference is that when a marking \( m' \) of a current leaf \( v' \) in a reachability tree strictly covers a marking \( m \) of a node \( v \), lying on the path from the root to \( v' \), then in a coverability tree the node \( v' \) obtains a marking \( m' \), where \( m'_\omega(p) = \omega \), if \( m'(p) > m(p) \), and \( m'_\omega(p) = m'(p) \), if \( m'(p) = m(p) \). For generalized markings enabling of a transition and a firing rule is defined as for usual markings except that \( \omega \)-marked places are ignored. Each place \( p \), which was marked by \( \omega \), remains \( \omega \)-marked for all possible run continuations.
Let $\mathcal{N} = (P, T, F)$ be an unmarked Petri net and let $I : T \rightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ be a function such that for each $t \in T$ holds: $I(t) = (a_t, b_t)$ and $a_t \leq b_t$. Thus, the function $I$ associates an interval $[a_t, b_t]$ with each transition $t$ in $T$. We note $a_t$ with $\text{eft}(t)$ (earliest firing time for $t$) and $b_t$ with $\text{lft}(t)$ (latest firing time for $t$).

Here $a_t$ and $b_t$ are relative to the time, when $t$ was enabled last. When $t$ becomes enabled, it can not fire before $a_t$ time units have elapsed, and it has to fire not later than $b_t$ time units, unless $t$ got disabled in between by the firing of another transition. The firing itself of a transition takes no time. The time interval is designed by real numbers, but the interval bounds are nonnegative rational numbers. It is easy to see (cf. [18]) that w.l.o.g. the interval bounds can be considered as integers only. Thus, the interval bounds $a_t$ and $b_t$ of any transition $t$ are natural numbers, including zero and $a_t \leq b_t$ or $b_t = \infty$. A comprehensive introduction can be found in [18].

$Z = (\mathcal{N}, m_0, I)$ is called Time Petri net (TPN) and it was first introduced by Merlin [14]. The marked Petri net $(\mathcal{N}, m_0) := S(Z)$ is called the skeleton and $I$ - the interval function of $Z$.

Every possible situation in a given TPN can be described completely by a state $z = (m, h)$, consisting of a (place) marking $m$ and a transition marking $h$. The (place) marking, which is a place vector (i.e. the vector has as many components as places in the considered TPN), is defined as the marking notion in classic Petri nets. The time marking, which is a transition vector (i.e. the vector has as many components as transitions in the considered TPN), describes the time circumstances in the considered situation. In general, each TPN has infinite number of states.

The state space of a TPN can be characterized parametrically and it is shown that knowledge of the integer-states, i.e. states whose time markings are (nonnegative) integers, is sufficient to determine the entire behavior of the net at any point in time (cf. [17] and [18]). Thus, a reachability graph for a TPN can be defined so that the nodes of the graph are the reachable integer-states and a directed edge connects two nodes, from $z_1$ to $z_2$ if there is possible to change from $z_1$ to $z_2$, considered as states in the TPN. And finally, a reachability tree can be then also defined for TPN considering the reachable $p$-markings.

**Fig. 1.** The Time Petri net $Z$
In this paper we use the parametric states in order to restrict the behavior of a live and unbounded PN to a live and bounded one. The notions parametric state and parametric run can be easily defined by recursion. Let $Z = (P, T, F, V, m_0, I)$ be a Time Petri net and let $\sigma = t_1 \cdots t_n$ be a firing sequence in $Z$. Then, the parametric run $(\sigma(x), B_\sigma)$ of $\sigma$ in $Z$ with $\sigma(x) = x_0 t_1 x_1 \cdots x_{n-1} t_n x_n$ and the parametric state $(z_\sigma, B_\sigma)$ in $Z$ are recursively defined as follows:

**Basis:** $\sigma = \varepsilon$, i.e., $\sigma(x) = x_0$.

Then $z_\sigma = (m_\sigma, h_\sigma)$ and $B_\sigma$ are defined as follows:

1. $m_\sigma := m_0$;
2. $h_\sigma(t) := \left\{ \begin{array}{ll} x_0 & \text{if } t^- \leq m_\sigma \\ \# & \text{otherwise} \end{array} \right.$;
3. $B_\sigma := \{ 0 \leq h_\sigma(t) \leq \text{lft}(t) \mid t \in T \land t^- \leq m_\sigma \}$

**Step:** Assume that $z_\sigma$ and $B_\sigma$ are already defined for the sequence $\sigma = t_1 \cdots t_n$.

For $\sigma = t_1 \cdots t_n$, $t_{n+1} = w t_{n+1}$ we set

1. $m_\sigma := m_w + \Delta t_{n+1}$;
2. $h_\sigma(t) := \left\{ \begin{array}{ll} \# & \text{if } t^- \leq m_\sigma \\ h_w(t) + x_{n+1} & \text{if } t^- \leq m_\sigma \land t^- \leq m_w \land t_{n+1} \cap t = \emptyset \\ x_{n+1} & \text{otherwise} \end{array} \right.$;
3. $B_\sigma := B_w \cup \{ \text{eft}(t_{n+1}) \leq h_w(t_{n+1}) \} \cup \{ 0 \leq h_\sigma(t) \leq \text{lft}(t) \mid t \in T \land t^- \leq m_\sigma \}$.

A short example should illustrate the calculation of parametric states. We use $K_\sigma$ as a shorthand for $\{ z_\sigma \mid B_\sigma \}$.

Let us consider the Time Petri net $Z$ in Fig. 1.

It is easy to see that

$$K_\varepsilon = \{(0,1,1),(x_0,\#,\#,x_0)\mid \{0 \leq x_0 \leq 3\}\}.$$  

After firing the sequence $\sigma = t_4$ the net $Z_2$ is in a state belonging to $K_\sigma = K_{t_4}$.

$$K_{t_4} = \{(1,1,0),(x_0 + x_1,\#,x_1,\#)\mid \{2 \leq x_0 \leq 3, x_0 + x_1 \leq 5, 0 \leq x_1 \leq 4\}\}.$$  

The set of conditions $B_{t_4}$ is the union of the three sets

$$B_{t_4} = \{ \text{eft}(t_4) \leq h_{t_4}(t_4) \} = \{ 2 \leq x_0 \} \text{ and } \{ 0 \leq h_\sigma(t) \leq \text{lft}(t) \mid t^- \leq m_\sigma \} = \begin{cases} 0 \leq x_0 \leq 5, \\ 0 \leq x_1 \leq 4 \end{cases}.$$  

By repeatedly firing the transitions $t_3$ and $t_4$ we obtain the parametric states $z_{t_3 t_4}$ and $z_{t_4 t_3}$ and $K_{t_3 t_4}$ and $K_{t_4 t_3}$.
the set $B$ of inequalities in $K$terval function $I$ some $T$-invariant. Let Proposition 1.

Let $(N, m_0)$ be a live and unbounded Petri net. We would like to check, whether it is possible to make this net bounded, not losing its liveness, by transforming it into a Time Petri net (with the same skeleton), i.e. by adding intervals to its transitions. To solve this problem we will associate transition intervals (if possible), which would exclude runs leading to unboundedness.

We start by recalling some properties of live and bounded Petri nets considered in our article [12]. Then, instead to assign priorities to the transitions we add intervals. We will show that this time solution is more precise than the solution with priorities. In this connection “more precise” means that the set of reachable markings in the time-dependent net (sometimes properly) covers the set of reachable markings in the net with priorities.

It is clear that if a live PN is bounded then there exists a feasible cyclic run, including all transitions of the PN. Furthermore, a TPN is obtained from a PN by adding time interval to each transition. Then the reachability tree is a subgraph of the reachability tree of the PN. Hence, it is obvious that if for some interval function $I$ the TPN $Z = (N, m_0, I)$ is live and bounded, then there exists a feasible cyclic run in $(N, m_0)$ which includes all transitions of $N$.

**Proposition 1.** Let $(N, m_0)$ be a live and unbounded Petri net. If there exists an interval function $I$ such that the TPN $(N, I, m_0)$ is live and bounded, then there exists a $T$-invariant without zero components for $N$, i.e. all transitions in $N$ are covered by some $T$-invariant.

As we already set in [12], given a live and unbounded Petri net $(N, m_0)$, before looking for times, which would transform the net into a bounded (and still live) Petri net, it makes sense first to check necessary conditions. First one could compute $T$-invariants.

$$
K_{t_3 t_4} = \{(0, 1, 1), (x_0 + x_1 + x_2, x_3, x_4) | \begin{array}{l}
2 \leq x_0 \leq 3, \quad x_0 + x_1 \leq 5, \\
2 \leq x_1 \leq 4, \quad x_0 + x_1 + x_2 \leq 5, \\
0 \leq x_2 \leq 3
\end{array} \}
$$

$$
K_{t_3 t_4} = \{(1, 1, 0), (x_0 + x_1 + x_2 + x_3, x_4, x_5) | \begin{array}{l}
2 \leq x_0 \leq 3, \quad 2 \leq x_1 \leq 4, \quad 2 \leq x_2 \leq 3, \\
0 \leq x_3 \leq 4, \quad x_0 + x_1 \leq 5, \quad x_0 + x_1 + x_2 \leq 5, \\
x_0 + x_1 + x_2 + x_3 \leq 5
\end{array} \}.
$$

Obviously, some of the inequalities are redundant. For instance, the inequalities of the set $B_{t_3 t_4}$ can be reduced to the set

$$
\begin{cases}
2 \leq x_0 \leq 3, & 2 \leq x_1 \leq 4, & 2 \leq x_2 \leq 3, \\
0 \leq x_3 \leq 4, & x_0 + x_1 + x_2 + x_3 \leq 5
\end{cases}
$$

In general, the number of inequalities in $B_\sigma$ is at most $\min\{2 \cdot (n \cdot |T| + 1), (n + 1) \cdot (n \sigma + 2)\}$ (cf. [18]).

Liveness can be defined in several ways for Petri nets [15]. We will use the standard “L4-live” variant, which states that every transition in a PN is potentially enabled in any reachable marking. More exactly, a transition $t$ in a Petri net $(N, m_0)$ is called live in $(N, m_0)$ if for every reachable marking $m$ in $(N, m_0)$ there exists a sequence of firings starting from $m$, which includes $t$.

### 3 Time Constrains for Boundedness

Let $(N, m_0)$ be a live and unbounded Petri net. We would like to check, whether it is possible to make this net bounded, not losing its liveness, by transforming it into a Time Petri net (with the same skeleton), i.e. by adding intervals to its transitions. To solve this problem we will associate transition intervals (if possible), which would exclude runs leading to unboundedness.

We start by recalling some properties of live and bounded Petri nets considered in our article [12]. Then, instead to assign priorities to the transitions we add intervals. We will show that this time solution is more precise than the solution with priorities. In this connection “more precise” means that the set of reachable markings in the time-dependent net (sometimes properly) covers the set of reachable markings in the net with priorities.

It is clear that if a live PN is bounded then there exists a feasible cyclic run, including all transitions of the PN. Furthermore, a TPN is obtained from a PN by adding time interval to each transition. Then the reachability tree is a subgraph of the reachability tree of the PN. Hence, it is obvious that if for some interval function $I$ the TPN $Z = (N, m_0, I)$ is live and bounded, then there exists a feasible cyclic run in $(N, m_0)$ which includes all transitions of $N$.

**Proposition 1.** Let $(N, m_0)$ be a live and unbounded Petri net. If there exists an interval function $I$ such that the TPN $(N, I, m_0)$ is live and bounded, then there exists a $T$-invariant without zero components for $N$, i.e. all transitions in $N$ are covered by some $T$-invariant.

As we already set in [12], given a live and unbounded Petri net $(N, m_0)$, before looking for times, which would transform the net into a bounded (and still live) Petri net, it makes sense first to check necessary conditions. First one could compute $T$-invariants.
for the net $\mathcal{N}$. If there is no T-invariant, covering all transitions in $\mathcal{N}$, then the net cannot be recovered, i.e., the net cannot become live due to adding time, priorities etc.

If there is such a T-invariant, then a more strong necessary condition can be checked: whether there exists a feasible cyclic run in $(\mathcal{N}, m_0)$, which includes all transitions in $\mathcal{N}$, i.e. a cyclic run realizing one of T-invariants with non-zero components. To do this check the algorithm, proposed in [5] by J. Desel, can be used. This algorithm is based on constructing a coverability net — a special extension of a coverability graph, and can take an exponential time. However, if a net does not have too much concurrency and a small number of unbounded places, this method can be acceptable.

Now let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net, and let the above necessary conditions are satisfied. We would like to find time intervals for the transition that will make the net bounded, keeping its liveness. The procedure will be illustrated by the net $(\mathcal{N}^*, m_0^*)$ in Fig. 2.

![Fig. 2. An example of a live and unbounded marked Petri net $(\mathcal{N}^*, m_0^*)$.](image)

The following algorithm is a modification of the algorithm given in [12]. Here we use the stages 1,2 and 3 of the ’old’ algorithm and add new stages 4 and 5.

**Stage 1. Find all minimal feasible cycles, which include all transitions.** As already mentioned, this can be done by the technique described by J. Desel in [5]. Moreover, following this technique for each minimal feasible cyclic run $\sigma$ we can simultaneously find a finite initial run $\tau$, such that $\tau\sigma^*$ is an initial run in $(\mathcal{N}, m_0)$.

If $(\mathcal{N}, m_0)$ does not have such cycles, then the problem does not have a solution. So, let

$$\mathcal{C}(\mathcal{N}, m_0) := \{ \tau\sigma \mid \tau\sigma^* \text{ is an initial run in } (\mathcal{N}, m_0),$$

$$\tau \text{ does not include } \sigma \text{ and }$$

$$\sigma \text{ includes all transitions in } \mathcal{N} \}$$

be a set of all minimal feasible cyclic runs together with prefixes leading to the cycles.

Thus, for example, the net $(\mathcal{N}^*, m_0^*)$ in Fig. 2 has five minimal cyclic runs with all transitions. Three of them have empty prefixes, and two have prefixes $\tau_1 = b$ and $\tau_2 = ba$, respectively:
Stage 2. Construct a spine tree. A spine tree is a subgraph of a reachability tree, containing exactly all runs from $C(N, m_0)$.

![Spine Tree Diagram](image)

Fig. 3. The spine tree for the net $(N^*, m_0^*)$.

The spine tree for Petri net $(N, m_0)$ from our example is shown in Fig. 3. Note, that a spine tree contains the behavior that should be saved to keep a Petri net live.

Stage 3. Construct a spine-based coverability tree. A spine-based coverability tree is a special kind of a coverability tree, that includes a spine tree as a backbone. Leaves in a spine-based coverability tree will be additionally colored with green or red. This coloring will be used then for computing transition priorities.

The spine-based coverability tree for a Petri net $(N, m_0)$ is defined constructively by the following algorithm:

Step 1. Start with the spine tree for $(N, m_0)$. Color all leaves in the spine tree in green.
Step 2. Repeat until all nodes are colored:
For each uncolored node \( v \) labeled with a marking \( m \):
1. check whether there is a marking \( m' \), directly reachable from \( m \) and not included in the current tree. For each such marking \( m' \), where \( m \rightarrow m' \):
   (a) Add a node \( v' \) labeled with \( m' \) as well as the corresponding arc from \( v \) to \( v' \) labeled with \( t \).
   (b) If the marking \( m' \) strictly covers a marking in some node on the path from the root to \( v' \), then \( v' \) becomes a leaf and gets the red color.
   (c) Otherwise, if the marking \( m' \) coincides with a marking labeling some node on the path from the root to \( v' \), then \( v' \) becomes a leaf and gets the green color.
   (d) Otherwise, leave \( v' \) uncolored.
2. Color the node \( v \) in yellow.

The spine-based coverability tree for our example net \((N^*, m^*_0)\) is shown in Fig. 4. Here node colors are used to illustrate the tree construction. A leaf and some inner node have the same color, if they have the same markings, or the leaf marking strictly covers the marking of its ancestor. Strictly covering leaves are marked with the \( \omega \)-symbol; they are 'red' leaves. All other leaves are 'green' leaves.

Fig. 4. The spine-based coverability tree for \((N^*, m^*_0)\):

Left: A leaf and some inner node have the same color, if they have the same markings, or the leaf marking strictly covers the marking of its ancestor. Strictly covering leaves are marked with the \( \omega \)-symbol. Right: The spine-based coverability tree for \((N^*, m^*_0)\) after finishing stage 3.

Stage 4. Compute a parametric state space. Let \( T \) be a spine-based coverability tree. Consider the TPN \((N, m_0, I)\) with time interval \([a_t, b_t]\) for each transition \( t \in T\).
All $a_t, b_t$ are unknown and have to be calculated in stage 5. By the construction of $T$, all its leaves are colored either in green, or red. In this stage we construct an interval function $I : T \rightarrow \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$. For this we consider every path from the root to a green leaf as a parametric run. Additionally, we forbid a branching to a red leaf using strict inequality.

(1) Let $v_g$ be a green leaf and let $\sigma$ be the path from the root to this leaf. Consider the parametric run $(\sigma(x), B_\sigma)$.

(2) Let $v_r$ be a red leaf. Consider the path $\sigma$ from the root to this leaf. Let $v^*$ be the youngest ancestor of $v_r$ such that at least one run goes from $v^*$ to a green leaf $v_g$. The initial node $v_0$ is labeled with the marking $m_0$. Let the node $v^*$ be labeled with $m^*, v_r$ with $m_r$ and $v_g$ with $m_g$. Finally, let $\sigma^*$ be the path from the root to the node $v^*, \sigma_r$ the path from the node $v^*$ to $v_g$ and $\tau_r \sigma_r$ the path from the node $v^*$ to $v_r$. That means, we have the situation

$$m_0 \xrightarrow{\sigma^*} m^* \xrightarrow{\sigma_g} v_g, \quad m_0 \xrightarrow{\sigma^*} m^* \xrightarrow{\tau_r \sigma_r} v, \quad \sigma = \sigma^* \tau_r \sigma_r$$

and there is not a path from a node in $\sigma_r$ to a green one. Hence, using time, we will forbid the firing of the transition $\tau_r$ in $m^*$. For this reason we add to $B_{\sigma^*}$ the constrain (strong inequality) $h_{\sigma_r}(\tau_r) < a_{t_r}$.

Stage 5. Compute a parametric state space. Let

$$B := \bigcup \{ B_\sigma \mid \sigma \text{ is an initial run to a green node} \} \cup \{ 0 \leq a_t \leq b_t \mid t \in T \}.$$ 

$B$ is the set of all constrains which have to be fulfilled in order to keep the live behavior of the net and to forbid all transition sequences leading to unboundedness. Clearly, $B$ is a system of linear inequalities and it can be solved in $\mathbb{Q}_0^+$. Actually, we are interested in finding solutions for all $a_t$’s and $b_t$’s such that the resulting system of inequalities is solvable. Of course, when we can find rational values for solutions for all $a_t$’s and $b_t$’s then we can find also integer values for them.

At this point we would like to notice that instead of the full parametric space of the spine-based coverability tree we can use only a part of them, consisting of paths including together all transitions of the net. In this case it is possible that some markings which does not lead to unboundedness in the PN will be not reachable in the TPN.

Applying the procedure of the stages 4 and 5 to the spine-based coverability tree for our example net $(N^*, m_0)$ (cf. Fig. 4) we sequentially obtain the following parametric state space and eventually the following inequality system $B$:

$K_\alpha = \{ (1, 0, 0, 1, 0, \{ x, x_0, x_1, x_2 \}) \mid 0 \leq x_0 \leq b_0 \}$,

$K_\beta = \{ (0, 1, 1, 1, 0, \{ x_3, x_4, x_5 \}) \mid a_b \leq x_0 \leq b_0, 0 \leq x_1 \leq b_a \}$,

$K_{\alpha B} = \{ (1, 0, 1, 1, 0, \{ x_6, x_7, x_8 \}) \mid a_a \leq x_0 \leq b_c, a_c \leq x_1 \leq b_a \}$,

$K_{\alpha B C} = \{ (0, 1, 0, 1, 0, \{ x_9, x_{10}, x_{11}, x_{12} \}) \mid a_c \leq x_0 \leq b_b, a_b \leq x_1 \leq b_a \}$,

$K_{\alpha B C D} = \{ (1, 0, 0, 1, 0, \{ x_6, x_7, x_8 \}) \mid a_a \leq x_0 \leq b_b, a_b \leq x_1 \leq b_a \}$,

$K_{\alpha B C D A} = \{ (1, 0, 1, 1, 0, \{ x_6, x_7, x_8 \}) \mid a_a \leq x_0 \leq b_b, a_b \leq x_1 \leq b_a \}$.
Here we have add the strong inequality $x_T < a_b$ in order to forbid the firing of the transition $b$ in the marking $(1, 0, 0, 0, 1)$. 

$$K_{babac} = \left\{ (1, 0, 0, 0, 1), (\emptyset, x_T, \emptyset, x_T) \right\} a_b \leq x_T \leq b_b, a_a \leq x_1 \leq b_a, a_b \leq x_2 \leq b_b, a_c \leq x_3 + x_4 \leq b_c, 0 \leq x_4 \leq b_d \} .$$

Subsequently, with respect to the properties of a interval function (cf. [18]) it has to be true:

$$0 \leq a_t \leq b_t \quad \text{for all } t \in \{a, b, c, d\}, \quad a_b \geq 1, \quad b_d \geq 1.$$
the TPN. Actually, having all interval bounds we construct the reachability graph/tree successively and without calculating any inequality systems, cf. [18]. The reachability graph of the TPN has 11 states and is was calculated with INA or tina (cf. [20], [21]). An analyzing protocol is included as an appendix.

**Proposition 2.** Let \((N, m_0)\) be a live and unbounded Petri net, for which there exists a feasible cyclic run, which includes all transitions in \(N\). Let then \(I\) be an interval function calculated for \((N, m_0)\) according to the algorithm described above. Then the TPN \((N, I, m_0)\) is live and bounded.

**Proof.** (Idea) Note, that if a node in a spine-based coverability tree has a descendant leaf, colored in red, then the marking of this node can enable an unbounded run, leading to an \(\omega\)-marking. If it has a descendant leaf, colored in green, then the marking of this node can enable a cycle, which includes all transitions in \(N\).

At each step at the stage 4 the parametric run forces initial runs to the markings labeling green leaves in the spine-based coverability tree of the Petri net and forbid the branching of each initial run ending into a marking which labels read leaf.

The set \(B\) of inequalities defined in stage 5 obtains all constains successively calculated in stage 4 and the conditions for interval bounds.

The resulting Time Petri net \((N, I, m_0)\) is live, because it retains all initial cycles with all transitions.

The net \((N, I, m_0)\) is bounded, because all unbounded branches in \((N, m_0)\) are cut by time constrains.

The above algorithm allows to calculate an interval function needed for the net boundedness, only in the case, when such function exist.

In order to calculate minimal values for the interval bounds \(a_t\) and \(b_t\) we can consider the Linear Programming

\[
\min \{ \sum_{t \in T} (a_t + b_t) \mid B \}.
\]

Note also, that in the above algorithm we keep all feasible cycles, containing all transitions. The algorithm can be modified to keep at least one such cycle.

Let us compare both algorithms for controlling the behavior of a Petri net: By using priorities, cf. [12], and by using time constrains as represented here. Both algorithms can "repair" a live and unbounded Petri net into a live and bounded. But, the second algorithm works also in some cases when the first one does not work and it works always when the first one does it. The reason for that is the statical use of the priority relation. The time intervals introduce also a kind of priority relation between the transitions, but this priority is a dynamic one. For example, for the net \((N^*, m_0)\) it is not possible to find a priority relation such that the firing of the transition \(d\) is prioritized to transition \(b\) but once, at the marking \((1, 0, 0, 0, 1)\) — where both transitions \(b\) and \(d\) may fire — \(b\) should fire. This is possible assigning time intervals, as we have seen. Thus, when the spine-based coverability tree would consist only of the left main branch, cf. Fig. 4, then there would not be a priority solution but a solution with time constrains.
4 Conclusion

In this paper we have studied the possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding time constrains as time intervals associated to each transition. We have presented necessary conditions for existence of such priorities. These conditions are not sufficient, but help to exclude unsolvable cases. Furthermore, we have represented an algorithm for computing (minimal) time intervals for transforming a live and unbounded Petri net into live and bounded and live net. The resulting net is a Time Petri net with the prior skeleton (underlying timeless net).

This algorithm converts a live and unbounded Petri net into a live and bounded Time Petri net even in some cases, when our previous algorithm for finding a priority relation for retaining the liveness does not work. Thus, the algorithm represented here is more powerful than the priority algorithm.

References


Evaluation of Decision Table Decomposition Using Dynamic Programming Classifiers

Michal Mankowski¹, Tadeusz Luba², and Cezary Jankowski²

¹ Warsaw University of Technology, Institute of Radioelectronics, Warsaw, Poland
m.mankowski@stud.elka.pw.edu.pl

² Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland
luba@tele.elka.pw.edu.pl, c.jankowski@stud.elka.pw.edu.pl

Abstract. Decision table decomposition is a method that decomposes given decision table into an equivalent set of decision tables. Decomposition can enhance the quality of knowledge discovered from databases by simplifying the data mining task. The paper contains a description of decision table decomposition method and their evaluation for data classification. Additionally, a novel method of obtaining attributes sets for decomposition was introduced. Experimental results demonstrated that decomposition can reduce memory requirements preserving the accuracy of classification.

Key words: Data Mining, Decomposition, Classifications, Dynamic Programming

1 Introduction

Increasing amount of data requires to develop new data analysis methods. A common approach in data mining is to make a prediction based on decision tables. Decomposition of a decision table to smaller subtables could be obtained by the divide and conquer strategy. This idea comes from logic synthesis and functional decomposition [1].

The fundamentals of decision system and logic synthesis are different, but there are many similarities between them. The decision system is usually described by a decision table, and combinatorial logic of a digital system by a truth table. Input variables of digital systems correspond to conditional attributes. Therefore, multiple terms from logic synthesis may be extended to data mining. Functional decomposition could be used to build a hierarchical decision system.

The functional decomposition was firstly used in logic synthesis of digital systems. In this situation, decomposition involves breaking a large logic functions, which are difficult to implement, into several smaller ones, which can be easy to implement. A similar problem in machine learning relies on disassembling the decision table to the subsystems in such a way that the original decision table can be recreated through a series of operations corresponding to the hierarchical decision making. But the most important is that we can induce noticeably simpler decision rules and trees for the resulting components to finally make the same decision as for the original decision table. [2–4]
For evaluation of decomposition, the decision trees and rules classifiers based on extensions of dynamic programming [5] were used. Decision tree and rules were sequentially optimized for different cost functions (for example, relative to number of misclassification and depth of decision trees). In the case of decision trees and rules this approach allowed to describe set of trees or rules by directed acyclic graph (DAG).

2 Basic Concepts

2.1 Preliminary Notions

Information system is a pair $\mathcal{A} = (U, A)$, where $U$ is a non-empty, finite set of objects called the universe, $A$ is a non-empty, finite set of attributes, i.e. each element $a \in A$ is a function from $U$ into $V_a$, where $V_a$ is the domain of $a$ called value set of $a$. Then, the function $\rho$ maps the product of $U$ and $A$ into the set of all values. The value of the attribute for a specific object is denoted by $\rho(u, a_i)$, with $u \in U$, $a_i \in A$

One or more distinguished attributes from set $A$ of information system may indicate a decision from rest of attributes. Such information system is called decision system. Formally, decision system is information system denoted by $\mathcal{A} = (U, A \cup D)$, where $A \cap D = \emptyset$. Attributes in set $A$ are referred to as conditional attributes while attributes in set $D$ are referred to as decision attributes. However, in the case when function $\rho$ maps $U \times (A \cup D)$ into the set of all attribute values such system is called decision table.

Let $\mathcal{A} = (U, A)$ be an information system. For each subset $B \subseteq A$ we define

$$IND_{\mathcal{A}}(B) = \{(u_p, u_q) \in U^2 : \forall a_i \in B, \rho(u_p, a_i) = \rho(u_q, a_i)\}$$

The attribute values $a_i$, i.e. $\rho_{pi} = \rho(u_p, a_i)$ and $\rho_{qi} = \rho(u_q, a_i)$ are compatible ($\rho_{pi} \sim \rho_{qi}$) if, and only if, $\rho_{pi} = \rho_{qi}$ or $\rho_{pi} = *$ or $\rho_{qi} = *$, where "*" represents attributes value "do not care". In the other case $\rho_{pi}$ and $\rho_{qi}$ are not compatible ($\rho_{pi} \not\sim \rho_{qi}$).

The consequence of this definition is compatibility relation $COM_{\mathcal{A}}(B)$ associated with every $B \subseteq A$:

$$COM_{\mathcal{A}}(B) = \{(u_p, u_q) \in U^2 : \forall a_i \in B, \rho(u_p, a_i) \sim \rho(u_q, a_i)\}$$

$COM_{\mathcal{A}}(B)$ classifies objects by grouping them into compatibility classes, i.e. $U/COM_{\mathcal{A}}(B)$, where $B \subseteq A$. Collection of subsets $U/COM_{\mathcal{A}}(B)$ is called $r$-partition on $U$ and denote as $\Pi_{\mathcal{A}}(B)$. R-partition on a set $U$ may be viewed as a collection of non-disjoint subsets of $U$, where the set union is equal $U$. All symbols and operations of partition algebra [6] are applicable to $r$-partitions. The $r$-partition generated by a set $B$ is the product of $r$-partitions generated by the attributes $a_i \in B$:

$$\Pi_{\mathcal{A}}(B) = \cap_{i} \Pi_{\mathcal{A}}(\{a_i\})$$

If $B = \{a_{i1}, ..., a_{ik}\}$, the product can be expressed as: $\Pi(B) = \Pi(a_{i1}) \cdot \ldots \cdot \Pi(a_{ik})$. We will write often $\cdot$ instead of $\cap$. 
2.2 Hierarchical Decomposition

To compress data and accelerate computations, hierarchical decomposition can be applied. The goal is to break down a decision table into two smaller subtables.

Let $\mathbb{F}$ be a functional dependency $D = \mathbb{F}(A)$ for a consistent decision system $\mathbb{A} = (U, A \cup D)$, where $A$ is a set of conditional attributes and $D$ is a set of decision attributes. Let $B_1, B_2$ be subsets of $A$ such that $A = B_1 \cup B_2$ and $B_1 \cap B_2 = \emptyset$. A simple hierarchical decomposition relative to $B_1, B_2$ exists for $\mathbb{F}(A)$ if and only if:

$$\mathbb{F}(A) = H(B_1, G(B_2)) = H(B_1, \delta)$$  \hspace{1cm} (4)

where $G$ and $H$ represent the following functional dependencies: $G(B_2) = \delta$ and $H(B_1, \delta) = D$, where $\delta$ is an intermediate attribute. The output of functions $\mathbb{F}(A)$ and $H$ are exactly the same. In other words we try to find a function $H$ depending on the variables of the set $B_1$ as well on the output $\delta$ of a function $G$ depending on the set $B_2$.

**Table 1.** An example decision table $\mathbb{A} = (U, A \cup D)$

<table>
<thead>
<tr>
<th></th>
<th>a_0</th>
<th>a_1</th>
<th>a_2</th>
<th>a_3</th>
<th>a_4</th>
<th>a_5</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Partition-based representation of data tables can be used to describe decomposition algorithms [6, 3]

**Theorem 1.** (\cite{3}) Functional dependencies $G$ and $H$ represent hierarchical decomposition of function $\mathbb{F}(A) = H(B_1, G(B_2))$, if there exists a r-partition $\Pi_G \geq \Pi(B_2)$ such that:

$$\Pi(B_1) \cdot \Pi_G \leq \Pi(D)$$  \hspace{1cm} (5)

where all the r-partitions are over the set of objects, and the number of values of component $G$ is equal to $L(\Pi_G)$ where $L(\Pi)$ denotes the number of blocks of r-partition $\Pi$.

In Theorem 1., r-partition $\Pi_G$ represents component $G$, and the product of r-partitions $\Pi(B_1)$ and $\Pi_G$ corresponds to $H$. The decision tables of the resulting components can be easily obtained from these r-partitions.
According to Theorem 1 the main problem is to find a partition $\Pi_G$. To solve that problem, a subset of original attributes $B_2$ and an $m$-block partition $\Pi(B_2) = \{K_1, K_2, \ldots, K_m\}$ generated by that subset is appropriate to consider. Two blocks $K_i, K_j$ of partition $\Pi(B_2)$ are compatible if and only if the partition $\Pi'$ obtained from $\Pi(B_2)$ by joining the blocks $K_i$ and $K_j$ into a single block $K_{ij}$ (without changing the other ones) satisfies the equation (5), i.e. iff $\Pi' \cdot \Pi_G \leq \Pi(D)$. Otherwise, $K_i, K_j$ are incompatible.

For decision table from Table 1 and sets of attributes $B_1 = \{a_0, a_5\}$, $B_2 = \{a_1, a_2, a_3, a_4\}$ the following set of incompatible pairs could be found: $E = \{(K_1, K_8), (K_2, K_4), (K_2, K_8), (K_3, K_7)(K_4, K_5)\}$. The subset of $n$ partition blocks, $\Pi(B_2) = \{K_{i_1}, K_{i_2}, \ldots, K_{i_n}\}$ where $K_{i_j} \in \Pi(B_2)$ is the compatible class of $\Pi(B_2)$ partition blocks iff all blocks of that subset are pairwise compatible. The compatibility class is referred to as maximal compatibility class (MCC) iff it does not belong to any other compatibility class of the partition concerned.

![Incompatibility graph and its coloring](image)

The decomposition process may be interpreted in terms of an incompatibility graph (Fig. 1). The edges represent the incompatible pairs of partition $\Pi(B_2) : (K_1, K_8), (K_2, K_4), (K_3, K_7)(K_4, K_5)$. It is clearly visible that the proper coloring of the graph specifies the compatible classes: $\{K_1, K_2, K_3, K_5\}$, $\{K_4, K_6, K_7, K_8\}$ and, as a consequence, the partition $\Pi_G = \{0, 1, 2, 4, 5, 7, 3, 6, 8\}$.

Another approach to building an incompatibility graph is to create a labeled partition matrix [7, 8] (Table 2). It should be noted that the columns represent all possible combinations of the attributes values in $B_2$. Each column thus denotes the behavior of decision table when the attributes in $B_1$ set are constant. Therefore each column can be treated as object from decision table. To build incompatibility graph it is necessary to apply the equation (2) to each pair of columns. When the compatibility relation is met then pair is compatible, otherwise it is incompatible.
Table 2. Partition matrix for the example set from Table 1 and the sets of attributes \( B_1 = \{a_0, a_5\} \) and \( B_2 = \{a_1, a_2, a_3, a_4\} \)

<table>
<thead>
<tr>
<th></th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( a_3 )</th>
<th>( a_4 )</th>
<th>( a_6 )</th>
<th>( a_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>2</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>*</td>
<td>3</td>
<td>*</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

\[ K_1 \quad K_1 \quad K_3 \quad K_4 \quad K_5 \quad K_6 \quad K_7 \quad K_8 \]

2.3 Attributes Selection Criteria for Further Decomposition

Simple hierarchical decomposition requires to divide a set of conditional attributes \( A \) to two disjoint subsets \( B_1 \) and \( B_2 \). Proposed idea of obtaining sets is based on the attributes relationship called attributes dependency from Rough Set theory [9].

Let \( C \) and \( B \) be sets of attributes, then \( B \) depends entirely on a set of attributes \( C \), denoted \( C \Rightarrow B \), if all values of attributes from \( B \) are uniquely determined by the values of attributes from \( C \). If \( B \) depends in degree \( k \), \( 0 \leq k \leq 1 \), on \( C \), then:

\[
k = \gamma(C, B) = \frac{|POS_C(B)|}{|U|}
\]

where

\[
POS_C(D) = \bigcup_{X \in U/B} C_*(X)
\]

called a positive region of the partition \( U/B \) with respect to \( C \), is the set of all elements of \( U \) that can be uniquely classified to blocks of the partition \( U/B \), by means of \( C \).

Proposed method allows us to measure dependency between all possible pairs of conditional attributes and decision attribute. Related dependency of one conditional attribute can be generated from a given information system: \( \mathcal{A} = (U, A \cup \{d\}) \), where \( A = \{a_0, ..., a_k\} \) is a set of conditional attributes and \( d \) is a decision attribute:

\[
r(x) = \frac{\sum_{i=0}^{k} \gamma(\{x, a_i\}, \{d\})}{|A|}, x \in A
\]

The above function of related dependency is used for comparison of attributes. This function is being calculated for each attribute, then the results are being sorted by the value of function \( r \). The most dependent attributes are put in set \( B_1 \), which corresponds to the final decision table \( H \).
Example 1. For Table 1 first step of the algorithm is to build a matrix of attribute dependency between each pair of condition attributes and decision attribute. Then the mean of partial results is calculated, which is represented by related dependency $r(x)$ in Table 3. These results can be sorted by value and divided into two equinumerous sets. If the number of attributes is odd, then set $|B_1| = |B_2| + 1$. An example of sorting and assignment attributes is presented in Table 4. The calculation of related dependency $r(x)$ allows formulating an accurate method for assessing sets $B_1$ and $B_2$, i.e. $B_1 = \{a_1, a_3, a_5\}$ and $B_2 = \{a_0, a_2, a_4\}$. Therefore the decomposition is as follows: $F(A) = H(\{a_1, a_3, a_5\}, G(\{a_0, a_2, a_4\}))$.

<table>
<thead>
<tr>
<th></th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.6</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.1</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
<td>0.6</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
</tr>
</tbody>
</table>

$r(x)$ 0.250 0.267 0.233 0.400 0.250 0.267

<table>
<thead>
<tr>
<th></th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.6</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.1</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
<td>0.6</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
</tr>
</tbody>
</table>

$r(x)$ 0.250 0.267 0.233 0.400 0.250 0.267

3 Classification Schema

3.1 Hierarchical Decision Making

Due to the decomposition of decision tables, there is a need for hierarchical decision system to evaluate this method for the purpose of classification [6]. This method is based on disassembling the decision table into the subtables. The most important advantage is the possibility to induce a simpler classification model, for example shorter decision rules or smaller decision tree for the resulting components to finally make the same decision as for the original decision table. Following the process of decomposition, we propose to take decisions hierarchically. For the part of the attributes $B_2$ a prediction model to calculate intermediate decision was built. Then this intermediate decision was used simultaneously with the attributes from $B_1$ to build final classification model. Then, on the basis of both, i.e., these attributes and the intermediate decision $\delta$, the final decision was taken (Fig. 2).

3.2 Dynamic Programming Classifiers

For decision prediction, the approach based on an extension of dynamic programming was used. These methods were developed in [5]. They allow sequential optimization of decision trees and rules relative to different cost functions, in particular between the
number of misclassifications and the depth of decision trees or the length of decision rules. Proposed algorithm constructs a directed acyclic graph (DAG), which represents structure of subtables of initial tables. For decision table \( A \) separable subtables of \( A \) described by systems of equalities of the kind \( a_i = b \) are considered as subproblems, where \( a_i \) is an attribute and \( b \) is its value. Classification and optimization of decision trees and rules are discussed in details in [5, 10].

In the applied approach to optimization of decision trees directed acyclic graph (DAG) represents a set of CART-like decision trees [11]. Set of Pareto optimal points for bi-criteria optimization problem is constructed. Two types of decision tree pruning have been compared. First is the multi-pruning for which, using validation subtable (part of training subtable) for each Pareto optimal point, a decision tree with minimum number of misclassification is found. Second, as an improvement of multi-pruning, is to use only the best split for a small number of attributes in each nodes of DAG graph, instead of using the best split for all attributes. This pruning is called restricted multi-pruning.

The system of decision rules as a prediction model was also considered. As in case of decision trees we used dynamic programming algorithm to create and optimize decision rules [10].

4 Experiments

To evaluate the proposed decomposition algorithm and hierarchical decision making idea the Dogger software system created in King Abdullah University of Science and Technology was used. Proposed algorithm has been tested on categorical datasets from UCI ML Repository [12]. A data sets were preprocessed. Duplicate rows were removed. There were some inconsistencies, i.e., there are instances with the same values of conditional attributes, but their decisions are different. The solution was to replace such set with a single row with most common decision. Results were obtained using the two-fold Cross-Validation evaluation repeated 100 times, each time using a different random selected testing subset. From training part, 70% of rows was used to generate decision trees and remaining part is preserved for validation.
Table 5. Compression results, where $S$ is size of original decision table and $S_D$ is a sum of tables after decomposition ($H$ and $G$).

<table>
<thead>
<tr>
<th>data set</th>
<th>rows</th>
<th>attributes</th>
<th>compression ($S_D/S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>flags</td>
<td>196</td>
<td>27</td>
<td>0.801</td>
</tr>
<tr>
<td>house</td>
<td>281</td>
<td>17</td>
<td>0.395</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>3198</td>
<td>37</td>
<td>0.209</td>
</tr>
<tr>
<td>breast cancer</td>
<td>268</td>
<td>10</td>
<td>0.754</td>
</tr>
<tr>
<td>cars</td>
<td>1730</td>
<td>7</td>
<td>0.261</td>
</tr>
<tr>
<td>spect-test</td>
<td>169</td>
<td>23</td>
<td>0.751</td>
</tr>
<tr>
<td>dermatology</td>
<td>366</td>
<td>35</td>
<td>0.352</td>
</tr>
</tbody>
</table>

The advantage of decomposition is due to the fact that two components (i.e. tables $G$ and $H$) require less memory than the original decision table. Let us express the size of the table as $S = n \sum b_i$, where $n$ is the number of objects, and $b_i = \lceil \log_2 |V_{a_i}| \rceil$ is the number of bits required to represent attribute $a_i$. Then, after the decomposition, we may compare the size of specific components with that of the original table (prior to decomposition). Results of compression are presented in Table 5.

Table 6. Comparison of accuracy error with and without decomposition.

<table>
<thead>
<tr>
<th>data set</th>
<th>rows</th>
<th>attributes</th>
<th>decomp.</th>
<th>dp rules</th>
<th>dp tree MP</th>
<th>dp tree RMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>flags</td>
<td>196</td>
<td>27</td>
<td>no</td>
<td>0.402 ± 0.041</td>
<td>0.409 ± 0.047</td>
<td>0.399 ± 0.041</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.402 ± 0.041</td>
<td>0.404 ± 0.038</td>
<td>0.417 ± 0.045</td>
</tr>
<tr>
<td>house</td>
<td>281</td>
<td>17</td>
<td>no</td>
<td>0.092 ± 0.013</td>
<td>0.064 ± 0.008</td>
<td>0.067 ± 0.009</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.076 ± 0.045</td>
<td>0.080 ± 0.047</td>
<td>0.073 ± 0.031</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>3198</td>
<td>37</td>
<td>no</td>
<td>0.019 ± 0.003</td>
<td>0.015 ± 0.004</td>
<td>0.062 ± 0.003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.061 ± 0.015</td>
<td>0.064 ± 0.016</td>
<td>0.011 ± 0.015</td>
</tr>
<tr>
<td>breast cancer</td>
<td>268</td>
<td>10</td>
<td>no</td>
<td>0.287 ± 0.022</td>
<td>0.302 ± 0.023</td>
<td>0.304 ± 0.026</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.274 ± 0.024</td>
<td>0.295 ± 0.020</td>
<td>0.297 ± 0.029</td>
</tr>
<tr>
<td>cars</td>
<td>1730</td>
<td>7</td>
<td>no</td>
<td>0.070 ± 0.009</td>
<td>0.062 ± 0.010</td>
<td>0.068 ± 0.004</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.122 ± 0.013</td>
<td>0.159 ± 0.022</td>
<td>0.112 ± 0.018</td>
</tr>
<tr>
<td>spect-test</td>
<td>169</td>
<td>23</td>
<td>no</td>
<td>0.048 ± 0.025</td>
<td>0.048 ± 0.019</td>
<td>0.047 ± 0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.056 ± 0.022</td>
<td>0.051 ± 0.016</td>
<td>0.049 ± 0.005</td>
</tr>
<tr>
<td>dermatology</td>
<td>366</td>
<td>35</td>
<td>no</td>
<td>0.229 ± 0.028</td>
<td>0.219 ± 0.028</td>
<td>0.221 ± 0.022</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>0.212 ± 0.030</td>
<td>0.226 ± 0.030</td>
<td>0.228 ± 0.027</td>
</tr>
</tbody>
</table>

Table 6. represents experimental results of classification using different dynamic programming algorithms. For each data set and method, accuracy error and standard deviation were calculated. As we can see the decomposition influences on accuracy. In cases of two data sets, i.e., kr-vs-kp and cars, accuracy error incomparably increased after decomposition. Note, that the compression for those data sets was very efficient.
However, for most of measurements, accuracy keeps a significant level or is slightly better. The biggest improvement occurring when dynamic programing rules were used.

5 Conclusion

Effective data aggregation algorithms have been sought after for a long time due to the increasing complexity of databases used in practice. Recently, some suggestions were put forward that decomposition algorithms, previously used mainly in logic synthesis of digital systems, may be applied for that purpose [13]. This approach is indeed very relevant as decision systems and logic circuits are very similar. Bearing this in mind, this paper demonstrates that a typical algorithm for the decomposition of binary data tables (representing Boolean functions) may be applied to the decomposition of data represented by multi-valued attributes used in decision systems.

The paper indicates the advantages and possibilities of decomposition algorithms for the purpose of classification. Results of experiments performed by proposed decomposition algorithm and Dagger system has been presented. New attributes selection criteria describing partitions for decomposition has been introduced and used in the experiments. Proposed method is particularly efficient in data compression. It allows to build simple classification model and save memory, simultaneously keep the accuracy. To achieved better results in accuracy data set decomposition requires further research, particularity with attributes selection criteria. Also, there is a need to extend the decomposition to deal with continuous attributes and noise in data.

Acknowledgments. The authors would like to thank professor Mikhail Moshkov and his team for their support while writing this paper. This research has been supported by King Abdullah University of Science and Technology.

References

10. Amin, T., Chikalov, I., Moshkov, M., Zielosko, B.: Dynamic programming approach to optimization of approximate decision rules. Information Sciences 221(0) (2013) 403 – 418
ExpTime Tableaux with Global Caching for Graded Propositional Dynamic Logic

Linh Anh Nguyen
Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We present the first direct tableau decision procedure for graded PDL, which uses global caching and has ExpTime (optimal) complexity when numbers are encoded in unary. It shows how to combine integer linear feasibility checking with checking fulfillment of existential star modalities for tableaux with global caching.

1 Introduction

Propositional dynamic logic (PDL) is a well-known modal logic [5, 8]. Originally, it was developed as a logic for reasoning about programs. However, its extensions are also used for other purposes. For example, converse-PDL with regular inclusion axioms (CPDL$_{reg}$) can be used as a framework for multiagent logics [4]. As a variant of PDL, $ALC_{reg}$ is a description logic for representing and reasoning about terminological knowledge. Several extensions of $ALC_{reg}$ have been studied by the description logic community [6].

The satisfiability problem in PDL is EXPTIME-complete [5]. In [15], Pratt gave a tableau decision procedure with global caching for deciding PDL. In [14], Nguyen and Szalas reformulated that procedure and extended it for dealing with checking consistency of an ABox w.r.t. a TBox in PDL (where PDL is treated as a description logic). The work [1] by Abate et al. gives another tableau decision procedure with global caching for PDL, which propagates unfulfillment of existential star modalities on-the-fly. There are also tableau decision procedures with global caching or state global caching for CPDL (PDL with converse) [13, 7] and CPDL$_{reg}$ [4, 10].

Graded modal logics allow graded modalities for reasoning about the number of successor states with a certain property. They have attracted attention from many researchers. In description logic, the counterpart of graded modalities is qualified number restrictions. Some well-known EXPTIME description logics with qualified number restrictions are $SHIQ$ and $SHOQ$. The description logic corresponding to graded CPDL is $CITQ$ [3]. De Giacomo and Lenzerini [3] proved that the satisfiability problem in $CITQ$ is EXPTIME-complete when numbers are encoded in unary. Tobies [16] proved that the satisfiability problem in $SHIQ$ is EXPTIME-complete even when numbers are encoded in binary.

1 See http://www.cs.man.ac.uk/~ezolin/ml/ for a list of related publications.
In this paper, we present the first direct tableau decision procedure for GPDL (graded PDL). Our procedure uses global caching and has ExPTIME (optimal) complexity when numbers are encoded in unary. It shows how to combine integer linear feasibility checking [9, 12] with checking fulfillment of existential star modalities for tableaux with global caching.

As related work on automated reasoning in GPDL and its extensions, De Giacomo and Lenzerini gave methods for translating the satisfiability problem in CIQ into CIF (a variant of CPDL with functionalities) [3], and in CIF into CPDL [2]. This established the complexity ExPTIME-complete for CIQ (and GPDL) when numbers are encoded in unary. However, this indirect method cannot give an efficient decision procedure for GPDL because it is not scalable w.r.t. numbers in graded modalities (i.e., qualified number restrictions). In particular, the translation from CIQ to CIF [3] may result in a formula with a quadratic length, and similarly for the translation from CIF to CPDL [3].

The rest of this paper is structured as follows. In Section 2, we present the notation and semantics of GPDL and recall automaton-modal operators [8, 4], which are used for our tableaux. We omit the feature of “global assumptions” as they can be expressed in PDL (by “local assumptions”). In Section 3, we present a tableau calculus for GPDL, starting with the data structure, the tableau rules and ending with the corresponding tableau decision procedure and its properties. In Section 4, we give an example for illustrating our procedure. Concluding remarks are given in Section 5.

2 Preliminaries

2.1 Graded Propositional Dynamic Logic

We use $\Sigma$ to denote the set of atomic programs, and $PROP$ to denote the set of propositions (i.e., atomic formulas). We denote elements of $\Sigma$ by letters like $\sigma$ and $\varrho$, and elements of $PROP$ by letters like $p$ and $q$.

A Kripke model is a pair $M = (\Delta^M, \cdot^M)$, where $\Delta^M$ is a set of states and $\cdot^M$ is an interpretation function that maps each proposition $p \in PROP$ to a subset $p^M$ of $\Delta^M$ and each atomic program $\sigma \in \Sigma$ to a binary relation $\sigma^M$ on $\Delta^M$. Intuitively, $p^M$ is the set of states in which $p$ is true and $\sigma^M$ is the binary relation consisting of pairs (input state, output state) of the program $\sigma$.

Formulas and programs of the base language of GPDL are defined respectively by the following grammar, where $p \in PROP$, $\sigma \in \Sigma$ and $n$ is a natural number:

$$
\varphi ::= \top | \bot | p | \neg \varphi | \varphi \land \varphi | \varphi \lor \varphi | \varphi \rightarrow \varphi | (\alpha)\varphi | [\alpha]\varphi | \geq n \sigma.\varphi | \leq n \sigma.\varphi
$$

$$
\alpha ::= \sigma | \alpha;\alpha | \alpha \cup \alpha | \alpha^* | \varphi?
$$

Notice that we use the notation $\geq n \sigma.\varphi$ and $\leq n \sigma.\varphi$ as in description logic instead of $\langle \sigma \rangle_{\geq n} \varphi$ and $\langle \sigma \rangle_{\leq n} \varphi$ (as the latter do not look “dual” to each other).

We use letters like $\alpha, \beta$ to denote programs, and $\varphi, \psi, \xi$ to denote formulas.

The intended meaning of program operators is as follows:

- $\alpha;\beta$ stands for the sequential composition of $\alpha$ and $\beta$.
\[ (\alpha; \beta)^M = \alpha^M \circ \beta^M \quad (\alpha \cup \beta)^M = \alpha^M \cup \beta^M \]
\[ (\alpha^*)^M = (\alpha^M)^* \quad (\varphi?)^M = \{(x, x) \mid \varphi^M(x)\} \]
\[ \top^M = \Delta^M \quad \bot^M = \emptyset \]
\[ (\neg \varphi)^M = \Delta^M \setminus \varphi^M \quad (\varphi \rightarrow \psi)^M = (\neg \varphi \lor \psi)^M \]
\[ (\varphi \land \psi)^M = \varphi^M \land \psi^M \quad (\varphi \lor \psi)^M = \varphi^M \lor \psi^M \]
\[ (\langle \alpha \rangle \varphi)^M = \{x \in \Delta^M \mid \exists y (\alpha^M(x, y) \land \varphi^M(y))\} \]
\[ ([\alpha] \varphi)^M = \{x \in \Delta^M \mid \forall y (\alpha^M(x, y) \rightarrow \varphi^M(y))\} \]
\[ (\geq n \sigma. \varphi)^M = \{x \in \Delta^M \mid \#\{y \in \Delta^M \mid \sigma^M(x, y) \land \varphi^M(y)\} \geq n\} \]
\[ (\leq n \sigma. \varphi)^M = \{x \in \Delta^M \mid \#\{y \in \Delta^M \mid \sigma^M(x, y) \land \varphi^M(y)\} \leq n\} \]

**Fig. 1.** Interpretation of complex programs and complex formulas.

- \( \alpha \cup \beta \) stands for the set-theoretical union of \( \alpha \) and \( \beta \)
- \( \alpha^* \) stands for the reflexive and transitive closure of \( \alpha \)
- \( \varphi? \) stands for the test operator.

Informally, a formula \( \langle \alpha \rangle \varphi \) represents the set of states \( x \) such that the program \( \alpha \) has a transition from \( x \) to a state \( y \) satisfying \( \varphi \). Dually, a formula \( [\alpha] \varphi \) represents the set of states \( x \) from which every transition of \( \alpha \) leads to a state satisfying \( \varphi \). A formula \( \geq n \sigma. \varphi \) (resp. \( \leq n \sigma. \varphi \)) represents the set of states \( x \) such that the program \( \sigma \) has transitions from \( x \) to at least (resp. at most) \( n \) pairwise different states satisfying \( \varphi \).

Formally, the interpretation function of a Kripke model \( M \) is extended to interpret complex formulas and complex programs as shown in Figure 1.

We write \( M, w \models \varphi \) to denote \( w \in \varphi^M \). For a set \( \Gamma \) of formulas, we write \( M, w \models \Gamma \) to denote that \( M, w \models \varphi \) for all \( \varphi \in \Gamma \). If \( M, w \models \varphi \) (resp. \( M, w \models \Gamma \)), then we say that \( M \) satisfies \( \varphi \) (resp. \( \Gamma \)) at \( w \), and that \( \varphi \) (resp. \( \Gamma \)) is satisfied at \( w \) in \( M \).

A formula is in negation normal form (NNF) if it does not use \(-\) and it uses \( \neg \) only immediately before propositions, and furthermore, it does not contain subformulas of the form \( \geq 0 \sigma. \varphi \) or \( \leq 0 \sigma. \varphi \). Every formula can be transformed to an equivalent formula in NNF. By \( \overline{\varphi} \) we denote the NNF of \( \neg \varphi \).

### 2.2 Automaton-Modal Operators

The alphabet \( \Sigma(\alpha) \) of a program \( \alpha \) and the regular language \( \mathcal{L}(\alpha) \) generated by \( \alpha \) are specified as follows:

\(^2\) Note that \( \Sigma(\alpha) \) contains not only atomic programs but also expressions of the form \( (\varphi?) \), and a program \( \alpha \) is a regular expression over its alphabet \( \Sigma(\alpha) \).
where for sets of words $M$ and $N, M . N = \{ \alpha \beta \mid \alpha \in M, \beta \in N \}, M^0 = \{ \varepsilon \}$ (where \varepsilon denotes the empty word), $M^{n + 1} = M . M^n \text{ for } n \geq 0$, and $M^* = \bigcup_{n \geq 0} M^n$.

We will use letters like $\omega$ to denote either an atomic program from $\Sigma$ or a test (of the form $\varphi ?$). A word $\omega_1 \ldots \omega_k \in L(\alpha)$ can be treated as the program $(\omega_1; \ldots; \omega_k)$, especially when it is interpreted in a Kripke model.

Recall that a finite automaton $A$ over alphabet $\Sigma(\alpha)$ is a tuple $(\Sigma(\alpha), Q, I, \delta, F)$, where $Q$ is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma(\alpha) \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A run of $A$ on a word $\omega_1 \ldots \omega_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, \omega_i, q_i)$ holds for every $1 \leq i \leq k$. It is an accepting run if $q_k \in F$. We say that $A$ accepts a word $w$ if there exists an accepting run of $A$ on $w$. The set of words accepted by $A$ is denoted by $L(A)$.

We will use the following convention:

- given a finite automaton $A$, we always assume that $A = (\Sigma_A, Q_A, I_A, \delta_A, F_A)$
- for $q \in Q_A$, we define $\delta_A(q) = \{ (\omega, q') \mid (\omega, q', q') \in \delta_A \}$.

As a finite automaton $A$ over alphabet $\Sigma(\alpha)$ corresponds to a program (the regular expression recognizing the same language), it is interpreted in a Kripke model as follows:

$$A^M = \bigcup \{ \gamma^M \mid \gamma \in L(A) \}. \quad (1)$$

For each program $\alpha$, let $A_{\alpha}$ be a finite automaton recognizing the regular language $L(\alpha)$. The automaton $A_{\alpha}$ can be constructed from $\alpha$ in polynomial time. We extend the base language with the auxiliary modal operators $[A, q]$ and $\langle A, q \rangle$, where $A$ is $A_{\alpha}$ for some program $\alpha$ and $q$ is a state of $A$. Here, $[A, q]$ and $\langle A, q \rangle$ stand respectively for $[(A, q)]$ and $\langle (A, q) \rangle$, where $(A, q)$ is the automaton that differs from $A$ only in that $q$ is its only initial state. We call $[A, q]$ (resp. $\langle A, q \rangle$) a universal (resp. existential) automaton-modal operator.

In the extended language, if $\varphi$ is a formula, then $[A, q] \varphi$ and $\langle A, q \rangle \varphi$ are also formulas. The semantics of these formulas are defined as usual, treating $\langle A, q \rangle$ as a program with semantics specified by (1).

Given a Kripke model $M$ and a state $x \in \Delta^M$, we have that $x \in ([A, q] \varphi)^M$ (resp. $x \in \langle (A, q) \varphi \rangle^M$) iff

$$x_k \in \varphi^M \text{ for all (resp. some) } x_k \in \Delta^M \text{ such that there exist a word } \omega_1 \ldots \omega_k \text{ (with } k \geq 0 \text{) accepted by } (A, q) \text{ with } (x, x_k) \in (\omega_1; \ldots; \omega_k)^M.$$ 

The condition $(x, x_k) \in (\omega_1; \ldots; \omega_k)^M$ means there exist states $x_0 = x, x_1, \ldots, x_{k-1}$ of $M$ such that, for each $1 \leq i \leq k$, if $\omega_i \in \Sigma$ then $(x_{i-1}, x_i) \in \omega_i^M$, else $\omega_i = (\psi_i ?)$ for some $\psi_i$ and $x_{i-1} = x_i$ and $x_i \in \psi_i^M$. Clearly, $\langle A, q \rangle \varphi$ is dual to $[A, q] \neg \varphi$ for any formula $\varphi$. 

\[ \Sigma(\sigma) = \{ \sigma \} \]
\[ \Sigma(\varphi ?) = \{ \varphi ? \} \]
\[ \Sigma(\beta, \gamma) = \Sigma(\beta) \cup \Sigma(\gamma) \]
\[ \Sigma(\beta \cup \gamma) = \Sigma(\beta) \cup \Sigma(\gamma) \]
\[ \Sigma(\beta^*) = \Sigma(\beta) \]
\[ \Sigma(\beta^*) = (\Sigma(\beta))^* \]
3 A Tableau Calculus for GPDL

In this section we present a tableau calculus for checking whether a given finite set of formulas in NNF is satisfiable. We specify the data structure, the tableau rules, the corresponding tableau decision procedure and state its properties.

3.1 The Data Structure

Let $EdgeLabels = \{\text{testingUnsat}, \text{checkingFeasibility}\} \times \Sigma$. For $e \in EdgeLabels$, let $e = (\pi_T(e), \pi_S(e))$. Thus, $\pi_T(e)$ is called the type of the edge label $e$, and $\pi_S(e)$ is an atomic program.

A tableau is a rooted graph $G = (V, E, \nu)$, where $V$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, $\nu \in V$ is the root, each node $v \in V$ has a number of attributes, and each edge $(v, w)$ may be labeled by a set $ELabels(v, w) \subseteq EdgeLabels$. The attributes of a tableau node $v$ are:

- $Type(v) \in \{\text{state, non-state}\}$,
- $Status(v) \in \{\text{unexpanded, p-expanded, f-expanded, unsat}\}$,
- $Label(v)$, which is a finite set of formulas, called the label of $v$,
- $RFmls(v)$, which is a finite set of so called reduced formulas of $v$,
- $ILConstraints(v)$, which is a finite set of integer linear constraints.

We call $v$ a state if $Type(v) = \text{state}$, and a non-state otherwise. If $(v, w) \in E$ then we call $v$ a predecessor of $w$ and $w$ a successor of $v$. An edge outgoing from a node $v$ is labeled iff $Type(v) = \text{state}$. The statuses p-expanded, f-expanded and unsat mean “partially expanded”, “fully expanded”, and “unsatisifiable”, respectively. $Status(v)$ may be p-expanded only when $Type(v) = \text{state}$, and $RFmls(v) \neq \emptyset$ only when $Type(v) = \text{non-state}$.

$ILConstraints(v)$ is available only when $Type(v) = \text{state}$ and $Status(v) \notin \{\text{unexpanded, p-expanded}\}$.

The constraints use variables $x_{w,e}$ indexed by a pair $(w, e)$ such that $(v, w) \in E$, $e \in ELabels(v, w)$ and $\pi_T(e) = \text{checkingFeasibility}$. Such a variable specifies how many copies of the successor $w$ using the edge label $e$ will be created for $v$.

We apply global caching in the sense that if $v_1$ and $v_2$ are different nodes then either $Type(v_1) \neq Type(v_2)$ or $Label(v_1) \neq Label(v_2)$ or $RFmls(v_1) \neq RFmls(v_2)$.

By $FullLabel(v)$ we denote the set $Label(v) \cup RFmls(v)$.

3.2 Tableau Rules

Our tableau calculus $C_{GPDL}$ for the GPDL is specified by a number of static rules, the (forming-state) rule, two transitional rules and the (unsat) rule for updating unsatisfiability of nodes. The rules except (unsat) are used to expand nodes of tableaux. Static rules are written downwards, with a set of formulas above the line as the premise, which represents the label of the node to which the rule is applied, and a number of sets of formulas below the line as the (possible) conclusions, which represent the labels of the
Table 1. The static rules of C_{GPDL}

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\forall) \quad X, \varphi \land \psi \over X, \varphi, \psi$</td>
<td>$(\forall) \quad X, \varphi \lor \psi \over X, \varphi \mid X, \psi$</td>
</tr>
<tr>
<td>if $\alpha \notin \Sigma$, $\alpha$ is not a test, and $I_{\alpha \omega} = {q_1, \ldots, q_k}$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{aut}_1) \quad X, [\alpha] \varphi \over X, [\alpha, q_1] \varphi, \ldots, [\alpha, q_k] \varphi$</td>
<td>$(\text{aut}_1) \quad X, \langle \alpha \rangle \varphi \over X, \langle \alpha, q_1 \rangle \varphi \mid \ldots \mid X, \langle \alpha, q_k \rangle \varphi$</td>
</tr>
<tr>
<td>if $\delta_A(q) = { (\omega_1, q_1), \ldots, (\omega_k, q_k) }$ and $q \notin F_A$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{[A]}) \quad X, [A, q] \varphi \over X, [\omega_1][A, q_1] \varphi, \ldots, [\omega_k][A, q_k] \varphi$</td>
<td>$(\text{[A]}) \quad X, \langle A, q \rangle \varphi \over X, \langle \omega_1 \rangle \langle A, q_1 \rangle \varphi \mid \ldots \mid X, \langle \omega_k \rangle \langle A, q_k \rangle \varphi$</td>
</tr>
<tr>
<td>if $\delta_A(q) = { (\omega_1, q_1), \ldots, (\omega_k, q_k) }$ and $q \in F_A$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{[A]}_f) \quad X, [A, q] \varphi \over X, [\omega_1][A, q_1] \varphi, \ldots, [\omega_k][A, q_k] \varphi, \varphi$</td>
<td>$(\text{[A]}_f) \quad X, \langle A, q \rangle \varphi \over X, \langle \omega_1 \rangle \langle A, q_1 \rangle \varphi \mid \ldots \mid X, \langle \omega_k \rangle \langle A, q_k \rangle \varphi \mid X, \varphi$</td>
</tr>
<tr>
<td>$(\Box ?) \quad X, [\psi?] \varphi \over X, \psi \mid X, \varphi$</td>
<td>$(\Diamond ?) \quad X, \langle \psi? \rangle \varphi \over X, \psi, \varphi$</td>
</tr>
<tr>
<td>$(\geq, \sigma) \quad X, \geq n \sigma, \varphi \over X, \geq n \sigma, \varphi, (\sigma) \varphi$ (n ≥ 1)</td>
<td></td>
</tr>
<tr>
<td>$(\Diamond \geq) \quad X, \langle \sigma \rangle \varphi \over X, \langle \sigma \rangle \varphi, \geq 1 \sigma, \varphi$</td>
<td></td>
</tr>
</tbody>
</table>

successor nodes resulting from the application of the rule. Possible conclusions of a static rule are separated by $. If a rule is unary (i.e. with only one possible conclusion) then its only conclusion is “firm” and we ignore the word “possible”. The meaning of a static rule is that if the premise is satisfiable then some of the possible conclusions are also satisfiable.

We use $\Gamma, X, Y$ to denote sets of formulas and write $\Gamma, \varphi$ to denote $\Gamma \cup \{\varphi\}$. The static rules of $C_{GPDL}$ are specified in Table 1. For any among them, the distinguished
Function \(\text{NewSucc}(v, \text{type}, \text{label}, r\text{Fmls}, e\text{Label})\)

Global data: a rooted graph \((V, E, \nu)\).

Purpose: create a new successor for \(v\).

create a new node \(w\), \(V := V \cup \{w\}\), \(E := E \cup \{(v, w)\}\);
\(\text{Type}(w) := \text{type}\), \(\text{Status}(w) := \text{unexpanded}\);
\(\text{Label}(w) := \text{label}\), \(r\text{Fmls}(w) := r\text{Fmls}\);
if \(\text{Type}(v) = \text{state}\) then \(\text{ELabels}(v, w) := \{e\text{Label}\}\);

return \(w\);

Function \(\text{ConToSucc}(v, \text{type}, \text{label}, r\text{Fmls}, e\text{Label})\)

Global data: a rooted graph \((V, E, \nu)\).

Purpose: connect a node \(v\) to a successor, which is created if necessary.

if there exists a node \(w\) such that \(\text{Type}(w) = \text{type}\), \(\text{Label}(w) = \text{label}\) and \(r\text{Fmls}(w) = r\text{Fmls}\) then
\(E := E \cup \{(v, w)\}\);
if \(\text{Type}(v) = \text{state}\) then \(\text{ELabels}(v, w) := \text{ELabels}(v, w) \cup \{e\text{Label}\}\);
else
\(w := \text{NewSucc}(v, \text{type}, \text{label}, r\text{Fmls}, e\text{Label})\);

return \(w\);

The formula of the premise is called the principal formula of the rule. A static rule \(\rho\) is applicable to a node \(v\) if the following conditions hold:

- \(\text{Status}(v) = \text{unexpanded}\) and \(\text{Type}(v) = \text{non-state}\),
- the premise of the rule is equal to \(\text{Label}(v)\),
- the conditions accompanied with \(\rho\) are satisfied,
- the principal formula of \(\rho\) does not belong to \(r\text{Fmls}(v)\).

The last condition means that if the principal formula belongs to \(r\text{Fmls}(v)\) then \(\rho\) has been applied to an ancestor node of \(v\) that corresponds to the same state in the intended Kripke model as \(v\), and therefore should not be applied again.

If \(\rho\) is a static rule applicable to \(v\), then the application is as follows:

- Let \(\varphi\) be the principal formula and \(X_1, \ldots, X_k\) the possible conclusions of \(\rho\).
- For each \(1 \leq i \leq k\), do \(\text{ConToSucc}(v, \text{non-state}, X_i, r\text{Fmls}(v) \cup \{\varphi\}, \text{null})\), which is specified on page 50.
- \(\text{Status}(v) := f\text{-expanded}\).

The (forming-state) rule is applicable to a node \(v\) if \(\text{Type}(v) = \text{non-state}\) and no static rule is applicable to \(v\). Application of this rule to such a node \(v\) is done by calling \(\text{ConToSucc}(v, \text{state}, \text{Label}(v), \emptyset, \text{null})\).

The transitional partial-expansion rule (TP) is applicable to a node \(v\) if \(\text{Type}(v) = \text{state}\) and \(\text{Status}(v) = \text{unexpanded}\). Application of this rule to such a node \(v\) is done as follows:

1. for each \(\langle \sigma \rangle \varphi \in \text{Label}(v)\) (where \(\sigma \in \Sigma\)), do
we continue modifying $E_v$ such extended nodes. So, we first start with the set follows:

1. $\mathcal{E} := \emptyset, \mathcal{E}' := \emptyset$
2. for each $(\geq n \sigma, \varphi) \in Label(v)$ do
   $\mathcal{E} := \mathcal{E} \cup \{(\sigma, X)\}$, where $X = \{\varphi\} \cup \{\psi \mid [\sigma] \psi \in Label(v)\}$
3. for each $(\leq n \sigma, \varphi) \in Label(v)$ do
   (a) for each $(\sigma, X) \in E$ do
   i. if $\{\varphi, \top\} \cap X = \emptyset$ then $\mathcal{E}' := \mathcal{E}' \cup \{(\sigma, X \cup \{\varphi\}), (\sigma, X \cup \{\top\})\}$
   (i.e., $(\sigma, X)$ is replaced by $(\sigma, X \cup \{\varphi\})$ and $(\sigma, X \cup \{\top\})$)
   ii. else $\mathcal{E}' := \mathcal{E}' \cup \{(\sigma, X)\}$
   (b) $\mathcal{E} := \mathcal{E}', \mathcal{E}' := \emptyset$
4. repeat
   for each $(\leq n \sigma, \varphi) \in Label(v), (\sigma, X) \in E$ and $(\sigma, X') \in \mathcal{E}$ such that $\varphi \in X \cap X'$, $(\sigma, X \cup X') \notin \mathcal{E}$ and $X \cup X'$ does not contain any pair of the form $\psi, \overline{\psi}$, add $(\sigma, X \cup X')$ to $\mathcal{E}$ (i.e., the merger of $(\sigma, X)$ and $(\sigma, X')$ is added to $\mathcal{E}$)
   until no tuples were added to $\mathcal{E}$ during the last iteration
5. for each $(\sigma, X) \in \mathcal{E}$ do
   $\text{ConToSucc}(v, \text{non-state}, X, \emptyset, (\text{checkingFeasibility}, \sigma))$
6. $\text{ILConstraints}(v) := \{x_{w,e} \geq 0 \mid (v, w) \in E, e \in ELabels(v, w) \text{ and } \pi_T(e) = \text{checkingFeasibility}\}$
7. for each $\varphi \in Label(v)$ do
   (a) if $\varphi$ is of the form $\geq n \sigma, \psi$ then add to $\text{ILConstraints}(v)$ the constraint
   $\sum\{x_{w,e} \mid (v, w) \in E, e \in ELabels(v, w), e = (\text{checkingFeasibility}, \sigma), \psi \in Label(w)\} \geq n$
   (b) if $\varphi$ is of the form $\leq n \sigma, \psi$ then add to $\text{ILConstraints}(v)$ the constraint
   $\sum\{x_{w,e} \mid (v, w) \in E, e \in ELabels(v, w), e = (\text{checkingFeasibility}, \sigma), \psi \in Label(w)\} \leq n$
8. $\text{Status}(v) := \text{f-expanded}$.

We give here an explanation for the rule (TF). To satisfy a requirement $(\geq n \sigma, \varphi) \in Label(v)$, one can first create a successor $w$ of $v$ specified by the pair $(\sigma, X)$ computed at the step 2, where $X$ presents $Label(w)$, and then clone $v$ to create $n$ successors for $v$ (or only record the intention somehow). The label of $w$ contains only formulas necessary for realizing the requirement $(\geq n \sigma, \varphi)$ and the related ones of the form $[\sigma] \psi$ at $v$. To satisfy requirements of the form $\leq n' \sigma, \varphi'$ at $v$, we tend to use only copies of such nodes like $w$ extended with either $\varphi'$ or $\overline{\varphi'}$ (for easy counting) as well as the mergers of such extended nodes. So, we first start with the set $\mathcal{E}$ constructed at the step 2, which consists of pairs with information about successors to be created for $v$. We then modify $\mathcal{E}$ by taking into account necessary extensions for the nodes (see the step 3). After that we continue modifying $\mathcal{E}$ by taking into account also appropriate mergers of nodes (see
Definition 1. Suppose \( \text{Status}(v) \neq \text{unsat} \) and \( \langle A, q \rangle \varphi \in \text{Label}(v) \). A trace of \( \langle A, q \rangle \varphi \) starting from \( v \) is a sequence \( (v_0, \varphi_0), \ldots, (v_k, \varphi_k) \) such that:

- \( v_0 = v \) and \( \varphi_0 = \langle A, q \rangle \varphi \),
- for every \( 1 \leq i \leq k \), \( v_i \) is a successor of \( v_{i-1} \), \( \text{Status}(v_i) \neq \text{unsat} \), and \( \varphi_i \) is a formula of \( \text{FullLabel}(v_i) \) such that
  - if the tableau rule expanding \( v_{i-1} \) is a static rule and \( \varphi_{i-1} \) is not its principal formula then \( \varphi_i = \varphi_{i-1} \),
  - else if the rule is \( \langle A \rangle \) or \( \langle A \rangle_f \) then \( \varphi_{i-1} \) is the principal formula of the form \( \langle A, q' \rangle \varphi \) and \( \varphi_i \) is the formula obtained from \( \varphi_{i-1} \),
  - else if the rule is \( \langle ? \rangle \) then \( \varphi_{i-1} \) is the principal formula of the form \( \langle ? \rangle \langle A, q' \rangle \varphi \) and \( \varphi_i = \langle A, q' \rangle \varphi \),
  - else \( \text{Type}(v_{i-1}) = \text{state} \), \( \varphi_{i-1} \) is of the form \( \langle \sigma \rangle \langle A, q' \rangle \varphi \), \( v_i \) is a successor of \( v_{i-1} \) resulting from the application of the tableau rule \( (\text{TP}) \) to \( v_{i-1} \), (testingUnsat, \( \sigma \)) \in \text{ELLabels}(v_{i-1}, v_i) \), and \( \varphi_i = \langle A, q' \rangle \varphi \).

The trace is called a \( \Diamond \)-realization for \( \langle A, q \rangle \varphi \) at \( v_0 \) if \( \varphi_k = \varphi \).

The \( \text{unsat} \) rule is specified as follows: set \( \text{Status}(v) := \text{unsat} \) if \( \text{Status}(v) \neq \text{unsat} \) and one of the following holds:

1. \( \perp \in \text{Label}(v) \) or there exists \( \{ \varphi, \neg \varphi \} \subseteq \text{Label}(v) \);
2. \( \text{Type}(v) = \text{non-state} \) and, for every \( (v, w) \in E \), \( \text{Status}(w) = \text{unsat} \);
3. \( \text{Type}(v) = \text{state} \) and there exist \( (v, w) \in E \) and \( e \in \text{ELabels}(v, w) \) such that \( \pi_T(e) = \text{testingUnsat} \) and \( \text{Status}(w) = \text{unsat} \);
4. \( \text{Type}(v) = \text{state} \), \( \text{Status}(v) = \text{f-expanded} \) and \( \text{ILConstraints}(v) \cup \{ x_{w, e} = 0 \mid (v, w) \in E, e \in \text{ELabels}(v, w), \pi_T(e) = \text{checkingFeasibility} \) and \( \text{Status}(w) = \text{unsat} \) \) is infeasible;
5. there does not exist any \( \Diamond \)-realization for some \( \langle A, q \rangle \varphi \in \text{Label}(v) \) at \( v \) when all paths starting from \( v \) do not contain any node that can be modified by some tableau rule.

3.3 Checking Unsatisfiability

Let \( \Gamma \) be a finite set of formulas in NNF. A \( \mathcal{C}_{\text{GPDL}} \)-tableau for \( \Gamma \) is a tableau \( G = (V, E, \nu) \) constructed as follows. At the beginning, \( V = \{ \nu \} \), \( E = \emptyset \) and the attributes of the root \( \nu \) are specified as follows: \( \text{Type}(\nu) = \text{non-state} \), \( \text{Status}(\nu) = \text{unexpanded} \), \( \text{Label}(\nu) = \Gamma \) and \( \text{RFmls}(\nu) = \emptyset \). Then, while \( \text{Status}(\nu) \neq \text{unsat} \) and there is a tableau rule applicable to some node \( v \), apply that rule to \( v \).\(^3\) Observe that the set of all formulas that may appear in the labels of the nodes of \( G \) is finite. Due to global caching, \( G \) is finite and can be effectively constructed.

\(^3\) As an optimization, it makes sense to expand \( v \) only when there may exist a path from the root to \( v \) that does not contain any node with status unsat.
Theorem 1. Let $\Gamma$ be a finite set of formulas in NNF and $G = (V,E,\nu)$ an arbitrary $C_{GPDL}$-tableau for $\Gamma$. Then, $\Gamma$ is unsatisfiable iff $\text{Status}(\nu) = \text{unsat}$.

To check satisfiability of a finite set $\Gamma$ of formulas in NNF, one can construct a $C_{GPDL}$-tableau $G = (V,E,\nu)$ for $\Gamma$ and return “no” when $\text{Status}(\nu) = \text{unsat}$, or “yes” otherwise. We call this the $C_{GPDL}$-tableau decision procedure. Various optimization techniques [11] can be applied to this procedure.

Corollary 1. The $C_{GPDL}$-tableau decision procedure has EXPTIME complexity when numbers are encoded in unary.

Theorem 1 and Corollary 1 can be proved in a similar way as done for the tableau decision procedures for CPDL$_{reg}$ [10], SHIQ [9] and SHOQ [12]. Proofs for them will be provided later for the full version of the current paper.

4 An Illustrative Example

Consider

$\Gamma = \{\langle \sigma^* \rangle p, \neg p, [\sigma; \sigma^*] \neg p, [\sigma] (\neg p \lor \neg q), \geq 1000 \sigma.q, \leq 1000 \sigma.(p \lor q)\}$

and let

$A_1 = A_{\sigma^*} = (\{\sigma\}, \{0\}, \{0\}, \{0,\sigma,0\}, \{0\})$

$A_2 = A_{\sigma^*;\sigma^*} = (\{\sigma\}, \{0,1,2\}, \{0\}, \{(0,\sigma,1),(1,\sigma,2),(2,\sigma,2)\}, \{2\})$.

A $C_{GPDL}$-tableau $G = (V,E,\nu)$ for $\Gamma$ is constructed as follows:

- At the beginning, $G$ contains only the non-state $\nu$ with $\text{Label}(\nu) = \Gamma$.
- Applying $(\text{aut}_\ominus)$ to $\nu$, this node is connected to a new non-state $v_1$ with

\[
\text{Label}(v_1) = \Gamma - \{\langle \sigma^* \rangle p\} \cup \{\langle A_1,0 \rangle p\}.
\]

- Applying $(\text{aut}_\cap)$ to $v_1$, this node is connected to a new non-state $v_2$ with

\[
\text{Label}(v_2) = \text{Label}(v_1) - \{[\sigma; \sigma^*] \neg p\} \cup \{[A_2,0] \neg p\}.
\]

- Applying $(\text{[A]})$ to $v_2$, this node is connected to a new non-state $v_3$ with

\[
\text{Label}(v_3) = \text{Label}(v_2) - \{[A_2,0] \neg p\} \cup \{[\sigma][A_2,1] \neg p\}.
\]

- Applying $(\geq_0)$ to $v_3$, this node is connected to a new non-state $v_4$ with

\[
\text{Label}(v_4) = \text{Label}(v_3) \cup \{\sigma q\}.
\]

- Applying $(\text{[A]}_f)$ to $v_4$ using the principal formula $\langle A_1,0 \rangle p$, this node is connected to two new non-states $v_5$ and $v_6$ with

\[
\text{Label}(v_5) = \text{Label}(v_4) - \{\langle A_1,0 \rangle p\} \cup \{p\}
\]

\[
\text{Label}(v_6) = \text{Label}(v_4) - \{\langle A_1,0 \rangle p\} \cup \{\sigma \langle A_1,0 \rangle p\}.
\]
- Since \( \{ p, \neg p \} \subseteq \text{Label}(v_5) \), applying the (unsat) rule to \( v_5 \), this node gets status unsat.
- Applying \((\land, \geq)\) to \( v_6 \), this node is connected to a new non-state \( v_7 \) with

\[
\text{Label}(v_7) = \text{Label}(v_6) \cup \{ \geq 1 \sigma, \langle A_1, 0 \rangle p \} = \{ \langle \sigma \rangle \langle A_1, 0 \rangle p, \geq 1 \sigma, \langle A_1, 0 \rangle p, \neg p, [\sigma][A_2, 1] \neg p, [\sigma](\neg p \lor \neg q), \\
\geq 1000 \sigma, q, \langle \sigma \rangle q, \leq 1000 \sigma.(p \lor q) \}.
\]
- Applying the (forming-state) rule to \( v_7 \), this node is connected to a new state \( v_8 \) with \( \text{Label}(v_8) = \text{Label}(v_7) \).
- Applying (TP) to \( v_8 \), this state is connected to two new non-states \( v_9 \) and \( v_{10} \), with \( \text{ELabels}(v_8, v_9) = \text{ELabels}(v_8, v_{10}) = \{ \langle \text{testingUnsat}, \sigma \rangle \} \) and

\[
\text{Label}(v_9) = \{ \langle A_1, 0 \rangle p, [A_2, 1] \neg p, \neg p \lor \neg q \} \quad \text{Label}(v_{10}) = \{ q, [A_2, 1] \neg p, \neg p \lor \neg q \}.
\]
- Applying (TF) to \( v_8 \), this state is connected to additional new non-states \( v_{11} - v_{16} \), with \( \text{ELabels}(v_8, v_i) = \{ e \} \), where \( e = \langle \text{checkingFeasibility}, \sigma \rangle \) and \( 11 \leq i \leq 16 \), and

\[
\text{ILConstraints}(v_8) \text{ consists of } x_{v_i, e} \geq 0, \text{ for } 11 \leq i \leq 16, \text{ and the following:}
\]

\[
\begin{align*}
x_{v_{11}, e} + x_{v_{12}, e} + x_{v_{13}, e} + x_{v_{16}, e} & \geq 1 \\
x_{v_{13}, e} + x_{v_{14}, e} + x_{v_{15}, e} + x_{v_{16}, e} & \geq 1000 \\
x_{v_{11}, e} + x_{v_{13}, e} + x_{v_{15}, e} & \leq 1000.
\end{align*}
\]
- Consider the node \( v_{12} \). To shorten the presentation, we ignore details about expansions for \( v_{12} \) and its descendants. We have \( \{\langle A_1, 0 \rangle p, [A_2, 1] \neg p, \neg p \lor \neg q\} \subseteq \text{Label}(v_{12}) \). It can be seen that there will not be any \( \diamond \)-realization for \( \langle A_1, 0 \rangle p \) at \( v_{12} \) (there will be a cycle going through nodes with status different from unsat). As a consequence, \( \text{Status}(v_{12}) \) will be changed at some step to unsat by the (unsat) rule.
- Consider the node \( v_{15} \). We have \( \{\langle A_1, 0 \rangle p, [A_2, 1] \neg p, q, \neg p \lor \neg q\} \subseteq \text{Label}(v_{15}) \). Similarly as for \( v_{12} \), it can be seen that there will not be any \( \diamond \)-realization for \( \langle A_1, 0 \rangle p \) at \( v_{15} \). As a consequence, \( \text{Status}(v_{15}) \) will be changed at some step to unsat by the (unsat) rule.
- Observe that \( \{ q, \neg p \lor \neg q \} \subseteq \text{Label}(v_{14}) \subseteq \text{Label}(v_{16}) \). Clearly, \( \text{Status}(v_{14}) \) and \( \text{Status}(v_{16}) \) will be changed at some steps to unsat.
- Consider the moment when the statuses of the nodes \( v_{12}, v_{14}, v_{15} \) and \( v_{16} \) have been changed to unsat and consider the set that extends \( \text{ILConstraints}(v_8) \) with
\[ x_{v_i,e} = 0 \text{ for } i \in \{12, 14, 15, 16\}. \] This set is reduced to the following one w.r.t. feasibility:

\[
\begin{align*}
  x_{v_{11},e} &\geq 1 \\
  x_{v_{13},e} &\geq 1000 \\
  x_{v_{11},e} + x_{v_{13},e} &\leq 1000.
\end{align*}
\]

Clearly, it is infeasible. As a consequence, \( \text{Status}(v_8) \) is changed to unsat by the \((\text{unsat})\) rule. By applying this rule in the propagation manner, the statuses of the nodes \( v_7, v_6, v_4 - v_1, v \) are changed to unsat one after the other. According to Theorem 1, we claim that the set \( \Gamma \) is unsatisfiable.

5 Conclusions

We have given the first direct tableau decision procedure for GPDL, which has \( \text{Exp-TIME} \) (optimal) complexity when numbers are encoded in unary. It uses global caching and exploits our technique of integer linear feasibility checking [9].

We have implemented our procedure in the scope of the reasoner TGC2.\(^4\) This reasoner also allows converse modalities [13, 9] and ABoxes [14, 9]. As far as we know, it is the first reasoner that can decide GPDL.

Preliminary experiments with TGC2 showed that our method deals with number restrictions (graded modalities) much better than the well-known reasoners Racer, FaCT++, HermiT and Pellet for description logics.

Acknowledgments

This work was done in cooperation with the project 2011/02/A/HS1/00395, which is granted by the Polish National Science Centre (NCN).

References


Estimation and Feature Selection by Application of Knowledge Mined from Decision Rules Models

Wieslaw Paja¹ and Krzysztof Pancerz²,³

¹ Department of Computer Science,
Faculty of Mathematics and Natural Sciences,
University of Rzeszow, Prof. St. Pigiona Str. 1, 35-310 Rzeszow, Poland
wpaja@ur.edu.pl
² University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszow, Poland
³ University of Management and Administration
Akademicka Str. 4, 22-400 Zamosc, Poland
kpancerz@wszia.edu.pl

Abstract. Feature selection methods, as a preprocessing step to machine learning, are effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. However, the recent increase of dimensionality of data poses a severe challenge to many existing feature selection methods with respect to the efficiency and effectiveness. In this work, we introduce a novel concept, relevant feature selection based on information gathered from decision rule models. A new measure for a feature rank based on the feature frequency and rule quality is additionally defined. The efficiency and effectiveness of our method is demonstrated through exemplary use of five real-world datasets. Six different classification algorithms were used to measure the quality of learning models built on original features and on selected features.

Key words: Feature selection, feature ranking, decision rules, dimensionality reduction, relevance and irrelevance

1 Introduction

In the era of the acquisition of vast amounts of data, different domain information databases, efficient analysis and retrieval of regularity have become an extremely important task. The issue of classification and object recognition is applied in many fields of human activity. Data mining is fraught with many aspects which hinder it, like a very large number of observations, too many attributes, the insignificance of the part of variables for the classification process, mutual interdependence of conditional variables, the simultaneous presence of variables with different types, the presence of undefined values of variables, the presence of erroneous values of the variables, uneven distribution of categories for the target variable. Thus, the development of efficient methods for significant feature selection is valid.

Feature selection (FS) methods are frequently used as a preprocessing step to machine learning experiments. An FS method can be defined as a process of choosing a
subset of original features so that the feature space is optimally reduced according to a certain evaluation criterion. Feature selection has been a fruitful field of research and development since 1970’s and it has been proven to be effective in removing irrelevant features, increasing efficiency in learning tasks, improving learning performance like predictive accuracy, and enhancing comprehensibility of the learned results [1].

The feature selection methods are typically divided into three classes based on how they combine the selection algorithm and the model building: filter, wrapper and embedded FS methods. Filter methods select features with respect to the model. They are based only on general features like the correlation with the variable to be predicted. These methods select only the most interesting variables. Then, a selected subset will be a part of the classification model. Such methods are effective in computation time and robust to overfitting [2]. However, some redundant, but relevant features can remain unrecognized. In turn, wrapper methods evaluate subsets of features which allow to detect the possible interactions between variables [1,3]. However, the increase in overfitting risk, when the number of observations is insufficient, is possible. Additionally, the significant computation time, when the number of variables is large, highly increases. The third type, called embedded methods, is intended for reducing the classification of learning. Methods in this group try to combine the advantages of both methods mentioned previously. Thus, the learning algorithm takes advantage of its own variable selection algorithm. Therefore, it needs to know initially what a good selection is, which limits its exploitation [4].

Kohavi and John [1] observed that there are several definitions of relevance that may be contradictory and misleading. They proposed two degrees of relevance (strong and weak) that are required to encompass all notions usually associated with this term. In their approach the relevance is defined in the absolute terms, with the help of the ideal Bayes classifier. In this context, a feature \( X \) is strongly relevant when removal of \( X \) alone from the data always results in deterioration of the prediction accuracy of the ideal Bayes classifier. In turn, a feature \( X \) is weakly relevant if it is not strongly relevant and there exists a subset of features \( S \), such that the performance of the ideal Bayes classifier on \( S \) is worse than the performance on \( S \cup \{ X \} \). A feature is irrelevant if it is neither strongly nor weakly relevant.

Nilsson et al. [5] introduced the formal definition of two different feature selection problems:

1. **Minimal Optimal Feature Selection (MOFS)** consisting in identification of minimal set of features to obtain the optimal quality of classification.
2. **All Relevant Feature Selection (ARFS)**, where the problem is to find all the variables that may, under certain conditions, improve the classification.

There are two important differences between these problems. The first one is detection of attributes with low importance (ARFS) [6], which may be completely obscured by other, more important attributes, from the point of view of the classifier (MOFS). The second difference is to find the boundary between the variables poorly, but realistically related to the decision and those for which such a relation is created as a result of random fluctuations. The formal definition of the problem of all relevant feature selection (ARFS) as a distinct problem from the classical minimal optimal feature selection (MOFS), was proposed recently in 2007 [5].
In our research, we used the *contrast variable* concept to distinguish between relevant and irrelevant features [6]. It is a variable that does not carry information on the decision variable by design that is added to the system in order to discern relevant and irrelevant variables. Here, it is obtained from the real variables by random permutation of values between objects. The use of contrast variables was, for the first time, proposed by Stoppiglia et al. [7] and then by Tuv et al. [8].

2 Methods and Algorithms

During experiments the following general procedure was applied:

1. **Step 1. Selection of dataset and features for investigation.**
   (a) Application of a set of ranking measures to calculate importance for each feature:
      i. With set of contrast features.
      ii. Without contrast features.
   (b) Definition (selection) of the most important feature subset.

2. **Step 2. Application of different machine learning algorithms for classification of unseen objects using the 10-fold cross validation method:**
   (a) Using all original features.
   (b) Using only selected, important features.

3. **Step 3. Comparison of gathered results using different evaluation measures.**

In the first step, a dataset as well as a feature for investigation were defined. Then, different ranking measures were applied to estimate importance of each feature. In order to check specificity of the feature selection, the dataset was extended by adding contrast variables. It means that each original variable was duplicated and its values were randomly permuted between all objects. Hence, a set of non-informative by design shadow variables was added to original variables. The variables that were selected as important more significantly than random, were examined further, using different tests. To define the level of feature importance, six well-known ranking measures were applied: *Relieff, Information Gain, Gain Ratio, Gini Index, SVM weight*, and *RandomForest*. Additionally, our new measure, called *RQualityFS*, was introduced. It is based on the frequency of presence of different feature in a rule model generated from an original dataset and it also takes into consideration the quality of the rules in which this feature occurs. Rank quality of the \( i \)-th feature could be presented as follow:

\[
Q_{A_i} = \sum_{j=1}^{n} Q_{R_j} \{ A_i \} 
\]

where \( n \) is a number of rules inside the model, \( Q_{R_j} \) defines the classification quality of the rule \( R_j \) and \( \{ A_i \} \) describes the presence of the \( i \)-th attribute, usually it is equal to 1 (the feature occurred) or to 0 (the feature did not occur).

In turn, the quality of the rule is defined as follows:

\[
Q_{R_j} = \frac{E_{corr}}{E_{corr} + E_{incorr}}
\]
where \( E_{corr} \) depicts a number of correctly matched learning objects by the \( j \)-th rule and \( E_{incorr} \) depicts a number of incorrectly matched learning objects by this rule.

During the second step, a test probing the importance of variables was performed by analyzing the influence of variables used for model building on the prediction quality. Six different machine learning algorithms were applied to build different predictors for the original set of features and for selected features: Classification Tree (CT), Random Forest (RF), CN2 decision rules algorithm (CN2), Naïve Bayes (NB), k-Nearest Neighbors (kNN), and Support Vector Machine (SVM). During this step, a 10-fold cross validation paradigm was used. Ten known evaluation measures were utilized in each predictor: Classification Accuracy (CA), Sensitivity, Specificity, Area Under ROC curve (AUC), Information Score (IS), F1 score (F1), Precision, Brier measure, Matthew Coefficient Correlation (MCC) parameter, and finally Informadness (Inform.) ratio [9].

3 Investigated Datasets

Our initial investigations focus on applying the developed algorithm on several real-world datasets. Five datasets have been used during experiments. Four of them are gathered from the UCI ML repository, while the fifth set has been developed earlier by the authors [10]. A summary of datasets is presented in Table 1. These datasets have diverse numbers of objects, features and their types as well as classes.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># instances</th>
<th># features</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>286</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Heart disease</td>
<td>303</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>32</td>
<td>56</td>
<td>3</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>339</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>Skin cancer</td>
<td>548</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

4 Results and Conclusions

To illustrate the proposed methodology, only results for Breast cancer datasets will be presented in details. The first step of the experiment revealed six features, that were recommended as important by all or almost all ranking measures. In Table 2, we can observe that deg-malig, node-caps, irradiat, inv-nodes, breast, and menopause features create a stable and core set of features which have the highest values of seven measures of importance, particularly using RQualityFS measure, introduced in our investigation. In the same table, comparison with importance of contrast values (italic rows and contrast index) is also presented. The most important contrast feature is tumor-size (contrast) for which RQualityFS measure, defined earlier, is equal to 2.34. In this way, we
also treated a threshold that separates the core, relevant set of attributes from other less informative attributes. Most of the measures (except SVM weight) used in this approach show that the selected set of features has higher values of these parameters than the gathered threshold value (underlined values). These values are denoted in bold style in Table 2. Hereby, we can observe that different measures give different thresholds.

Table 2. Ranking of features using seven different measures

<table>
<thead>
<tr>
<th>Feature</th>
<th>ReliefF</th>
<th>Inf. Gain</th>
<th>Gini gain</th>
<th>SVM weight</th>
<th>RF</th>
<th>RQualityFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>deg-malig</td>
<td>0.08</td>
<td>0.08</td>
<td>0.05</td>
<td>0.02</td>
<td>0.07</td>
<td>2.01</td>
</tr>
<tr>
<td>node-caps</td>
<td>0.15</td>
<td>0.06</td>
<td>0.08</td>
<td>0.02</td>
<td>0.05</td>
<td>1.21</td>
</tr>
<tr>
<td>irradiat</td>
<td>0.13</td>
<td>0.03</td>
<td>0.03</td>
<td>0.01</td>
<td>0.00</td>
<td>0.88</td>
</tr>
<tr>
<td>inv-nodes</td>
<td>0.15</td>
<td>0.07</td>
<td>0.05</td>
<td>0.02</td>
<td>0.03</td>
<td>0.32</td>
</tr>
<tr>
<td>breast</td>
<td>-0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
<td>0.32</td>
</tr>
<tr>
<td>menopause</td>
<td>-0.03</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>tumor-size (contrast)</td>
<td>-0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.04</td>
<td>0.01</td>
</tr>
<tr>
<td>Age</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.04</td>
<td>-0.12</td>
</tr>
<tr>
<td>breast (contrast)</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
</tr>
<tr>
<td>tumor-size</td>
<td>0.07</td>
<td><strong>0.06</strong></td>
<td>0.02</td>
<td>0.01</td>
<td>0.10</td>
<td>0.04</td>
</tr>
<tr>
<td>age (contrast)</td>
<td>0.05</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>-0.06</td>
</tr>
<tr>
<td>deg-malig (contrast)</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td><strong>0.46</strong></td>
</tr>
<tr>
<td>menopause (contrast)</td>
<td><strong>0.09</strong></td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.06</td>
<td>0.02</td>
</tr>
<tr>
<td>irradiat (contrast)</td>
<td>-0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.16</td>
</tr>
<tr>
<td>breast-quad (contrast)</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.03</td>
<td>-0.05</td>
</tr>
<tr>
<td>inv-nodes (contrast)</td>
<td>-0.02</td>
<td><strong>0.02</strong></td>
<td><strong>0.02</strong></td>
<td><strong>0.01</strong></td>
<td><strong>0.14</strong></td>
<td>0.07</td>
</tr>
<tr>
<td>node-caps (contrast)</td>
<td>-0.04</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.02</td>
<td>-0.03</td>
</tr>
<tr>
<td>breast-quad (contrast)</td>
<td>-0.05</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td><strong>0.16</strong></td>
<td>0.13</td>
</tr>
</tbody>
</table>

The second step of the experiment was devoted to evaluation of prediction of the quality of utilized machine learning algorithms described in Section 2. During this step, six different algorithms were applied using the 10-fold cross validation method. The average results for the Breast cancer dataset are shown in Figure 1. This procedure was utilized for two specified sets:
1. The original set containing all descriptive features.
2. Only the set of selected features defined on the basis of its importance calculated in the first step.

Additionally, all average results for the Breast cancer dataset and for other investigated data are collected in Tables from 4 to 9. On the basis of these results, it could be stressed that the set of selected features, which contains only 6 from 9 attributes, has the similar (even better) prediction quality than it was observed with all 9 original attributes. For instance, the popular AUC and classification accuracy (CA) measures in data analysis increased (see Figure 1). Furthermore, all other measures also increased a little. With the exception of the Brier score, which decreased, but the lower the Brier score is for a set of predictions, the better the predictions are calibrated [11].

Fig. 1. Summary results and their standard deviations for Breast cancer dataset.

Similar results were obtained for other investigated datasets (see Figures from 2 to 5). Effectiveness of six learning algorithms used in this research can be observed on the basis of the results collected in Tables from 4 to 9. A summary of feature selection is presented in Table 3.

All numbers of features in the selected sets are significantly smaller than in the original one. The average is only about 37% of original features selected. Using these selected sets promising initial results of classification efficiency can be gained together with substantial reduction of the problem of dimensionality.

Acknowledgments. This work was supported by the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszow.

References

## Table 3. Summary of selected features in benchmark datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># original</th>
<th># selected</th>
<th>% of original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>9</td>
<td>6</td>
<td>67</td>
</tr>
<tr>
<td>Heart disease</td>
<td>13</td>
<td>5</td>
<td>38</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>56</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>17</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>Skin cancer</td>
<td>13</td>
<td>6</td>
<td>46</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>22</strong></td>
<td><strong>5</strong></td>
<td><strong>37</strong></td>
</tr>
</tbody>
</table>

## Table 4. Average results of Classification tree on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.71</td>
<td>0.58</td>
<td>0.63</td>
<td>0.10</td>
<td>0.58</td>
<td>0.63</td>
<td>0.44</td>
<td>0.21</td>
<td>0.17</td>
<td>0.16</td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.76</td>
<td>0.76</td>
<td>0.79</td>
<td>0.51</td>
<td>0.76</td>
<td>0.76</td>
<td>0.42</td>
<td>0.52</td>
<td>0.52</td>
<td>0.46</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.43</td>
<td>0.44</td>
<td>0.72</td>
<td>0.34</td>
<td>0.42</td>
<td>0.43</td>
<td>0.89</td>
<td>0.16</td>
<td>0.16</td>
<td>0.43</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.39</td>
<td>0.19</td>
<td>0.97</td>
<td>0.79</td>
<td>1.16</td>
<td>0.20</td>
<td>0.18</td>
<td>0.79</td>
<td>0.16</td>
<td>0.14</td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.79</td>
<td>0.78</td>
<td>0.92</td>
<td>1.30</td>
<td>0.78</td>
<td>0.78</td>
<td>0.34</td>
<td>0.70</td>
<td>0.70</td>
<td>0.64</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.62</strong></td>
<td><strong>0.55</strong></td>
<td><strong>0.79</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.55</strong></td>
<td><strong>0.56</strong></td>
<td><strong>0.58</strong></td>
<td><strong>0.35</strong></td>
<td><strong>0.34</strong></td>
<td><strong>0.37</strong></td>
</tr>
</tbody>
</table>

**Fig. 2.** Summary results and their standard deviations for Heart disease dataset.
Table 5. Average results of Random forest on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.75</td>
<td>0.59</td>
<td>0.59</td>
<td>0.70</td>
<td>0.08</td>
<td>0.58</td>
<td>0.79</td>
<td>0.37</td>
<td>0.32</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.61</td>
<td>0.61</td>
<td>0.71</td>
<td>0.10</td>
<td>0.61</td>
<td>0.75</td>
<td>0.36</td>
<td>0.33</td>
<td>0.21</td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.81</td>
<td>0.81</td>
<td>0.81</td>
<td>0.89</td>
<td>0.41</td>
<td>0.81</td>
<td>0.82</td>
<td>0.27</td>
<td>0.63</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.86</td>
<td>0.35</td>
<td>0.76</td>
<td>0.77</td>
<td>0.31</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.35</td>
<td>0.33</td>
<td>0.65</td>
<td>0.73</td>
<td>0.17</td>
<td>0.34</td>
<td>0.43</td>
<td>0.61</td>
<td>0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.53</td>
<td>0.75</td>
<td>0.68</td>
<td>0.30</td>
<td>0.50</td>
<td>0.49</td>
<td>0.59</td>
<td>0.26</td>
<td>0.28</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.45</td>
<td>0.21</td>
<td>0.97</td>
<td>0.87</td>
<td>1.03</td>
<td>0.33</td>
<td>0.37</td>
<td>0.71</td>
<td>0.35</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.11</td>
<td>0.96</td>
<td>0.82</td>
<td>0.68</td>
<td>0.37</td>
<td>0.31</td>
<td>0.80</td>
<td>0.25</td>
<td>0.07</td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.83</td>
<td>0.79</td>
<td>0.93</td>
<td>0.97</td>
<td>1.11</td>
<td>0.80</td>
<td>0.85</td>
<td>0.27</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.69</td>
<td>0.90</td>
<td>0.94</td>
<td>0.99</td>
<td>0.68</td>
<td>0.72</td>
<td>0.33</td>
<td>0.61</td>
<td>0.58</td>
</tr>
<tr>
<td>Average</td>
<td>0.64</td>
<td>0.54</td>
<td>0.79</td>
<td>0.83</td>
<td>0.56</td>
<td>0.57</td>
<td>0.65</td>
<td>0.44</td>
<td>0.41</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>0.62</td>
<td>0.54</td>
<td>0.79</td>
<td>0.80</td>
<td>0.48</td>
<td>0.58</td>
<td>0.61</td>
<td>0.48</td>
<td>0.40</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 6. Average results of CN2 rules on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.72</td>
<td>0.57</td>
<td>0.57</td>
<td>0.61</td>
<td>0.12</td>
<td>0.56</td>
<td>0.68</td>
<td>0.46</td>
<td>0.22</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.60</td>
<td>0.60</td>
<td>0.66</td>
<td>0.14</td>
<td>0.61</td>
<td>0.74</td>
<td>0.39</td>
<td>0.31</td>
<td>0.21</td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
<td>0.84</td>
<td>0.58</td>
<td>0.81</td>
<td>0.83</td>
<td>0.33</td>
<td>0.64</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>0.74</td>
<td>0.73</td>
<td>0.73</td>
<td>0.76</td>
<td>0.43</td>
<td>0.74</td>
<td>0.75</td>
<td>0.43</td>
<td>0.48</td>
<td>0.47</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.44</td>
<td>0.44</td>
<td>0.71</td>
<td>0.65</td>
<td>0.39</td>
<td>0.44</td>
<td>0.46</td>
<td>0.72</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>0.56</td>
<td>0.55</td>
<td>0.77</td>
<td>0.69</td>
<td>0.25</td>
<td>0.68</td>
<td>0.58</td>
<td>0.60</td>
<td>0.46</td>
<td>0.32</td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.45</td>
<td>0.21</td>
<td>0.97</td>
<td>0.87</td>
<td>1.03</td>
<td>0.33</td>
<td>0.37</td>
<td>0.71</td>
<td>0.35</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.11</td>
<td>0.96</td>
<td>0.82</td>
<td>0.68</td>
<td>0.37</td>
<td>0.31</td>
<td>0.80</td>
<td>0.25</td>
<td>0.07</td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.82</td>
<td>0.79</td>
<td>0.93</td>
<td>0.94</td>
<td>1.32</td>
<td>0.81</td>
<td>0.84</td>
<td>0.27</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>0.76</td>
<td>0.70</td>
<td>0.90</td>
<td>0.92</td>
<td>1.09</td>
<td>0.72</td>
<td>0.78</td>
<td>0.34</td>
<td>0.64</td>
<td>0.60</td>
</tr>
<tr>
<td>Average</td>
<td>0.65</td>
<td>0.56</td>
<td>0.80</td>
<td>0.78</td>
<td>0.69</td>
<td>0.59</td>
<td>0.64</td>
<td>0.50</td>
<td>0.42</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>0.63</td>
<td>0.54</td>
<td>0.79</td>
<td>0.77</td>
<td>0.52</td>
<td>0.62</td>
<td>0.63</td>
<td>0.51</td>
<td>0.43</td>
<td>0.33</td>
</tr>
</tbody>
</table>
### Table 7. Average results of Naive Bayes classifier on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.73</td>
<td>0.66</td>
<td>0.66</td>
<td>0.69</td>
<td>0.16</td>
<td>0.67</td>
<td>0.43</td>
<td>0.33</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.83</td>
<td>0.83</td>
<td>0.83</td>
<td>0.90</td>
<td>0.62</td>
<td>0.83</td>
<td>0.27</td>
<td>0.66</td>
<td>0.66</td>
<td></td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.62</td>
<td>0.64</td>
<td>0.81</td>
<td>0.74</td>
<td>0.67</td>
<td>0.63</td>
<td>0.72</td>
<td>0.44</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.40</td>
<td>0.17</td>
<td>0.97</td>
<td>0.81</td>
<td>0.31</td>
<td>0.31</td>
<td>0.75</td>
<td>0.28</td>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.78</td>
<td>0.77</td>
<td>0.92</td>
<td>0.96</td>
<td>1.24</td>
<td>0.78</td>
<td>0.27</td>
<td>0.71</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.67</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.84</strong></td>
<td><strong>0.74</strong></td>
<td><strong>0.64</strong></td>
<td><strong>0.65</strong></td>
<td><strong>0.49</strong></td>
<td><strong>0.48</strong></td>
<td><strong>0.45</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Table 8. Average results of kNN classifier on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.71</td>
<td>0.60</td>
<td>0.60</td>
<td>0.65</td>
<td>0.16</td>
<td>0.61</td>
<td>0.47</td>
<td>0.24</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.85</td>
<td>0.51</td>
<td>0.76</td>
<td>0.36</td>
<td>0.53</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.43</td>
<td>0.46</td>
<td>0.72</td>
<td>0.66</td>
<td>0.35</td>
<td>0.44</td>
<td>0.68</td>
<td>0.17</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.49</td>
<td>0.26</td>
<td>0.98</td>
<td>0.84</td>
<td>1.48</td>
<td>0.41</td>
<td>0.27</td>
<td>0.26</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.81</td>
<td>0.82</td>
<td>0.93</td>
<td>0.94</td>
<td>1.40</td>
<td>0.82</td>
<td>0.29</td>
<td>0.75</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.64</strong></td>
<td><strong>0.58</strong></td>
<td><strong>0.80</strong></td>
<td><strong>0.79</strong></td>
<td><strong>0.78</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.59</strong></td>
<td><strong>0.51</strong></td>
<td><strong>0.39</strong></td>
<td></td>
</tr>
</tbody>
</table>
Table 9. Average results of SVM classifier on original (normal font) and selected sets (italic font)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>Sens</th>
<th>Spec</th>
<th>AUC</th>
<th>IS</th>
<th>F1</th>
<th>Prec</th>
<th>Brier</th>
<th>MCC</th>
<th>Inform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>0.73</td>
<td>0.59</td>
<td>0.69</td>
<td>0.69</td>
<td>0.09</td>
<td>0.59</td>
<td>0.37</td>
<td>0.27</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.73</td>
<td>0.58</td>
<td>0.58</td>
<td>0.68</td>
<td>0.08</td>
<td>0.58</td>
<td>0.38</td>
<td>0.25</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>Heart disease</td>
<td>0.83</td>
<td>0.83</td>
<td>0.90</td>
<td>0.53</td>
<td>0.38</td>
<td>0.83</td>
<td>0.24</td>
<td>0.65</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.86</td>
<td>0.36</td>
<td>0.76</td>
<td>0.32</td>
<td>0.53</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>Lung cancer</td>
<td>0.62</td>
<td>0.56</td>
<td>0.79</td>
<td>0.71</td>
<td>0.46</td>
<td>0.40</td>
<td>0.62</td>
<td>0.33</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.56</td>
<td>0.55</td>
<td>0.77</td>
<td>0.61</td>
<td>0.38</td>
<td>0.68</td>
<td>0.57</td>
<td>0.46</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>Primary tumor</td>
<td>0.36</td>
<td>0.22</td>
<td>0.97</td>
<td>1.19</td>
<td>0.32</td>
<td>0.22</td>
<td>0.94</td>
<td>0.20</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.24</td>
<td>0.12</td>
<td>0.96</td>
<td>0.71</td>
<td>0.85</td>
<td>0.26</td>
<td>1.05</td>
<td>0.11</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>Skin cancer</td>
<td>0.84</td>
<td>0.83</td>
<td>0.94</td>
<td>1.37</td>
<td>0.84</td>
<td>0.85</td>
<td>0.21</td>
<td>0.78</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.71</td>
<td>0.90</td>
<td>0.93</td>
<td>1.09</td>
<td>0.72</td>
<td>0.74</td>
<td>0.33</td>
<td>0.63</td>
<td>0.61</td>
</tr>
<tr>
<td>Average</td>
<td>0.68</td>
<td>0.61</td>
<td>0.82</td>
<td>0.81</td>
<td>0.67</td>
<td>0.61</td>
<td>0.60</td>
<td>0.48</td>
<td>0.45</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>0.61</td>
<td>0.55</td>
<td>0.79</td>
<td>0.76</td>
<td>0.55</td>
<td>0.60</td>
<td>0.59</td>
<td>0.53</td>
<td>0.40</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Fig. 3. Summary results and their standard deviations for Lung cancer dataset.
Fig. 4. Summary results and their standard deviations for Primary tumor dataset.

Fig. 5. Summary results and their standard deviations for Skin cancer dataset.


Information Systems and Soft Sets

Zofia Machnicka and Marek Palasinski

University of Information Technology and Management
Rzeszow, Poland
zmachnicka@wsiz.rzeszow.pl
mpalasinski@wsiz.rzeszow.pl

Abstract. It will be shown that each information system can be considered a soft set and each finite soft set can be considered an information system.

1 Basic Information

The notion of an information system is well established in the literature, e.g. [1, 4]. Informally, an information system consists of a finite nonempty set of objects and a finite nonempty set of attributes. Each attribute assigns to each object some value \( a(x) \), which is a member of a specific finite set \( V_a \), called the range of the attribute \( a \). Thus an attribute can be thought of as a function \( a : V \to V_a \), or as a sequence of elements of \( V_a \). Such a tuple is represented as a column in a matrix (or table) representing the information system. The rows of the matrix are labelled by objects and the columns by attributes. The matrix dimension is \( n \times p \), where \( n \) is the number of objects and \( p \) the number of attributes, see Table 1 below.

| \( a_0 \) | \( \cdots \) | \( a_k \) | \( \cdots \) | \( a_p \) |
| \( x_0 \) | \( a_{00} \) | \( \cdots \) | \( a_{k0} \) | \( \cdots \) | \( a_{p0} \) |
| \( \cdots \) | \( \cdots \) | \( \cdots \) | \( \cdots \) | \( \cdots \) |
| \( x_i \) | \( a_{0i} \) | \( \cdots \) | \( a_{ki} \) | \( \cdots \) | \( a_{pi} \) |
| \( \cdots \) | \( \cdots \) | \( \cdots \) | \( \cdots \) | \( \cdots \) |
| \( x_n \) | \( a_{0n} \) | \( \cdots \) | \( a_{kn} \) | \( \cdots \) | \( a_{pn} \) |

Here \( V = \{x_0, \ldots, x_n\} \), \( A = \{a_0, \ldots, a_p\} \), and \( a_{ki} \) is the value of the attribute \( a_k \) on \( x_i \), i.e., \( a_{ki} = a_k(x_i) \). Thus for each \( a \in A \), we have that \( a \in V_a \), and \( A \subseteq \bigcup \{V_a : a \in A\} \), where the union is disjoint. Similarly, as each object \( x \in V \) labels a row in the matrix, it can be identified with a tuple of values of attributes from \( A \) on this object. Hence the set of objects \( V \) can be identified with a subset of the Cartesian product of sets \( V_a \) (see [4])

\[ V \subseteq \prod_{a \in A} V_a. \]
Definition 1. An information system is a pair $S = (V, A)$, where $V$ and $A$ are nonempty finite sets, such that for each $a \in A$ there exists a finite set $V_a$ such that $a : V \rightarrow V_a$.

The elements of $V$ are called objects of the system $S$ and the elements of $A$ attributes of the system. An element $x \in V$ is identified with a tuple $x = \langle a(x) : a \in A \rangle$.

An information system $S$ is called two valued iff for every $a \in A$, the set $V_a$ has two elements, which we denote by 0 and 1. Expressions of the form

$$a_{i_1} = b_{i_1} \land \cdots \land a_{i_k} = b_{i_k} \rightarrow a_{i_t} = b_{i_t},$$

or

$$a_{i_1} = b_{i_1} \land \cdots \land a_{i_k} = b_{i_k} \rightarrow a_{i_t} \neq b_{i_t},$$

where $a_{i_1}, \ldots, a_{i_k}$, $a_{i_t}$ are attributes and $b_{i_1}, \ldots, b_{i_k}, b_{i_t}$ are their possible values, i.e., $b_{i_j} \in V_{i_j}$, are called rules. A rule of the first form is called a deterministic association rule while the one of the second form is called an inhibitory association rule. Mathematically, any rule is an implication and a true rule is a rule that is true as an implication for any object from $V$. A rule is realizable if its predecessor is true for at least one object from $V$.

It was shown in [2] that every information system can be equivalently replaced by a two valued information system. We recall the procedure in Section 3.

2 Soft Sets

The notion of a soft set has been proposed in [3] as a mathematical approach to uncertainty, alternative to that of a fuzzy set. It has subsequently provoked a lot of research.

Let $U$ be a set called a universe and let $E$ be another set, disjoint with $U$, called the set of parameters.

Definition 2. (Following [3]) Let $U$ be a set. A pair $\langle F, E \rangle$ is called a soft set over $U$ if and only if $F$ is a mapping from $E$ into the set of all subsets of the set $U$.

A soft set then can be seen as a parametrized family $\{ F(\varepsilon) : \varepsilon \in E \}$ of subsets of the set $U$ and the elements of $E$ are called parameters. In the terminology of [3], for each $\varepsilon \in E$, the set $F(\varepsilon)$ is called an $\varepsilon$-approximation of the soft set. If both $U$ and $E$ are finite then we will call the soft set $\langle F, E \rangle$ finite. In the next section we show that there is a natural connection between soft sets and information systems.

3 Connection Between Soft Sets and Information Systems

The following procedure of getting a two valued system $S^{(2)}$ from a given information system $S$ was presented at the HSI conference in 2010 ([2]) and we recall it here for completeness. Let $S = (V, A)$, where $A = (a_1, \ldots, a_n)$ and for each $i = 1, \ldots, n$
let the set of values of the attribute $a_i$ be $\{0, \ldots, k_i - 1\}$. We define the associated two-valued system $S^{(2)}$ by setting its attributes set to be

$$\{a_{10}, \ldots, a_{1k_i-1}, \ldots, a_{n0}, \ldots, a_{nk_n-1}\},$$

each one assuming one of the two values 0, 1. The set of objects remains the same. For any rule $r$ for $S$ we define the corresponding rule $r^{(2)}$ for $S^{(2)}$, replacing each expression of the form $a_i = j$ by $a_{ij} = 1$ and $a_{ij} = 0$ in other case and in case the rule is an inhibitory one expression $a_i \neq j$ by $a_{ij} = 0$. We then have the following lemma.

**Lemma 1.** For any rule $r$, if $r$ is true and realizable in $S$ then $r^{(2)}$ is true and realizable in $S^{(2)}$.

The proof of the lemma is straightforward, by contradiction. To show the converse, one needs to define new rules (each rule true and realizable in $S^2$, using the inverse translation, leads to a rule of a new kind).

**Remark 1.** In case of binary information systems, if two systems $S_1$ and $S_2$ are different then the sets of true and realizable implications associated with $S_1$ and $S_2$, respectively, are different (see [1, 2]).

From now on let $S^{(2)}$ denote a binary information system. So $S^{(2)}$ can be presented as in Table 1 above, where the value $a_{ki}$ of the attribute $a_k$ on $x_i$ is 0 or 1.

It is now easy to see that each information system $\langle V, A \rangle$ can be considered a soft set. It is enough to take into account that any subset $W \subseteq V$ can be uniquely identified with its characteristic function $f: V \to \{0, 1\}$ such that

$$f(x) = \begin{cases} 1, & \text{when } x \in W \\ 0, & \text{otherwise.} \end{cases}$$

For $k = 1, \ldots, p$, the $k$-th column in the table above can be considered a function $a_k : V \to \{0, 1\}$, with $a_k(x_i) = a_{ki}$, for each $j = 1, \ldots, n$, so each column determines a subset of $V$. Take the function $F : A \to \mathcal{P}(V)$ such that $F(a_k)$ is the subset of $V$ determined by the column in the table corresponding to the attribute $a_k \in A$. Then $\langle F, A \rangle$ is a soft set over $U$.

On the other hand, consider a finite soft set $\langle F, E \rangle$ over a universe $V$. Assume that $V = \{x_0, \ldots, x_n\}$ and $E = \{e_0, \ldots, e_p\}$, for some $n$ and $p$. To show that this soft set determines the unique information system, take $A = \{F(e) : e \in E\}$. For each $k = 0, \ldots, p$ and $i = 0, \ldots, n$ let $a_k$ be the characteristic function of the set $F(e_i)$. Then the values of $a_k$ form the $k$-th column in table 1. So the soft set $\langle F, E \rangle$ over $V$ is identified with the information system $\langle V, A \rangle$. Clearly, the two identifications are mutual inverses, so each soft set determines a unique information system and vice versa. This allows us to state the main result of this note.

**Theorem 1.** Given a finite set $V$ there is a one-one correspondence between the information systems of the form $\langle V, A \rangle$ and finite soft sets over the universe $V$.

As the finite set $V$ in the theorem is arbitrary, the theorem can be restated as
Corollary 1. Every information system can be regarded as a finite soft set and every finite soft set can be regarded as an information system.

Remark 2. As the anonymous referee points out, the main result can be alternatively explained in more general terms as follows. Given a family $A$ of zero-one sequences of a common length $n$, one can treat this family as the set of parameters $E$. Let $U$ be any set of $n$ elements. Then each parameter $\varepsilon \in E$, being a sequence of zeros and ones is a characteristic function of some subset of $U$, call this subset $F(\varepsilon)$. Then $F : E \rightarrow \mathcal{P}(U)$ and the pair $(F, E)$ is a soft set. The original set of sequences $A$ can be recovered from this soft set as the family of characteristic sequences of sets $F(\varepsilon)$, for all $\varepsilon \in E$.

References

An Approach for Resolving Conflicts in Automatic Medical Objects Classification

Przemyslaw W. Pardel¹, Jan G. Bazan¹, Jacek Zarychta², and Stanislawa Bazan-Socha³

¹ Interdisciplinary Centre for Computational Modelling, University of Rzeszow, Pigonia 1 Str., 35-310 Rzeszow, Poland
{ppardel, bazan}@ur.edu.pl

² Radiology consultant, Department of Pulmonology, Pulmonary Hospital, Gladkie 1 Str., 34-500, Zakopane, Poland
jzar@mp.pl

³ II Department of Internal Medicine, Jagiellonian University Medical College, Skawinska 8 Str., 31-066 Krakow, Poland
mmsocha@cyf-kr.edu.pl

Abstract. We describe a new approach for resolving conflicts for automatic identifying human organs from a medical CT images. The main premise of this approach is the use of classifier created with using two-level classifier and domain knowledge advisers decisions. We test our approach on multiple CT images of chest organs (trachea, lungs, bronchus) and demonstrate usefulness and effectiveness of the resulting classifications. The presented approach can be used to assist in solving more complex medical problems.

Key words: CT images, conflicts resolving, concept approximation, classifiers, decision trees, medical object recognition, object classification, domain knowledge, organs identifying, medical system

1 Introduction

A design of human–machine interface is the most important aspect of computer aided interpretation of medical image exams. Assists include decision support, reminder and navigation techniques to help avoid diagnosis errors, content-based data mining capabilities, and access to reference libraries. Human–machine systems should take advantage of computer capabilities to increase physicians interpretation capabilities [11].

An automatic identification of medical objects visualized by Computed Tomography (CT) imagery (e.g., organs, blood vessels, bones, etc.), without any doubt, could be useful, to support solving many complex medical problems using computer tools. Our approach is based on a two-level classifier. On the lower level, our approach uses a classical classifier based on a decision tree that is calculated on the basis of the local discretization (see, e.g., [7, 2]). This classifier is constructed and based on the features extracted from images using methods known from literature (see [6, 3] for more details). At a higher level of our two-level classifier, a collection of advisers works that is able to verify actions performed earlier by the lower-level classifier. This is possible by
using domain knowledge injected to advisers. Each of the adviser is constructed as a simple algorithm based on a logical formula, that on input receives selected information extracted from a tested image and a decision returned by the lower-level classifier, and the output returns confirmation or negation for the suggestion generated by the lower-level classifier. It consists in the fact, that in a situation where the decision taken by the lower-level classifier, is clearly incompatible with domain knowledge, the advisers suggestions and classifier decision are used to create conflict resolving classifier. Thanks to this, increases the accuracy of such the two-level classifier. To illustrate the method and to verify the effectiveness of presented classifiers, we have performed several experiments with the data sets obtained from Second Department of Internal Medicine, Collegium Medicum, Jagiellonian University, Krakow, Poland.

In the Section 2, we describe the problem of medical image understanding. Second section present conception of design a system for automatic medical objects classification. Finally, we present the complete structure of two-level classifier with method for resolving conflicts for the automatic classification of chest organs (see Section 4).

### Table 1. "Low-Level" Features

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>Distance to the first image in the series (mm)</td>
</tr>
<tr>
<td>SIZE</td>
<td>Object size</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Object width</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Object height</td>
</tr>
<tr>
<td>DFL</td>
<td>The distance from the object to the left edge of the image</td>
</tr>
<tr>
<td>DFT</td>
<td>The distance from the object to the top edge of the image</td>
</tr>
<tr>
<td>R1</td>
<td>Size of the object located in the region R1</td>
</tr>
<tr>
<td>R2</td>
<td>Size of the object located in the region R2</td>
</tr>
<tr>
<td>R3</td>
<td>Size of the object located in the region R3</td>
</tr>
<tr>
<td>R4</td>
<td>Size of the object located in the region R4</td>
</tr>
<tr>
<td>R5</td>
<td>Size of the object located in the region R5</td>
</tr>
<tr>
<td>R6</td>
<td>Size of the object located in the region R6</td>
</tr>
<tr>
<td>R7</td>
<td>Size of the object located in the region R7</td>
</tr>
<tr>
<td>R8</td>
<td>Size of the object located in the region R8</td>
</tr>
<tr>
<td>R9</td>
<td>Size of the object located in the region R9</td>
</tr>
<tr>
<td>CIRCUIT</td>
<td>Object circuit</td>
</tr>
<tr>
<td>TFACTOR</td>
<td>Object thickness factor</td>
</tr>
<tr>
<td>SFACTOR</td>
<td>Object shape factor</td>
</tr>
</tbody>
</table>

## 2 Medical Image Understanding

A process of radiological interpretation generally includes the understanding of medical image content resulting in recognition of possible pathology symptoms, most often
called detection, and assessment of comprehensive image information in a context of
current clinical case-knowledge. It involves image-based detection of disease, defining
disease extent, determining etiology of the disease process, assisting in designing of the
clinical management plans for the patient, based on imaging findings, and following
response to the therapy [12].

Table 2. "Domain Knowledge" Features

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CENTER</td>
<td>Center of the object region (e.g. R1, R2 . . .)</td>
</tr>
<tr>
<td>OIL</td>
<td>The number of objects on the right side</td>
</tr>
<tr>
<td>OIR</td>
<td>The number of objects on the left side</td>
</tr>
<tr>
<td>OIA</td>
<td>The number of objects above</td>
</tr>
<tr>
<td>OIB</td>
<td>The number of objects below</td>
</tr>
<tr>
<td>DTNLO</td>
<td>Distance to the nearest object on the left side</td>
</tr>
<tr>
<td>DTNRO</td>
<td>Distance to the nearest object on the right side</td>
</tr>
<tr>
<td>DTNAO</td>
<td>Distance to the nearest object above</td>
</tr>
<tr>
<td>DTNBO</td>
<td>Distance to the nearest object below</td>
</tr>
<tr>
<td>SNLO</td>
<td>The size of the nearest object on the left side</td>
</tr>
<tr>
<td>SNRO</td>
<td>The size of the nearest object on the right side</td>
</tr>
<tr>
<td>SNAO</td>
<td>The size of the nearest object above</td>
</tr>
<tr>
<td>SNBO</td>
<td>The size of the nearest object below</td>
</tr>
</tbody>
</table>

The other area of application of the automatic image understanding technique is
deep and requires a detailed analysis of particularly difficult images, especially in case
of doubts and difficulties in deciding on final diagnosis. A very important difference
between all traditional methods of automatic image processing (or recognition) and the
new paradigm for image understanding is that there is one directional scheme of the data
flow in the traditional methods; there are two-directional interactions between signals
(features) extracted from the image analysis and expectations resulting from the knowl-
edge of image contents, as given by experts (physicians). The results of all analyses
of medical image characteristics and objects visible in them, generated by computers,
allow the physician to base his/her reasoning on much more reliable and quantifiable
premises than just a visual assessment of that image, improving both the effectiveness
of his/her activities, and the feeling of reliability and security. Finally, the increasing
acceptance of techniques for the automatic recognition and classification of biological
objects distinguished in medical images can help the doctor make the right diagnostic
decisions, although these techniques sometimes require the doctor to be able to criti-
cally assess the automatically suggested categories, as every recognition technique car-
ries some level of error, while nothing excuses the doctor’s personal responsibility for
his/her decisions [9].

Medical image analysis is one of the areas of computer vision where domain knowl-
edge plays a very important role, because localized pixel information obtained from CT
images is often ambiguous and unreliable [5]. The history of knowledge-based medical image analysis is older than the history of practical usage of CT imaging. One of the early studies in knowledge-based medical image analysis was done by Harlow and Eisenbeisc [4] on radiographic image segmentation, when CT imaging was not yet available in hospitals. They proposed a top-down control system using a tree-structured model description containing knowledge about locations and spatial relations of parts/organs of the human body. In his thesis work, Selfridge [13] discussed image understanding systems in general and divided the causes of difficulties into problems of model selection, segmentation techniques, and parameter setting [5].

![Fig. 1. Examples of "domain knowledge" features extraction](image)

We conclude that the automatic detection of organs is the first step to understand medical images and it is necessary to begin the process of proper medical diagnosis support. To understand the CT image correctly, a computer should detect and recognize all medical objects located on the image by using domain knowledge. The knowledge about objects located in the medical image, allows the correct identification of areas
related to various medical problems. To understand medical image correctly, a computer should detect and recognize quite correctly all medical objects located on the image by using domain knowledge (extremely challenging task even for a man).

3 Conception of Design a System for Automatic Medical Objects Classification

3.1 A General Description

In order to understand the medical images, it is important to create a tool for understanding the interior of the human body on different levels of abstraction and tracking of interaction between the observed medical objects. The main issues to be addressed include problems with the quality of the medical image data, problems with domain knowledge descriptions and problems with modeling and exploration of the human body, which is very complex. The system should include the assumptions, such that the system should support work of doctors (not replace), expert always decide, system should allow for future sharing of knowledge and should naturally communicate in order to exchange knowledge (speech).

<table>
<thead>
<tr>
<th>Class</th>
<th>Object</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR</td>
<td>Trachea</td>
<td>671 (8.96%)</td>
</tr>
<tr>
<td>RL</td>
<td>Right lunge</td>
<td>1621 (21.64%)</td>
</tr>
<tr>
<td>LL</td>
<td>Left lunge</td>
<td>1616 (21.57%)</td>
</tr>
<tr>
<td>RB</td>
<td>Right main bronchi</td>
<td>190 (2.54%)</td>
</tr>
<tr>
<td>LB</td>
<td>Left main bronchi</td>
<td>211 (2.82%)</td>
</tr>
<tr>
<td>LL+RL</td>
<td>Object by gluing the left and right lungs</td>
<td>55 (0.73%)</td>
</tr>
<tr>
<td>OT</td>
<td>Other objects</td>
<td>3127 (41.74%)</td>
</tr>
</tbody>
</table>

3.2 "Low-Level” Features (LLF)

There is no "ideal set of features” which characterize the object. Features are selected individually depending on the recognized objects. In the computer analysis of the images, extracted features from the image, can be assigned to one of the categories, such as non-transformed structural characteristics (e.g. moments, power, amplitude information, energy, etc.), transformed structural characteristics (e.g. frequency and amplitude spectra, subspace transformation methods, etc.), structural descriptions (formal languages and their grammars, parsing techniques, and string matching techniques) and graph descriptors (e.g. attributed graphs, relational graphs, and semantic networks) described in detail in [6] and [3]. In this publication we call these features as Low-Level Features (LLF). In total, for the purposes of the experiments we define 18 LLF features (see Table 1).
3.3 "Domain Knowledge” Features (DKF)

To understand the image, it is also necessary to define the additional features that will define the acquired domain knowledge from experts. We call these features Domain Knowledge Features (DKF). DKF can be assigned to one of the categories, such as:

- features used to describe domain knowledge about the number of objects that surround an analyzed object,
- features used to describe domain knowledge about the distance from analyzed object to surrounding objects,
- features used to describe domain knowledge about the size of objects that surround an analyzed object,
- features used to describe domain knowledge about position of an object.

In total, for the purposes of the experiments we define 13 DKF features (see Table 2).

3.4 Medical Data

Our experiments were carried out on the data obtained from the clinical hospital Jagiellonian University Medical College in Kraków (the patients were diagnosed with asthma). The entire data set counted 26 patients (19 woman, 7 man). The average age of patients was 58.12 years (st.dev. 6.78 years, age range from 47 to 70 years). In all patients, volumetric CT torso scans were performed at both full inspiration and expiration with using 16-channel multi-detector CT scanner Toshiba (manufacturer’s model name:...
Aquilion). The acquired data were reconstructed using a kernel (FC86) with 1 mm increments. Images were stored in the Digital Imaging and Communications in Medicine (DICOM) format. For each patient was taken 300 to 400 images (full inspiration) with a resolution of 512x512 pixels. The total size of the data set for the experiment count 9655 CT images.

From all images we select every fifth image (20% of all images, 5mm increments) to pre-processing. As a result of the segmentation process, we acquired 7491 objects for experiments. For all the objects we set LLF and DKF features, further all objects are classified by an expert to one of the 7 classes (chest organs) presented in the Table 3.

Table 4. Comparison of the results Train&Test (Methods 1,2,3 and 4)

<table>
<thead>
<tr>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Acc</td>
</tr>
<tr>
<td>TR</td>
<td>94,00%</td>
</tr>
<tr>
<td>RL</td>
<td>97,31%</td>
</tr>
<tr>
<td>LL</td>
<td>97,64%</td>
</tr>
<tr>
<td>RB</td>
<td>78,55%</td>
</tr>
<tr>
<td>LB</td>
<td>76,77%</td>
</tr>
<tr>
<td>LL+RL</td>
<td>87,95%</td>
</tr>
<tr>
<td>OT</td>
<td>94,73%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method 3</th>
<th>Method 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Acc</td>
</tr>
<tr>
<td>TR</td>
<td>79,61%</td>
</tr>
<tr>
<td>RL</td>
<td>81,66%</td>
</tr>
<tr>
<td>LL</td>
<td>80,73%</td>
</tr>
<tr>
<td>RB</td>
<td>60,77%</td>
</tr>
<tr>
<td>LB</td>
<td>49,33%</td>
</tr>
<tr>
<td>LL+RL</td>
<td>22,38%</td>
</tr>
<tr>
<td>OT</td>
<td>75,12%</td>
</tr>
</tbody>
</table>

The entire data set was divided 20 times randomly into two sets - a set with training data and a set with test data (around 70% of the data getting into a training set - 18 patients, other (around 30%) into test set - 8 patients). Experiments 1,2,3 and 4 we conducted on these datasets. In 5-th experiment the entire data set was divided 20 times randomly into three sets - a set with training data and a set with test data (9 patients of the data getting into a training set, 9 patients into valid set and other 8 patients into test set).
4 Methods and Experiments

To verify the effectiveness of classification we prepare five methods. In methods one to four with using training data we built a classifier, which has been tested on test data. In method five with using training data we built a two-level classifier. Decision from this classifier has been used with advisers decisions to create second classifier on valid data. This classifier is used to resolve conflicts between domain knowledge advisers and two-level classifier.

We designed a classifier to the automatic classification of chest organs. In all methods we have implemented classifiers in the IMPLA (Image Processing Laboratory), which is a continuation of the RSES-lib library (forming the kernel of the RSES system [1]), in the field of image processing. The IMPLA has developed recently in Interdisciplinary Centre for Computational Modelling, University of Rzeszów, Poland.

4.1 Method 1

Method 1 is a method based on the decision tree with local discretization (LLF features, the quality of a given cut is computed as a number of objects pairs discerned by this cut and belonging to different decision classes, see, e.g., [7, 2]). The method gave good results (see Table 4).

4.2 Method 2

The second method was similar to the method 1 and based on the decision tree with local discretization. This method has used both LLF and DKF features (see Table 4).

4.3 Method 3

Method 3 was similar to the method 1 and based on the decision tree with local discretization. This method has used only DKF features (see Table 4).

4.4 Method 4: Two-level Classifier with ”advisers”

Method 4 is a method based on two-level classifier with ”advisers” (Figure 2). In this approach classification decision is dependent on suggestions of domain knowledge advisers. DKA suggest decisions based on domain knowledge e.g.”Left lung is located on the right side of medical image”, ”Object located on the left side of medical image is probably not a left lung”. We prepare 15 DKA for all chest organs. Advisers are divided into two groups:

- YES advised - Advisers to advise on YES e.g.”yes, this is probably the left lung” (6 DKA),
- NO advised - Advisers to advise on NO e.g.”no, this is probably not the left lung” (9 DKA).
Verify was followed on the basis of the DKF features e.g. if object center is located in region R3, R6 or R9 then YES adviser for right lunge take false decision. Advisers suggest what should be a decision (YES advisers) or suggested what should not be a decision (NO advisers). If any of the advisors suggested otherwise than the classifier (in some sense, the low-level classifier), decision was suspended (see [10]). All the decisions taken by the DKA pause the classifier decision where decisions are different. This is the direct reason for the decline coverage of the analyzed objects. By using domain knowledge we have obtained an improvement in the automatic classification of each chest organ. We presented the results of the experiments in the Table 4.

4.5 Method 5: Two-level Classifier with conflicts resolving

This method was similar to the method 4. In this experiment the entire data set was divided 20 times randomly into three sets - a set with training data, a set with validation
data and a set with test data (9 patients of the data getting into a training set, 9 patients into valid set and other 8 patients into test set). All the experiments (experiment with method 1 and 4 was prepared on merged training and validation data sets) we conducted on these datasets. Classification decision is dependent on suggestions of domain knowledge advisers (see [10]). Advisers decisions and classifier decisions has been used to create second classifier (conflict resolving classifier, C-RC) on valid data. This classifier is used to resolve conflicts between domain knowledge advisers and two-level classifier (Figure 3). The classifier C-RC is computed as a set of all decision rules with minimal number of descriptors (see, e.g., [2]). If any of the advisors suggested otherwise than
Table 5. Comparison of the results Train&Valid&Test (Methods 1, 4 and 5)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TR</td>
<td>91.44%</td>
<td>6.20%</td>
<td>97.59%</td>
<td>1.48%</td>
<td>92.92%</td>
<td>95.54%</td>
<td>3.77%</td>
<td></td>
</tr>
<tr>
<td>RL</td>
<td>97.46%</td>
<td>0.87%</td>
<td>98.29%</td>
<td>0.64%</td>
<td>99.14%</td>
<td>97.40%</td>
<td>0.82%</td>
<td></td>
</tr>
<tr>
<td>LL</td>
<td>97.42%</td>
<td>0.97%</td>
<td>98.43%</td>
<td>0.63%</td>
<td>98.95%</td>
<td>97.65%</td>
<td>0.90%</td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>77.02%</td>
<td>9.44%</td>
<td>89.71%</td>
<td>5.48%</td>
<td>84.53%</td>
<td>80.02%</td>
<td>5.17%</td>
<td></td>
</tr>
<tr>
<td>LB</td>
<td>76.67%</td>
<td>6.77%</td>
<td>83.19%</td>
<td>7.14%</td>
<td>89.99%</td>
<td>74.16%</td>
<td>5.32%</td>
<td></td>
</tr>
<tr>
<td>LL+RL</td>
<td>77.82%</td>
<td>23.58%</td>
<td>98.68%</td>
<td>2.69%</td>
<td>76.30%</td>
<td>95.74%</td>
<td>6.95%</td>
<td></td>
</tr>
<tr>
<td>OT</td>
<td>95.05%</td>
<td>1.47%</td>
<td>96.50%</td>
<td>1.36%</td>
<td>97.88%</td>
<td>95.80%</td>
<td>1.00%</td>
<td></td>
</tr>
</tbody>
</table>

the classifier, decision is taken with using conflict resolving classifier (Figure 4). By using conflict resolving classifier we have obtained an improvement in the automatic classification of almost each chest organ (except left bronchi) and coverage of the analyzed objects 100% and we improved classification stability. We presented the results of the experiments in the Table 5.

5 Conclusions and Further Works

The results of experiments performed on medical data sets indicate that the presented approach seems to be promising. The use of domain knowledge and the addition of a classifier resolving conflicts between advisers significantly improved the quality of the medical object identification. The next steps will focus on the use of time dependencies between medical images (object tracking in time).

The presented approach can be used in the future to support solving more complex medical problems. We plan to use the results of research, among other things, to treatment of an asthmatic airway remodeling (see, e.g., [8] for more details) and develop more advanced methods of using domain knowledge to construct more effective classifiers.

Acknowledgment

This work was partially supported by the Polish National Science Centre grant DEC-2013/09/B/ST6/01568 and by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland.

References


An Afficient Equivalence-Checking Algorithm for a Model of Programs with Commutative and Absorptive Statements

Vladislav Podymov
National Research University Higher School of Economics
3 Kochnovsky Proezd, Moscow 125319, Russia
valdus@yandex.ru

Abstract. We present an efficient equivalence-checking algorithm for a propositional model of programs with semantics based on (what we call) progressive monoids on the finite set of statements generated by relations of a specific form. We consider arbitrary set of relations for commutativity (relations of the form \( ab = ba \) for statements \( a, b \)) and left absorption (relations of the form \( ab = b \) for statements \( a, b \)) properties. The main results are a polynomial-time decidability for the equivalence problem in the considered case, and an explicit description of an equivalence-checking algorithm which terminates in time polynomial in size of programs.

Key words: program models, equivalence checking, semigroups, commutativity, left absorption

1 Introduction

The paper addresses the functional equivalence problem for imperative programs: given two programs, do determine whether they compute the same function. This problem is known to be undecidable in general (follows from Rice’s theorem [1]), but is highly important to have means to determine if given programs are equivalent: theoretically, the equivalence problem complexity shows what influence the structure of programs has on their meaning, i.e. which structural properties can be analyzed and which cannot; practically, equivalence-checking algorithms can be used in a huge variety of program analysis problems: optimization, verification, refactoring, obfuscation, and so on — in which any two programs (e.g. original and transformed, or ideal and real) are tested for having the same functionality.

One way to overcome the negativity of Rice’s theorem is to provide sufficient conditions for program equivalence. For example, an optimizing compiler performs a sequence of program transformation steps, each step preserves the original program functionality, and sufficient equivalence conditions may be used to prove transformation soundness and provide optimization algorithms known to be sound. The range of sound and efficient optimization capabilities is impressive: compilers have means to remove some unreachable or dead code, to restructure control flow graph and its linear code blocks, to optimize a variety of efficiency characteristics [2]. Nevertheless, there is still a long way to go. For example, the block of code
if (P(z)) {x=f(x); y=y+z;} else {x=g(x); x=v+z;}
if (!P(z)) y=y+z; else x=v+z;

and a much simpler block y=y+z; x=v+z; are equivalent, if we assume that the functions P, f, g are side-effect free and terminate, but simplifications of this kind are unlikely to be done automatically by modern compilers. To prove these blocks are equivalent, it is sufficient to use the following properties:

1. the value of P(z) remains unchanged;
2. the order of statements x = f(x); and y = y + z; has no effect on the result, and the same holds for x = v + z; and y = y + z;
3. the statement x = g(x); is dead if x = v + z; acts right after it, and the same holds for x = f(x); and x = v + z;

We refer to (2) as commutativity, and (3) as left absorption properties for statements, and focus on these two properties only. Suppose we have a statement a : (x_1, ..., x_n) = f(y_1, ..., y_k): "compute f on arguments y_1, ..., y_k, and put the result into x_1, ..., x_n". Let Mod(a) be the set \{x_1, ..., x_n\}, and Use(a) be \{y_1, ..., y_k\}. An easy way to extract commutativity and left absorption for statements is to analyze their Use- and Mod-sets: if Use(a) ∩ Mod(b) = Mod(a) ∩ Use(b) = Mod(a) ∩ Mod(b) = ∅, then a and b are commutative; if Mod(a) ⊆ Mod(b) and Mod(a) ∩ Use(b) = ∅, then b is left-absorptive for a.

To investigate the equivalence problem with respect to these properties we use the following approach: describe a program model which preserves the original program syntax but has a much simpler abstract semantics preserving some original semantical properties (programs in such a model are often called program schemata); propose an equivalence notion for schemata such that if two given schemata are equivalent, then the original programs are also equivalent; solve the equivalence problem for schemata, and use the resulting equivalence-checking algorithm to verify the original program equivalence.

We consider the model called "propositional programs" (PPs for short) proposed in [3, 4]. A PP is basically a finite automaton such that: vertices are labeled with statement symbols and correspond to program control locations in which a current data state is changed; transitions are labeled with all possible valuations for Boolean condition symbols; vertices are connected in the same way as in the original program. An interpretation which defines how a PP operates consists of: a set of abstract data states; a meaning of each statement symbol as a function for data state modification; a meaning of each condition symbol as a truth-value marking of data states. In general case these components are defined in terms of Kripke frames and models [5], but this paper focuses on commutativity and left absorption properties only: in this case data states are elements of a monoid on the set of statements generated by a set of relations of the form ab = ba (for commutativity) and ab = b (for left absorption) where a, b are statement symbols. For short, we call such a monoid a CA-monoid. We call two PPs equivalent in a CA-monoid M if for each interpretation based on M last data states of their executions are equal. All necessary definitions are given in the further sections.

The main goal of the paper is to provide an efficient equivalence-checking algorithm for PPs operating in CA-monoids. We call an algorithm efficient if it terminates.
in a polynomial time. Unfortunately, even for the trivial case when no commutativity and left absorption is assumed the equivalence problem for PPs is known to be co-NP-complete [6]. But for a fixed finite signature (alphabet of statements and conditions) for the trivial case PPs are known to be closely related to finite automata [7] and have an equivalence-checking algorithm having a very low time complexity [8]. Thus, hereafter a fixed finite signature is assumed, which means that we study the hardness with respect to structural program features. Another unpleasant fact is that equivalence-checking results for automata and PPs are not interchangeable for nontrivial cases, for example, for arbitrary commutativity the equivalence problem for automata is known to be undecidable [9], while for schemata it is decidable in polynomial time for a syntactically and semantically more general case [10].

CA-monoids considered in this paper are assumed to be progressive: we cannot obtain the same element twice appending non-neutral element to the right via composition. Note that for left-cancellative monoids the progressiveness is the same as the atomicity of the neutral element. The main result of the paper is an improvement of known decidability of the equivalence problem in the considered case [11] to polynomial-time decidability (theorem 2), and an explicit description of a polynomial-time decision algorithm (section 5). Note that if commutativity and left absorption properties are extracted in the way showed above, then the resulting CA-monoid is necessarily progressive. Moreover, [12] provides a simple-to-check progressiveness criterion for CA-monoids.

To make a few remarks on related work, we may mention polynomial-time decidability results on equivalence problems for one-counter automata [13] (NL-completeness), specific context-free languages [14, 15], specific recursive schemata [10], and decidability results for multitape automata [16] and $k$-valued transducers [17]. Note that there are much more results on the equivalence problem and quite a bit of them are mentioned.

The structure of the paper is as follows. Section 2 provides basic definitions. Section 3 discusses the main idea of the analysis technique and basic properties of the main object to be analyzed — a graph of consistent computations (GCC). The result of section 4 is a deep analysis of a GCC which allows to describe an polynomial-time equivalence-checking algorithm. The main theorem and the algorithm description are given in section 5.

2 Basic Definitions

Hereafter $\mathfrak{A}$ and $\mathcal{L}$ are finite alphabets of basic assignment statements, or actions for short, and logical conditions respectively. A propositional program (PP)

\[ \pi = (L, en, ex, T, B) \]

consists of: a finite set of control locations $L$ including entry $en$ and exit $ex$; a transition function $T : (L \setminus \{ex\}) \times \mathcal{L} \rightarrow L$; a binding function $B : L \rightarrow \mathfrak{A}$. We write $l \xrightarrow{\delta} \pi \ l'$ and $l \rightarrow \pi \ l'$ instead of $T(l, \delta) = l'$ for readability. We define the size $|\pi|$ of a PP $\pi$ to be the number of its control locations: $|\pi| = |L|$. The meaning of a PP is briefly discussed in the introduction except of the following: a logical condition may be understood as
a valuation for all Boolean conditions, which means that exactly one logical condition holds for each abstract data state.

To define semantics of PPs in general case we should use Kripke frames and models [5], but in the considered case (commutativity and left absorption for actions) we simplify it using the notation of semigroup theory. A semigroup \((S, \ast)\) consists of an arbitrary set \(S\) of elements, and a binary associative operation on \(S\). A monoid \(M = (S, \ast, \varepsilon)\) is a semigroup \((S, \ast)\) together with its neutral element \(\varepsilon\). To define how a PP operates we use a monoid on the set of actions: each action from \(\mathcal{A}\) is an element of the monoid, and each other element \(s\) can be decomposed into actions \((s = a_1 \ast \cdots \ast a_k, k \geq 0, a_i \in \mathcal{A}\) for \(1 \leq i \leq k\)). We write \([a_1 \ldots a_k]\) to denote such \(s\). Hereafter we use only monoids on the set of actions. An interpretation \(I = (M, \xi)\) for a PP consists of a monoid \(M = (S, \ast, \varepsilon)\), and a valuation \(\xi : S \rightarrow \mathcal{L}\) for logical conditions.

A path for a program \(\pi\) is a sequence of program locations of the form \(l_1 \rightarrow_\pi l_2 \rightarrow_\pi \ldots\). For paths \(pt', pt''\), by \(pt' \rightarrow pt''\) we denote a path which starts with a path \(pt'\) and continues with a path \(pt''\). A path is called a trace if \(l_1\) is the entry of \(\pi\). A path is full if it either is infinite or leads to the exit of \(\pi\). A full trace is called a run. A trace \(l_1 \delta_1 \rightarrow_\pi \ldots l_2 \delta_2 \rightarrow_\pi \ldots\) is called an I-trace for an interpretation \(I\) if \(\delta_i = \xi ([B(l_1) \ldots B(l_i)])\) for \(i \geq 1\). To simplify the notation, we write \(B(l_1, \ldots, l_i)\) along with \(B(l_1) \ldots B(l_i)\), and \([l_1, \ldots, l_i]\) along with \([B(l_1), \ldots, B(l_i)]\). The result of a finite I-run \(rn\) is the element \([rn]\).

An infinite I-run loops and has the indefinite result \(\bot\). Two programs are I-equivalent if the results of their I-runs are equivalent. For a monoid \(M\) two programs are \(M\)-equivalent if they are \(I\)-equivalent for each interpretation \(I = (M, \xi)\).

The equivalence problem investigated in this paper is the \(M\)-equivalence problem for PPs where \(M\) is a monoid: given two PPs, to check if they are \(M\)-equivalent. The problem is solved for specific monoids \(M\), which we call progressive CA-monoids. A monoid is progressive if for each triple of its elements \(s, s_1, s_2\), if \(s_1 \not\equiv \varepsilon\) or \(s_2 \not\equiv \varepsilon\), then \(s \not\equiv s_1 \ast s_2\). Note that the elements of a progressive monoid are partially ordered w.r.t. its operation, i.e. \(s_1 \ast s_2 \not\equiv s_1\) for \(s_2 \not\equiv \varepsilon\).

In other words, if we introduce the following order \(\leq: s_1 \leq s_2\) iff \(\exists s\). \(s_2 = s_1 \ast s\) — then a monoid is ordered iff \(\leq\) is a non-strict partial order. Along with \(\leq\) we use naturally induced orders \(<, \geq, \rangle\), as well as \(s_1 \| s_2\) to denote the incomparability of \(s_1, s_2\). A CA-monoid is a monoid generated by an arbitrary set of relations of the form \(ab = ba\) and \(ab = b\ (a, b \in \mathcal{A})\); the former one is a commutativity relation, the latter is a (left) absorption relation. A monoid generated by a set of relations can be defined as follows:

- each element is a set of words over the alphabet \(\mathcal{A}\);
- \(g \in [h]\) iff \(g\) can be obtained from \(h\) with a finite number of rewritings using generating relations (a subword from the left part of a relation to the right part and vice versa).

The main result of the paper is the polynomial-time decidability of the \(M\)-equivalence problem for PPs for an arbitrary progressive CA-monoid \(M\) together with an explicit description of a decision algorithm. The description of the algorithm depends on a description of \(M\), i.e. on a set of commutativity and absorption relations defining \(M\). The algorithm is polynomial in time in size of input programs.
3 Graph of Consistent Computations

To simplify notation, hereafter we assume an arbitrary progressive CA-monoid $M = (S, *, e)$ to be defined, as well as PPs

$$
\pi_1 = (L_1, en_1, ex_1, T_1, B_1)
$$

and

$$
\pi_2 = (L_2, en_2, ex_2, T_2, B_2).
$$

The main idea of how we analyze the $M$-equivalence problem for PPs is similar to those developed in [3, 4], [10], [17] and can be described as follows. Suppose we have an interpretation $I = (M, \xi)$. We construct the $I$-traces $tr_1, tr_2$ of $\pi_1$ and $\pi_2$ in the following manner: we start with one-element $I$-traces of $\pi_1, \pi_2$; if $|tr_1| = |tr_2|$, then we add the next location to both traces; if $|tr_1| > |tr_2|$, then we add the next location to $tr_2$; otherwise we add the next location to $tr_1$. If any of the traces $tr_i$ is an $I$-run, then we do not add anything to it. As the result we have a sequence of pairs of traces leading either to the $I$-runs of $tr_1, tr_2$, or to an $I$-run and an $I$-trace. Consider any such pair $(tr_1 \rightarrow l_1, tr_2 \rightarrow l_2)$. To find what results can be obtained with computations starting in these traces, all we need to know is last program locations of these traces and data states they lead to. Thus, we replace this pair with a tuple $(l_1, l_2, [tr_1 \rightarrow l_1], [tr_2 \rightarrow l_2])$.

Hereafter we write $[s]$ to denote the set of all minimal members of $s$, $s \in S$: $\{g \mid g \in s, |g| = \min_{u \in s} |u|\}$ — and $\|s\|$ to denote the length of words in $[s]$. Also, we write $[h]$ instead of $[[h]]$, and call a word $h \in \mathbb{A}^*$ a minimal member if $h \in [h]$.

Let $(l_1, l_2, s_1, s_2)$ be a tuple as described above, and $s$ be a longest common prefix $s$ of $s_1$, $s_2$: $s * s'_1 = s_1; s * s'_2 = s_2; \|s\|$ has a maximal possible value. To check the equivalence of programs, we do not need the results themselves, but to check if the results are equal. Thus, we replace $s_1, s_2$ with $s'_1, s'_2$ in the tuple. Due to the construction rules for tuples, $\|s'_2\| \leq 1$, which means that $s'_2 = [\alpha]$ for $\alpha \in \mathbb{A} \cup \{\lambda\}$. Then we replace $(l_1, l_2, s_1, s_2)$ with a reduced tuple $(l_1, l_2, s'_1, \alpha)$. Looking over all possible interpretations $I$, we connect all possible tuples in a graph $\Gamma$, which we call a graph of consistent computations (GCC) in the following way. The vertices of $\Gamma$ are all possible tuples $(l_1, l_2, s, \alpha)$, where $l_1, l_2$ are locations of $\pi_1, \pi_2$, $s$ is a data state, and $\alpha \in \mathbb{A} \cup \{\lambda\}$. If there exists an interpretation $I$ such that tuples $t_1, t_2$ are what we get from adjacent pairs of traces obtained with the construction rules for $I$, then $t_1$ and $t_2$ are connected with an edge. To eliminate the enumeration of all interpretations, we define $\Gamma$ using $M$, $\pi_1$, and $\pi_2$ only, and then prove that to analyze $\Gamma$ is sufficient to understand whether $\pi_1, \pi_2$ are $M$-equivalent.

To explain the term “consistent”, we define the notion of consistency for traces of PPs in the following way. Traces of $\pi_1, \pi_2$ are ($M$-)consistent if they are $I$-traces for some interpretation $I = (M, \xi)$. Thus, an alternative description of the $M$-equivalence problem for PPs is: given two PPs, to check if for each pair of $M$-consistent runs their results are equal. The following proposition shows how to check trace consistency without enumerating of all possible interpretations.

**Proposition 1 ([3]).** Let $M$ be an progressive monoid, and $tr_1, tr_2$ be traces of $\pi_1, \pi_2$. Then $tr_1, tr_2$ are $M$-consistent iff for any their prefixes $tr'_1 \rightarrow_{\pi_1} l_1, tr'_2 \rightarrow_{\pi_2} l_2$ holds: if $|tr'_1| = |tr'_2|$, then $\exists \delta. tr'_1 \overset{\delta}{\rightarrow}_{\pi_1} l_1$ and $tr'_2 \overset{\delta}{\rightarrow}_{\pi_2} l_2$. 
Now we give an alternative definition of $\Gamma$, which does not require enumeration of all interpretations. The nodes of $\Gamma$ are 4-tuples of the form $v = (l_1, l_2, s, \alpha)$ where $l_i \in L_i$, $s \in S$, and $\alpha \in \mathfrak{A} \cup \{\lambda\}$. The root of $\Gamma$ is the node $(en_1, en_2, s, \alpha)$ where:

- if $[B_1(en_1)] \leq [B_2(en_2)]$, then $s = \epsilon$, otherwise $s = [B_1(en_2)]$; if $[B_2(en_2)] \leq [B_1(en_1)]$, then $\alpha = \lambda$, otherwise $\alpha = B_2(en_2)$. Some of the nodes $(l_1, l_2, s, \alpha)$ are terminal and have no outgoing edges:

1. $l_1 = ex_1$, $\alpha \neq \lambda$;
2. $l_2 = ex_2$, $s \neq \epsilon$;
3. $l_1 = ex_1, l_2 = ex_2$, and either $s_1 \neq \epsilon$, or $\alpha \neq \lambda$.

For cases (1), (2) (which are not mutually exclusive, but exclude (3)) we call a node disproving (the equivalence of PPs); Edges outgoing from a nonterminal node $v = (l_1, l_2, s, \alpha)$ carry the following labels (one edge for each $\delta \in \Sigma$): $(\delta, \delta)$ if $l_1 \neq ex_1$, $l_2 \neq ex_2$, and $s = [\alpha] = \epsilon$; $(\epsilon, \delta)$ if $l_2 \neq ex_2$, $\alpha = \lambda$, and either $l_1 = ex_1$ or $s \neq \epsilon$: $(\delta, \epsilon)$ otherwise. Here $\epsilon$ is a special symbol, $\epsilon \notin \Sigma$. To finish the definition of $\Gamma$, we describe the node $v' = (l'_1, l'_2, s', \alpha')$ s.t. $v \xrightarrow{(\sigma_1, \sigma_2)} v'$. If $\sigma_1 = \epsilon$, then $l'_1 = l_1$ and $s'' = s$, otherwise $l'_1 \xrightarrow{\sigma_1} l'_1$ and $s'' = s * [B_1(l'_1)]$. If $\sigma_2 = \epsilon$, then $l'_2 = l_2$ and $\alpha'' = \alpha$, otherwise $l'_2 \xrightarrow{\sigma_2} l'_2$ and $\alpha'' = B_2(l'_2)$. If $s'' = [\alpha] * s''$, then $s' = s''$ and $\alpha' = \lambda$, otherwise $s' = s''$ and $\alpha' = \alpha''$.

As the GCC is defined, we should prove that it contains as many traces as needed to check whether $\pi_1, \pi_2$ are $M$-equivalent. To do it, we show the correspondence between paths in $\Gamma$ and traces of $\pi_1, \pi_2$ (lemmas 1–3), and then show what paths should we search for to check whether $\pi_1, \pi_2$ are $M$-equivalent (theorem 1).

To shorten the notation, we give three more definitions. An r-path is a path in $\Gamma$ originating from its root. Projections $pr_1(\omega), pr_2(\omega)$ of an r-path $\omega$ are defined as follows. $pr_i((en_1, en_2, s, \alpha)) = en_i$. Let $\omega = \omega' \xrightarrow{(\sigma_1, \sigma_2)} (l_1, l_2, s, \alpha)$. Then:

- $pr_i(\omega) = pr_i(\omega')$ if $\sigma_i = \epsilon$;
- $pr_i(\omega) = pr_i(\omega') \xrightarrow{\sigma_i} l_i$ otherwise.

For an infinite r-path $\omega$ we define $pr_i(\omega)$ as the shortest r-path s.t. for each finite prefix $\omega''$ of $\omega$ holds: $pr_i(\omega'')$ is a prefix of $pr_i(\omega)$. An r-path $\omega$ is disproving in two cases: it leads to a disproving node; it is infinite, and for some its (infinite) tail $\omega_i$ and some $i \in \{1, 2\}$ holds: for a label $(\sigma_1, \sigma_2)$ of any edge of $\omega_i$ holds $\sigma_i = \epsilon$, and for any node $(l_1, l_2, s, \alpha)$ of $\omega_i$ the location $ex_i$ is reachable from $l_i$.

**Lemma 1.** Let $\omega$ be an r-path. Then $pr_1(\omega)$ and $pr_2(\omega)$ are consistent traces of $\pi_1, \pi_2$. Moreover, if $\omega$ leads to a node $(l_1, l_2, s, \alpha)$, then the following holds: there exists $s' \in S$ s.t. $s' * s = [pr_1(\omega)]$ and $s' * [\alpha] = [pr_2(\omega)]$; if $s \neq \epsilon$ and $\alpha \neq \lambda$, then $s$ and $[\alpha]$ are incomparable.

**Lemma 2.** Let $rn_1, rn_2$ be consistent runs of $\pi_1, \pi_2$. Then there is an r-path $\omega$ s.t. $pr_1(\omega)$ is a prefix of $rn_i$ (for $i \in \{1, 2\}$), and either $\omega$ is infinite, or it leads to a terminal vertex.
Lemma 3. Let $\omega$ be an $r$-path, $i \in \{1, 2\}$, and $rn_i = pr_i(\omega) \rightarrow_{\pi_i} pt$ be a run of $\pi_i$. Then $rn_i$ and $pr_{3-i}(\omega)$ are consistent traces.

Lemmas 1, 2 follow from proposition 1, the left-cancellativity of $M$, and the definition of $\Gamma$. To prove lemma 3 by contradiction, we note that if $rn_i$ and $pr_{3-i}(\omega)$ are not consistent, then due to proposition 1 and lemma 1 there exist a proper prefix $pt'$ of a path $pt$ and a proper prefix $tr'_{3-i}$ of $pr_{3-i}(\omega)$ s.t. $[pr_i(\omega) \rightarrow_{\pi_i} pt'] = [tr'_{3-i}]$. Then by lemma 1 there does not exist a prefix $\omega'$ of $\omega$ s.t. $pr_i(\omega) = tr'_{3-i}$, which is impossible by the definition of $pr_i$.

Theorem 1. PPs $\pi_1, \pi_2$ are $M$-equivalent iff $\Gamma$ contains no disproving paths.

Proof. Necessity. If $\Gamma$ contains a finite disproving path $\omega$, then by lemma 3 the traces $pr_1(\omega), pr_2(\omega)$ can be extended to consistent runs of $\pi_1, \pi_2$, and by lemma 2 the results of these runs are distinct. If $\Gamma$ contains an infinite disproving path $\omega$ and $i$-th components of edge labels of some its tail are equal to $e$, then the trace $pr_i(\omega)$ can be extended to a finite run $rn_i$ of $\pi_i$, while $pr_{3-i}$ is an infinite run of $\pi_{3-i}$, and by lemma 3 the runs $rn_i$ and $pr_{3-i}$ are consistent and have distinct results.

Sufficiency. If $rn_1, rn_2$ are consistent runs of $\pi_1, \pi_2$ with distinct results, then by lemma 2 we obtain an $r$-path $\omega$ which is either finite and leads to a disproving vertex due to lemma 1, or infinite, in which case exactly one of runs $rn_1, rn_2$ is infinite, and $\omega$ is an infinite disproving path.

Note that theorem 1 does not solve the $M$-equivalence problem for PPs, as $\Gamma$ is infinite in general case. The next section provide means to build a polynomial-time traversal of $\Gamma$ sufficient to solve the equivalence problem.

4 Analysis of Graph of Consistent Computations

The main idea of the further GCC analysis is to partition its nodes into a finite number of classes and prove that during the traversal it is sufficient to visit polynomially many nodes in each class, and after it we state either that $\pi_1$ and $\pi_2$ are not $M$-equivalent (lemma 7) or that other nodes in the class may be ignored in the further traversal (lemma 8). To do it, we introduce and analyze the notion which we called an absorption effect. An (absorption) effect induced by an element $s \in S$ is a function $\alpha_s : S \rightarrow S$ showing how the elements of $S$ are changed under “pressure” of absorption of $s$ appended to the right. Formally, $\alpha_s(s_1) = \arg \min_{s_2: s_1 \ast s = s_2 \ast s} ||s_2||$. To prove the soundness of the definition, we state a (what we call) commutativity of minimal: if $h, g$ are minimal members of the same element of $M$, then $g$ can be obtained from $h$ with commutativity rewritings only (without any absorption). The soundness is provided by the following lemma.

Lemma 4. An equation $X \ast s = s'$ has at most one solution $s''$ on $X$ s.t. $||s''||$ has minimal possible value among all solutions.

Proof. Suppose we have at least two such solutions: $s''_1$ and $s''_2$: $s''_1 \ast s = s' = s''_2 \ast s$. Let $h_1 \in [s''_1], h_2 \in [s''_2]$, and $h \in [s]$. Due to the commutativity of minimal, we
can obtain \(h_2 h\) from \(h_1 h\) using commutativity relations only. By symmetry of commutativity relation, we get the derivation of \(h h_2\) from \(h h_1\) (\(g\) is a reversed \(g\)). By the left-cancellativity of \(M\) and the same symmetry, \(h_1\) can be rewritten to \(h_2\) with commutativity only, which means \(s_1'' = s_2''\).

The next step is to introduce a partial order on the set of all effects (which we denote by \(\leq\)): \(\leq\) is a partial order if \(\leq\) is a reversed \(g\). By the inductive minimality, an effect \(\alpha\) determines a set of symbols deleted from a minimal member: if \(\alpha\) is a minimal member, \(\alpha\) determines a set of symbols deleted from \(\alpha\) by deleting some symbols. If there exist \(\alpha_1, \alpha_2\) such that \(\alpha_1 \leq \alpha_2\) and \(\alpha_2 \leq \alpha_1\), then \(\alpha_1\) deletes at least those symbols which are deleted by \(\alpha_2\) and vice versa. It means that for any minimal member \(h\) the effects \(\alpha_1\) and \(\alpha_2\) delete the same symbols from \(h\), and thus \(\alpha_1([h]) = \alpha_2([h])\).

Another pleasant property of the effects is that they can be (in some sense) computed by a finite automaton. It means that the set is finite, and that having a short description of an effect \(\alpha\) and an action \(a\), we can easily compute the description of \(\alpha_{a^s[a]}\). An automaton \(A = (Q, q_0, T_A, S_A, M_A)\) (over the alphabet \(\mathbb{A}\)) consists of: a finite set of states \(Q\) including an initial state \(q_0\); a transition function \(T : Q \times \mathbb{A} \rightarrow Q\); a set of labels \(S_A\); a marking function \(M_A : Q \rightarrow S_A\). We denote by \(A(q, h)\) a state to which \(A\) jumps reading the word \(h \in \mathbb{A}^*\), and use the following shortening: \(A(h) = A(q_0, h)\); \(M_A(q, h) = M_A(q, h)\); \(M_A(h) = M_A(h)\). The notion of automaton used in this paper is a minor generalization of a well-known finite deterministic acceptor with the only difference: instead of answers “yes” and “no”, here \(S_A\) is the set of all possible answers. We put the following meaning in the phrase “to compute an effect”: \(|S_A| = 1\); \(\alpha_{[h]} = \alpha_{[g]}\) iff \(M_A(h) = M_A(g)\); \(M_A([h]) = M_A([g])\). We call an automaton satisfying all these properties an \(\alpha\)-automaton. Note that intuitively the best way to define such an automaton is to say \(S_A = \) \(,\) but formally the set cannot be used explicitly in any algorithm as it contains functions with an infinite domain. The description of an \(\alpha\)-automaton is based on an automata-based recognition of absorption: there exists an automaton with the following two properties: reading an input word \(h\), it enters a state labelled with the set \(\{a \in \mathbb{A} \mid a \ast [h] = [h]\}\); reading words \(h, g\) s.t. \(h^{-} = [g^{-}]\), it enters the same state.

Lemma 6. For any progressive CA-monoid there exists an \(\alpha\)-automaton.

Proof. We start with an automaton \(A\) stated by the automata-based recognition of absorption. Then we divide the states of \(A\) into equivalence classes: \(q, q'\) are equal iff \(\forall h \in \mathbb{A}^*. M_A(q, h) = M_A(q, h)\). Note that we have a finite number \(N\) of such classes. Changing the labels of \(A\) to \(\{1, \ldots, N\}\), enumerating the equivalence classes, and assigning the label \(i\) to the \(i\)-th class, we get an automaton \(A'\) s.t. \(M_A'(h) = M_A'(g)\) iff \(\alpha_{h^{-}} = \alpha_{g^{-}}\). Then by changing the labels again we construct exactly \(N\) finite-state
acceptors, the $i$-th of which says if the original answer is $i$. Using classical automata-theoretic results [8], we reverse the language of each acceptor, construct their Cartesian product and reassign labels back to $\{1, \ldots, N\}$, and the result is exactly a required $\varepsilon$-automaton: $M_A(h) = M_A(g)$ iff $\varepsilon_h = \varepsilon_g$, and all states were constructed for elements of $S$, thus all parts of the definition are satisfied.

Hereafter we assume to be given an $\varepsilon$-automaton $A = (Q, q_0, T_A, S_A, M_A)$. By the definition of $A$, we write $A([h])$ along with $A(h)$. The last two notions required for the analysis of $\Gamma$ are an effect evolution and a generalized effect evolution. Intuitively, these notions show how an absorption effect is modified during a program run from some intermediate run location. An effect evolution $\varepsilon^h_k$ along the word $h = a_1 \ldots a_k \in \mathcal{A}^*$ induced by $s \in S$ is defined as follows. We start with the sequence $(\varepsilon_s, \varepsilon_s[a_1], \ldots, \varepsilon_s[h])$. Then we delete adjacent duplicates. The result is exactly $\varepsilon^h_k$. Note that any evolution is a chain in a finite (lemma 6) partially ordered (lemma 5) set, and thus the number of evolutions is finite. A generalized effect evolution $\varepsilon^{h'}_k$ along the word $h = a_1 \ldots a_k \in \mathcal{A}^*$ induced by a state $q \in Q$ is defined in the same way except we start with the sequence of pairs $((\varepsilon_s, \varepsilon_{M_A(q)}), (\varepsilon_{[a]}, \varepsilon_{M_A(q,a)}), \ldots, (\varepsilon_{[a]}, \varepsilon_{M_A(q,a)}))$. For the same reasons the set of all generalized evolutions is finite.

There is an important property of evolutions. For any full path $pt$ for a PP $\pi = (L, c_n, e_x, T, B)$ and any $s \in S$ there exists a full path $pt'$ (which we call reduced) s.t.: $\varepsilon^{B(pt)}_s = \varepsilon^{B(pt')}_s$; if $pt$ is finite, then $|pt'| \leq |L| \cdot |Q|$; if $pt$ is infinite, then $pt' = pt'_1 \rightarrow pt'_2 \rightarrow pt'_3 \rightarrow \ldots$ where $|pt'_1| + |pt'_2| \leq |L| \cdot |Q|$. To prove the existence of a reduced path $pt'$, it is sufficient to: mark $pt$ with states of $A$ starting with $A(s)$ modifying it with the actions of $pt$; delete loops between two repeated pairs of a program location and an automaton state; for an infinite path, replace an infinite tail corresponding to the last effect in the evolution with the repetition of a loop described above. The same property holds for generalized evolutions with an upper bound $|L| \cdot |Q|^2$ instead of $|L| \cdot |Q|$. The following lemmas are the result of the analysis of $\Gamma$, as they provide a finite partitioning of the nodes of $\Gamma$ required in the equivalence-checking algorithm.

**Lemma 7.** Let $q \in Q$; $l_1 \in L_1$; $l_2 \in L_2$; $\alpha \in \mathcal{A} \cup \{\lambda\}$; $m = \lfloor Q \cdot |\pi_1| \cdot |\pi_2| \rfloor + 2$; pairwise-distinct GCC nodes $(l_1, l_2, s_i, \alpha), i \in \{1, \ldots, m\}$, be reachable from the GCC root; $A(s_1) = \cdots = A(s_m) = q$; $l_1 \rightarrow pt_i$ be a full path of $\pi_i$ $(i \in \{1, 2\})$, and at least one of these paths is finite; $\varepsilon^{\alpha^{B_1}(pt_1)}_{a_1} = (a_1^1, a_1^2, \ldots, a_1^k)$; $\varepsilon^{\alpha^{B_2}(pt_2)}_{a_1} = (a_1^1, \ldots, a_p); 1 \leq k' \leq k$; the elements $a_i^{k'}(s_i), i \in \{1, \ldots, m\}$, be pairwise distinct; $\{a_1^1, a_p\} \cap \{a_1^{k'+1}, \ldots, a_1^k\} = \emptyset$.

Then $\pi_1$ and $\pi_2$ are not $M$-equivalent.

**Proof.** The main idea is to show that at least for one of the nodes $w_j$ traces of a projection of a path $\omega_j$ leading to this node can be expanded into consistent runs with different results. We show in details only one case (for other cases a proof technique works in the same way, though it looks a bit more complex): $pt_1$ and $pt_2$ are finite, $k' = k$. W.l.o.g. we assume $pt_1$ and $pt_2$ to be reduced paths, thus $|pt_i| \leq |\pi_i| \cdot |Q|^{3-i}$. By proposition 1 the only way for the runs not to be consistent is if their proper prefixes lead to a common element $s$, and next steps are made with distinct conditions $\delta_1, \delta_2$.
— we call it a collision. By lemma 3, for each \( \omega_p \) these prefixes are not shorter than the projections of \( \omega_p \). Thus, we have \([pr_1(\omega_p) \rightarrow pt_1] = [pr_2(\omega_p) \rightarrow pt_2]\) for some proper prefixes \( pt_1', pt_2' \) of \( pt_1 \) and \( pt_2 \). But as the elements \( x_k^1(s_p), x_k^2(s) \) are distinct for \( p \neq r \), the elements \([pr_1(\omega_p) \rightarrow pt_1'] \) and \([pr_1(\omega_p) \rightarrow pt_1'] \) are also distinct (as \( M \) is left-cancellative), while \([pr_2(\omega_p) \rightarrow pt_2'] = [pr_2(\omega_r) \rightarrow pt_2']\). Thus, a collision for different indexes \( p, r \) cannot happen for the same prefixes \( pt_1', pt_2' \) simultaneously.

On the other hand, we have at most \(|pt_1| \cdot |pt_2| \leq |\pi_1| \cdot |\pi_2| \cdot |Q|^3\) possible pairs of lengths for these prefixes, which means that at least for two indexes \( j', j'' \) there are no collisions: in other words, expansions for these indexes are consistent. Based on the same thought about distinction of \( x_k^1(s_{j'}), x_k^2(s_{j''}) \) as above for \( p, r \), we conclude that at least for one of indexes \( j', j'' \) the results of computations are distinct.

**Lemma 8.** Let: \( q \in Q \); \( l_1 \in L_1 \); \( l_2 \in L_2 \); \( \alpha \in \mathfrak{A} \cup \{ \lambda \} \); \( m = |Q|^3 \cdot |\pi_1| \cdot |\pi_2|^2 + 2 \); pairwise-distinct GCC nodes \( v_i = (l_1, l_2, s_i, \alpha) \), \( i \in \{1, \ldots, m\} \), are reachable from the GCC root; \( A(s_1) = \cdots = A(s_m) = q \); \( l_1 \rightarrow pt_1 \) is a full path of \( \pi_i \), \( i \in \{1, 2\} \); \( \tilde{\alpha}_q^{B_1(\pi_i)} = ((a_1^2, a_2^2, \ldots, a_k^2); \tilde{\alpha}_q^{B_2(\pi_2)} = (a_1, \ldots, a_s); 1 \leq k < k', 1 \leq s' < s; \) the elements \( a_k^1(s_i) \), \( i \in \{1, \ldots, m\} \), are pairwise different; \( a_{k+1}(s_1) = \cdots = a_{k+1}(s_m) \); \( \{a_1, \ldots, a_{s'} \} \cap \{a_1^2, \ldots, a_2^k \} = \emptyset \); \( a_{s'+1} \in \{a_{s'+1}^2, \ldots, a_k^2\} \); \( \omega \) be a GCC r-path leading to \( v_m' \); \( pr_1(\omega) \rightarrow pt_1 \) and \( pr_2(\omega) \rightarrow pt_2 \) be consistent runs having distinct results.

Then there exists \( j, 1 \leq j < m, \) s.t. \( v_j \) belongs to some disproving r-path.

We prove lemma 8 using the same technique as for lemma 7, so due to lack of space we omit the proof.

## 5 Equivalence-Checking Algorithm

The resulting equivalence-checking algorithm is to traverse \( \Gamma \) in any way. During the traversal visited nodes are partitioned into classes defined by a state \( q \), locations \( l_1, l_2 \), and a symbol \( \alpha \) (as in lemmas 7, 8): the nodes \((l_1, l_2, s, \alpha) \) and \((l_1', l_2', s', \alpha') \) belong to the same class iff \( l_1 = l_1', l_2 = l_2', A(s) = A(s') \), and \( \alpha = \alpha' \). If the traversal is finished, then the absence of disproving paths can be checked directly, and the answer is given by theorem 1. If \(|Q|^3 \cdot |\pi_1| \cdot |\pi_2|^2 + 2 \) nodes are visited in any class, then either the non-equivalence is stated (lemma 7), or all other nodes in the class are ignored from now on (lemma 8).

Let \( N \) be the total program size: \( N = |\pi_1| + |\pi_2| \). The number of distinct node classes is \( O(N^2) \). Each class contains \( O(N^3) \) nodes. Thus, no more than \( O(N^3) \) nodes are visited during the traversal. A minimal member \( g \in [h] \) can be constructed in time \( O(|h|) \) using the inductive minimality and lemma 6. Checking if \(|h| \leq |g| \) can be done in time \( O(|h| + |g|^2) \): construct a word \( g' \in [g] \); if \( h = \lambda \), then \(|h| \leq |g| \); otherwise \( h = ah' \), where \( a \in \mathfrak{A} \); if \(|ag| = |g| \) (it can be checked with \( A \)), then delete \( a \) from \( h \) and repeat the procedure; otherwise check whether \( g \) contains a letter \( a \); if not, then \(|h| \not\leq |g| \); if yes, then pick the leftmost occurrence of \( a \) in \( g \); \( g' = g'ag'' \); if each symbol from \( g' \) commutes with \( a \), then delete marked \( a \) from \( h \) and \( g \) and repeat the procedure; otherwise \(|h| \not\leq |g| \). The soundness of the latter algorithm is proved in the appendix. The elimination of a longest prefix can be performed in a similar way,
the difference is that we pick not the leftmost symbol of \( h \) but every symbol which can be shifted to the leftmost position with commutativity relations, and every completion is successful. Each element of \( M \) can be encoded as any its member (e.g. constructed by a computation). During the traversal the algorithm constructs words \( h \) of length at most \( O(N^5) \), and each element comparison can be done in time \( O(N^{10}) \). For each vertex the algorithm does at most \( O(N^3) \) comparisons of elements of \( M \), and this is the bottleneck of the algorithm. Thus, the algorithm performs \( O(N^5) \) steps, and each step is done in time \( O(N^{13}) \), and the total time is \( O(N^{18}) \). The last upper bound together with theorem 1 and lemmas 7, 8 states the validity of the main theorem of this paper.

**Theorem 2.** Let \( M \) be a progressive CA-monoid on a finite set of actions. Then the \( M \)-equivalence problem for propositional programs is decidable in time polynomial in size of programs.

6 Conclusion

We want to present three thoughts concluding the paper. The first one is that the considered case: propositional programs, commutativity, and left absorption — appears to be simple enough to have a polynomial-time decidability for the equivalence problem. The second one is that (nevertheless) this case is complex enough to require a rather nontrivial technique for its analysis. Finally, the third one is that it was mentioned in the introduction that there is still a long way to go, and this statement remains unchanged, as even in a simple example in the introduction we need not only commutativity and absorption (but also dependencies between statements and conditions) to prove the equivalence. To be able to prove the equivalence of “real” fragments of code, we should consider other properties of program primitives, which is the topic of future research.

References

Betweenness, Lukasiewicz Rough Inclusions, Euclidean Representations in Information Systems, Hyper-granules, Conflict Resolution

Lech T. Polkowski

Department of Mathematics and Computer Science
University of Warmia and Mazury
Olsztyn, Poland
and
Polish–Japanese Institute of IT. Warszawa, Poland
polkow@pjwstk.edu.pl

Abstract. The purpose of this note is to augment information systems with a relation of betweenness among things in the universe of the system and derive from it a geometric representation of granules of things in Euclidean spaces. Next, the notion of betweenness renders a service in introducing a new notion of a hyper–granule. An application to conflict resolution by defining coalitions of agents as granules or hyper–granules and their mixed strategies as elements of convex hulls spanned on things defining the granule is proposed. Finally, hyper–granules offer a new classifying algorithm which exploits neighborhoods of things and in a sense is an improved with respect to similarity variant of nearest neighbor classifier.

Key words: rough inclusions, betweenness, Euclidean representations of granules, hyper–granules, coalitions, conflict resolutions, classifier synthesis

1 Introduction: Basic Notions

We would like to recall here basic facts about notions mentioned in Abstract, to make this note self–contained. We begin with the Lukasiewicz rough inclusion.

1.1 The Lukasiewicz Rough Inclusion

Given a set of things $U$, a rough inclusion is a ternary relation $\mu \subseteq U \times U \times [0, 1]$ which renders the notion of ‘to be a part to a degree’. It is rooted in mereology, see, e.g., [3] whose basic notion is that of a part which is a transitive and irreflexive relation $\pi$ on the product $U \times U$ along with its reflexive closure, the ingredient ingr, i.e., $\text{ingr} = \text{part} \cup \text{'} = \text{'}$. The rough inclusion $\mu$ on the set $U \times U$ should satisfy the requirements

1. $\mu(x, y, 1)$ if and only if $\text{ingr}(x, y)$.
2. $\mu(x, y, 1)$ and $\mu(z, x, r)$ imply $\mu(z, y, r)$.
3. $\mu(x, y, r)$ and $s < r$ imply $\mu(x, y, s)$.
A rough inclusion $\mu$ induces a mereological distance $d_\mu$ by means of the formula

$$d_{\mu}(x,y) = \min\{\sup\{r : \mu(x,y,r)\}, \sup\{s : \mu(y,x,s)\}\}.$$  

(1)

Assuming that suprema in the formula (1) are achieved, we have

**Proposition 1.** $d_\mu(x,y) = 1$ if and only if $x = y$.

In the case when things in the universe $U$ are described by means of a finite set $A$ of attributes so each thing $x \in U$ is represented by its information set $\{a(x) : a \in A\}$, we can define the Łukasiewicz rough inclusion $\mu_L$ by taking as its value on a pair $x,y$ in $U$ the quotient of the cardinality of the indiscernibility set $IND(x,y) = \{a \in A : a(x) = a(y)\}$ and the cardinality of the set $A$:

$$\mu_L(x,y) = \frac{\text{card}(IND(x,y))}{\text{card}(A)},$$  

(2)

cf., [3], Ch.6.

### 1.2 Betweenness

The notion of betweenness plays an essential role in axiomatization of elementary geometry of Euclidean spaces due to Tarski, see Tarski and Givant [4], it is formalized as a relation $B(x,y,z)$ (‘$y$ is between $x$ and $z$’); intuitively, $B(x,y,z)$ means that $y$ lies on the straight line segment with endpoints $x,z$.

Van Benthem [1] proposed an extension of the betweenness relation based on the relation of nearness $N(x,y,z)$ (‘$x$ is closer to $y$ than $z$’) which in terms of the distance $d_\mu$ would be defined by means of

$$N(x,y,z) \text{ if and only if } d_\mu(x,y) > d_\mu(z,y).$$  

(3)

The relation $N$ thus defined, does satisfy all axioms for nearness in Van Benthem [1], i.e.,

1. N1. $N(z,u,v)$ and $N(v,u,w)$ imply $N(z,u,w)$ (transitivity).
2. N2. $N(z,u,v)$ and $N(u,v,z)$ imply $N(u,z,v)$ (triangle inequality).
3. N3. $N(z,u,z)$ is not true for each pair $u,z$ (irreflexivity).
4. N4. $z = u$ or $N(z,z,u)$ (selfishness).
5. N5. $N(z,u,v)$ implies $N(z,u,w)$ or $N(w,u,v)$ (connectedness).

For the proof, cf., [3] Ch. 6.

Betweenness relation in the sense of Van Benthem $T_B(z,u,v)$ (‘$u$ is between $z$ and $v$’) introduced in Van Benthem [1] is rendered by a formula

$$T_B(z,u,v) \Leftrightarrow \text{[for each } w \in U \text{ (} u = w \text{) or } N(u,z,w) \text{ or } N(u,v,w)]$$  

(4)

This means that for each thing $w$ distinct from $z$, either $u$ is closer to $z$ than is $w$ or $u$ is closer to $v$ than is $w$. 
2 Granules from Indiscernible Things

We assume that things in the set $U$ are described by means of values of $N$ real valued attributes in the set $A$; from this point of view the set $U$ becomes the set $R(U) \subseteq R^N$ of vectors in $R^N$. As said above, each thing $x \in U$ is represented by the vector $\langle a_i(x) \rangle_{i=1}^N$ where $< a_1, a_2, \ldots, a_N >$ is a fixed ordering of attributes.

We consider a pair $a, b$ of things in $U$ such that $a, b$ have in common attribute values in a set $\Delta \subset A$, where $\Delta$ is non–empty; formally, this means that $IND(a, b) = \Delta$. Let $\text{card}(\Delta) = \delta \cdot N$. (5)

Example 1. We consider things $a, b$ and a thing $c$ which agrees with $a$ and $b$ on the set $\Delta$ and has $\frac{1}{2} \cdot (N - \delta \cdot N)$ attribute values in common with $a$ and $\frac{1}{2} \cdot (N - \delta \cdot N)$ attribute values in common with $b$; we assume for simplicity that it is possible otherwise we should consider $\lfloor \frac{1}{2} \cdot (N - \delta \cdot N) \rfloor$ and $\lceil \frac{1}{2} \cdot (N - \delta \cdot N) \rceil$, respectively which would only make calculations more cumbersome. Please observe that we do not specify positions of values, i.e., we do not specify particular attributes on which these values are taken, with exception for $\Delta$ only. Therefore, it makes sense to identify all such things into a class $[c]$ in which all things have values on attributes in $\Delta$ same as $a$ and $b$ and share with each of $a, b$ one half of the remaining values.

Our Thesis is

Proposition 2. $c$ is between $a$ and $b$.

Proof. We find the distance $d_{\mu_L}$ between $a, c$ and $b, c$ with respect to the rough inclusion $\mu_L$; by definition (1),

$$d_{\mu_L}(a, c) = \frac{\delta \cdot N + \frac{1}{2} \cdot (N - \delta \cdot N)}{N} = \frac{1 + \delta}{2} = d_{\mu_L}(c, b).$$

(6)

We now consider an arbitrary thing $x$ which for some quotient $\alpha$ in $[0, 1]$ has $\alpha \cdot \delta \cdot N$ values of attributes in $\Delta$ in common with $a$ and for some quotient $\beta \in [0, 1]$ has $\beta \cdot (N - \delta \cdot N)$ values of attributes not in $\Delta$ in common with $a$ and at most $(1 - \beta) \cdot (N - \delta \cdot N)$ values in common with $b$. We have

$$d_{\mu_L}(x, a) = \frac{\alpha \cdot \delta \cdot N + \beta \cdot (N - \delta \cdot N)}{N} = \beta + (\alpha - \beta) \cdot \delta,$$

(7)

and

$$d_{\mu_L}(x, b) \leq \frac{\alpha \cdot \delta \cdot N + (1 - \beta) \cdot (N - \delta \cdot N)}{N} = 1 - \beta + (\alpha + \beta - 1) \cdot \delta.$$ (8)

Let us assume, to the contrary, that (1) $d_{\mu_L}(x, a) > d_{\mu_L}(c, a)$ and (2) $d_{\mu_L}(x, b) > d_{\mu_L}(c, b)$.

Condition (1) means after substitution of values in (6) and (7) that

$$\beta + (\alpha - \beta) \cdot \delta > \frac{1}{2} + \frac{\delta}{2},$$ (9)
\[ \alpha \cdot \delta + \beta \cdot (1 - \delta) > \frac{1}{2} \cdot \frac{\delta}{2}. \]  

(10)

Similarly, condition (2) yields after values in (6) and (8) are substituted into it,
\[ \alpha \cdot \delta - \beta \cdot (1 - \delta) > \frac{3}{2} \cdot \delta - \frac{1}{2}, \]  

(11)

Adding inequalities (10) and (11) yields
\[ 2 \cdot \alpha \cdot \delta > 2 \cdot \delta, \]  

(12)

and, as \( \delta > 0 \),
\[ \alpha > 1 \]  

(13)

which is impossible. This proves our proposition.

This example has served as a motivation for further generalizations.

We have already observed that we have not specified the attributes selected and actually we have discussed classes of equivalence of things, two things \( x, y \) being equivalent if and only if they have had same fractions of attribute values for attributes in \( \Delta \) and same fraction of attribute values not in \( \Delta \). Hence, for fractions \( \gamma \) of values in \( \Delta \) and \( \varepsilon \) of values in \( A \setminus \Delta \) common with \( a \), and, at most \( 1 - \varepsilon \) values of attributes in \( A \setminus \Delta \) in common with \( b \), we denote with the symbol \([\gamma, \varepsilon]\) the class of things satisfying those conditions. We regard, hence, the vector \([\gamma, \varepsilon]\) as the representation of that class in the vector space \( R^2 \), the representation space. In particular, the thing \( a \) is represented as \([1, 1]\), the thing \( b \) is represented as \([1, 0]\), and, the thing \( c \) is in the class \([1, \frac{1}{2}]\). Thus, in the Euclidean plane, \( c \) is the midpoint in the segment with endpoints \( b \) and \( c \), i.e. \( c \) is between \( a \) and \( b \) in the elementary geometry sense.

The proof above does suggest a more general result.

**Proposition 3.** For \( \alpha \in [0, 1] \), the class \([1, \alpha]\) is between classes \([1, 0]\) of \( b \) and \([1, 1]\) of \( a \) in the representation space.

**Proof.** This proof goes on similar lines as proof of the previous proposition. Let \( d \) denotes the class \([1, \alpha]\) and let \( x \) be in the class \([\gamma, \varepsilon]\). Assuming to the contrary that
\( (1) d_{\mu_L}(x, a) > d_{\mu_L}(d, a) \) and \( (2) d_{\mu_L}(x, b) > d_{\mu_L}(d, b) \), we obtain inequalities
\[ \gamma \cdot \delta + \varepsilon \cdot (1 - \delta) > \delta + \alpha \cdot (1 - \delta) \]  

(14)

and,
\[ \gamma \cdot \delta + (1 - \varepsilon) \cdot (1 - \delta) > \delta + (1 - \alpha) \cdot (1 - \delta) \]  

(15)

Sidewise addition of (14) and (15) yields the inequality
\[ 2 \cdot \gamma \cdot \delta > 2 \cdot \delta \]  

(16)

i.e., \( \gamma > 1 \), impossible. The proposition is proved.

It turns out that the whole interval from \([1, 0]\) to \([1, 1]\) consists of classes between \([1, 0]\) and \([1, 1]\). In the representation space of classes \([\gamma, \varepsilon]\), betweenness in the mereological sense coincides with betweenness in the geometric sense of the Euclidean geometry of the plane.
2.1 The Case $\Delta = \emptyset$

In this case $\delta = 0$ and our proofs above are not valid. In this case, given things $a, b$ in $U$, with $\text{IND}(a, b) = \emptyset$, for a choice of $\gamma \in [0, 1]$, we form things which have $\gamma \cdot N$ attribute values in common with $a$ and $(1 - \gamma) \cdot N$ attribute values in common with $b$. We represent this class of things as before with the vector $[\gamma, 1 - \gamma]$ in the Euclidean plane. In this representation, $a$ is represented as $[1, 0]$ and $b$ is represented as $[0, 1]$, so $[\gamma, 1 - \gamma]$ is a convex combination of $[1, 0]$ and $[0, 1]$.

**Proposition 4.** For each choice of $\gamma \in [0, 1]$, the class of things represented as $[\gamma, 1 - \gamma]$ is between $a$ and $b$ in the sense of betweenness relation $B$.

**Proof.** For a point $[\varepsilon, \delta]$ with $\varepsilon, \delta \in [0, 1]$ and $\varepsilon + \delta \leq 1$, if, e.g., $\varepsilon > \gamma$ then $\delta < 1 - \gamma$.

It follows that things and their classes which are between $a$ and $b$ are located in the representation space in the segment with endpoints $[1, 0]$ for $a$ and $[0, 1]$ for $b$, i.e., between these endpoints in the geometrical sense.

This suggests a generalization. We define a more general betweenness relation

$$GB(x, a_1, a_2, \ldots, a_n)$$

(*$x$ is between $a_1, a_2, \ldots, a_n*) if and only if for each thing $y \neq x$ the thing $x$ is closer than $y$ to some $a_i$ in the mereological sense of (3).

2.2 The General Case

We consider a set $V = \{a_1, a_2, \ldots, a_n\}$ of things in $U$. For a choice of $\gamma_1, \gamma_2, \ldots, \gamma_n \in [0, 1]$ with $\sum_i \gamma_i = 1$, which we summarily denote by the vector $\vec{\gamma}$, we denote as $(V, \vec{\gamma})$ the class of things which have the fraction $\gamma_i$ of attribute values in common with the thing $a_i$. As above, the fact is true that

**Proposition 5.** The class $(V, \vec{\gamma})$ of things represented by the vector $\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ satisfies the relation $GB((V, \vec{\gamma}), a_1, a_2, \ldots, a_n)$.

Proof of this proposition goes on lines of the preceding proof.

**Proposition 6.** The relation $GB(x, a_1, a_2, \ldots, a_n)$ holds for a class $x = (\{a_1, \ldots, a_n\}, [\gamma_1, \gamma_2, \ldots, \gamma_n])$ if and only if $[\gamma_1, \gamma_2, \ldots, \gamma_n]$ belongs in the convex hull of vectors $[1, 0, 0, \ldots, 0]$, $[0, 1, 0, 0, \ldots, 0], \ldots, [0, 0, \ldots, 0, 1]$ representing, in this order, classes $a_1, a_2, \ldots, a_n$.

3 A Geometric Representation of Granulation

In the general case, the process of forming of a class $x = (V, \vec{\gamma})$ represented by the vector $\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ can be regarded as forming of a granule of things $gr(V, \vec{\gamma})$. This granulation process is different of previously considered in that it does involve a sort of
a shuffling map, which takes into \( x \) things having specified fractions of attribute values in common with the corresponding classes \( a_i \) but over arbitrary sets of attributes of cardinality \( \gamma_i \cdot N \). This secures a kind of control over values in contradiction to our previous granulation paradigm in which only a fixed part of a given thing attribute values was coming from the granule center, cf., [3], Chs. 5–7.

For a given vector \( \tau = [\gamma_1, \gamma_2, \ldots, \gamma_n] \) in the representation space, representing the granule \( gr(V, \tau) \), the size of the granule \( gr(V, \tau) \) is

\[
\prod_{j \leq n} \left( N - \sum_{k=1}^{j} \gamma_k \cdot N \right). \tag{18}
\]

The number of granules of type represented by a vector \( \tau = [\gamma_1, \gamma_2, \ldots, \gamma_n] \) is the number of sequences \( k_1, k_2, \ldots, k_n \) of natural numbers such that \( \sum_i k_i = N \), i.e., it equals the number of integer–valued vectors on the hyperplane \( \sum_i x_i = N \), given recurrently by the function \( \phi(N, n) \):

\[
\phi(N, 1) = 1, \tag{19}
\]

\[
\phi(N, k + 1) = \sum_{j=0}^{N} \phi(N - j, k). \tag{20}
\]

It follows that \( \phi(N, n) \) is of order \( O(N^{n-1}) \).

Let us observe that those estimates concern all objects generated by the process described above; in reality, only a small fraction of those objects will exist in the set \( U \). Hence, we define a \( U \)–granule \( g(U, \tau) \) as \( g(\tau) \cap U \).

### 3.1 Granular Classifiers

Assume that a decision partition is imposed on the things in the set \( U \) into decision classes \( D_1, D_2, \ldots, D_m \). For a \( U \)–granule \( g = g(U, \tau) \), we denote with \( Pr[g, D_i] \) the probability that a randomly chosen thing in \( g \) is assigned to the class \( D_i \). Then the Bayesian decision on \( g \) is the class \( D(g) = D_i^* \) such that \( Pr[g, D_i^*] = \max \{ Pr[g, D_i] : i = 1, 2, \ldots, m \} \).

The mapping \( g \rightarrow D(g) \) from \( U \)–granules into their Bayesian decisions is the Bayesian granular classifier. It is deterministic as the covering into granules is finer than the partition into decision classes.

### 4 An Application Proposal: Conflict Resolution

We propose to apply the afore described results to the problem of conflict resolution. In a conflict, we have a finite number of agents declaring their standpoints on a number of issues; those standpoints are conflicting, i.e., on each issue there are agents having distinct standpoints. A resolution of a conflict means some process leading to
a rationally chosen set of non–conflicting standpoints for all agents. Examples can be
saddle points in two–person zero–sum games, either for pure or mixed strategies, or,
the Nash equilibrium points in continuous convex–concave games. In each of these
cases, agents reach a set of strategies which is satisfying for each of them.

Such a rationale is difficult to be obtained in less formalized conflicts, and, our
element deals with such a conflict. We base our approach on the example of a
conflict in Pawlak [2], for which that author gave an analysis in terms of the rough
set–theoretical approach.

Example 2. The conflict described in [2] does involve six agents: A1, A2, ..., A6 and
five issues I1, I2, ..., I5 with the standpoints 0 (disagreement), 1 (agreement) and N (the
neutral standpoint equivalent to ‘do not care’). Hence, when standpoints N and 0 or 1
are confronted, N means 0 or 1, respectively.

This case is visualized in Table Fig.1 below.

<table>
<thead>
<tr>
<th>agent</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A2</td>
<td>1</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>A4</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>A5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A6</td>
<td>N</td>
<td>1</td>
<td>0</td>
<td>N</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 1. The setting of a conflict

In this case, the set of agents $V = \{A_{i_1}, A_{i_2}, \ldots, A_{i_k}\}$ forming a granule is called a
coalition, and standpoints of them on issues are their pure strategies. Given coefficients
$\gamma_1, \ldots, \gamma_k$ summing up to 1, sequences of issues in the granule $(V, [\gamma_1, \ldots, \gamma_k])$ are mixed
strategies of agents. Given a partition of the set of agents into coalitions $C_1, C_2, \ldots, C_m$,
we call a conflict resolution a set $i_1, i_2, \ldots, i_m$ of mixed strategies issued from granules
generated by $C_1, C_2, \ldots, C_m$ which are consistent, i.e., conflict–less.

We use the Lukasiewicz rough inclusion as the measure of similarity of agents, and,
we include in one coalition agents for which pairwise similarity measures are greater
than their similarity degrees to other agents.

From Fig. 1, it follows that agents A1 and A6 are similar to degree of 0.8, agents
A2, A3, A4, A5 are similar to each other to degree of 1.0, and, agent A1 is similar to
A2, A3, A4, A5 to degree not greater than 0.4. Only A6 is similar to A2, A3, A4 to
degree 0.8, and, to A5 to degree 0.6.

We decide to consider coalitions $V_1 = \{A1, A6\}$ and $V_2 = \{A2, A3, A4, A5\}$.
Forming all possible granules over $V_1$ and $V_2$ yields possible mixed strategies of both
calitions. Due to their number, we list in Figs.2,3 a selection of six of those strategies
for each of the two coalitions.
We find exemplary boldfaced mixed strategies \textbf{N1ON1} for the coalition \textit{V1} and \textbf{NN0NN} for the coalition \textit{V2} which are identical for both coalitions: each of them can provide a conflict resolution; it is true that this requires a ‘gentleman’s agreement’ to accept the results of this procedure; for instance, each of these solutions would require that A1 gives up on its standpoint on issue $I_1$, which in real practice can be impossible due, e.g., to political and religious reasons. The specific bargained for standpoints can be: 01001, 11001, 01011, 11011.

[Figure 2. Selected mixed strategies for the coalition V1]

[Figure 3. Selected mixed strategies for the coalition V2]

5 \textbf{The Thing Point of View: Hyper–Granules}

The approach presented above begins with a group \textit{G} of things and ends with the notion of collection of things between \textit{G}. Among those things are virtual ones not present in the decision/information system. Therefore, we now propose the analysis from the viewpoint of things in the information system.

We consider a maximal set of things \textit{X} with the property

\[ \forall x \in X. \exists Y \subseteq X \setminus \{x\}. Btw(x, Y). \]

(21)

We call \textit{X} a \textit{hyper–granule}.

Lemma 1. For each $x \in X$, it is true that $Btw(x, X \setminus \{x\})$.

Lemma 2. For each $y \notin X$, it is true that $Btw(y, Y)$ holds for no $Y \subseteq X$. Moreover, there exists an attribute \(a\) such that the value $a(y) \neq a(x)$ for each $x \in X$. Contrarily, for each $x \in X$, and, for each attribute $a$, it is true that $a(x) \in \{a(y) : y \in X \setminus \{x\}\}$.

The hyper–granule \textit{X} is \textit{attribute–self–contained} in the sense of Lemma 2.

For the hyper–granule \textit{X}, we consider the complementary set of objects $U \setminus X$;
let $X'$ be a hyper–granule of objects in $U \setminus X$. Iterating this procedure, we define in the universe set $U$ a set of hyper–granules $\{X^{(i)} : i \in I\}$, where $X^{(j+1)}$ is a hyper–granule in the set $U \setminus \bigcup_{i \leq j} X^{(i)}$. The remnant of $U$ consist of outliers unable to enter any hyper–granule.

We denote with the symbol $H(x)$ the hyper–granule containing the thing $x$. For a thing $x$, we consider sets of thing in $H(x) \setminus \{x\}$ which we denote with the generic symbol $N(x)$ with the property that $Btw(x, N(x))$ and all coordinates of the vector representing $x$ with respect to $N(x)$ are positive; we will call any such set a neighborhood of $x$; in the example in Fig.4, for instance, $N(4) = \{8, 10\}$.

A neighborhood $N(x)$ of $x$ is irreducible if it is of minimal possible cardinality; such is $N(4)$ pointed to above.

6 Applications: Hyper–Granules as Coalitions in Conflicts and a Decision Prediction Modal Logic

We will discus first the problem of conflict resolution.

6.1 Hyper–Granules as Coalitions in Conflicts

We return to Fig. 1, showing standpoints of agents A1–A6 on issues I1–I5. As before, we regard the standpoint $N$ as 'don’t care' a fortiori $N$ can be 0 or 1. We observe that A1 cannot be considered as a candidate to the hyper–granule of A2–A6 as the issue I1 on A1 takes value 0 not taken by any of A2–A6.

On the other hand, due to our convention about $N$, agents A2–A6 make a hyper–granule $X$. This hyper–granule can produce a between element $NN0NN$ representing sixteen specific bargaining propositions, from 00000 to 11011. The agent A1 remains as an outlier due to value 0 on I1 which cannot be supplied by any other agent.

The bargaining between $X$ and A1 can focus on I3 on which A1 takes value of 1 and all other agents adopt the standpoint 0.

6.2 A Decision Prediction Logic and a Decision Assignment Algorithm

We will regard the universe $U$ of the informatron system as the training set on which the decision is given, and we consider the additional test set on which decision is to be learned on the basis of its values on $U$.

Given a test thing $x$ about which we assume that there exist neighborhoods of it in the set $U$, and for a formula $\phi$, we declare that

$$ x \models \phi \iff y \models \phi \text{ for each irreducible } N(x) \text{ and each } y \in N(x) . \quad (22) $$

Modal operators $L$ of necessity and $M$ of possibility are introduced as follows.

$$ x \models L\phi \iff y \models \phi \text{ for each } N(x) \text{ and each } y \in N(x) . \quad (23) $$
\[ x \models M\phi \iff y \models \phi \text{ for some } N(x) \text{ and some } y \in N(x). \]  
(24)

It follows that
\[ L = \neg M\neg. \]  
(25)

\[ x \models L(\phi \Rightarrow \psi) \Rightarrow (x \models L\phi \Rightarrow x \models L\psi). \]  
(26)

\[ x \models L\phi \Rightarrow x \models \phi. \]  
(27)

Our decision relation formula \( \phi \) is \( \nu_d \in A \) where \( A \subseteq V_d \), i.e., \( A \) is a subset of the set of decision values; the formula \( x \models (\nu_d \in A) \) reads that decision value proposed for \( x \) is in the set \( A \) of decision values. Necessitation means stressing this hypothesis by conforming it on all neighborhoods of \( x \), and, possibility indicates possible sets of values of decision for \( x \).

### 7 Appendix: Computational Aspects of Hyper–Granules

We consider an information system \( S = (U, A, V) \) where \( U \) is a set of things, \( A \) is a set of attributes, and, \( V \) is a set of attribute values. We will need the notion of a dual information matrix \( S^* \) defined as the triple \((A, V, U)\), where for each pair \((a, v)\) the entry in the cell \( S^*(a, v) \) is

\[ \{x \in U : a(x) = v\}. \]  
(28)

**Computing hyper–granules**

The Algorithm proceeds as follows.

**HYPER–GRANULE \((U, A, V)\)**

1. Form the dual information matrix \( S^* \);
2. For each \( x \) in \( U \) do
   3. if there is a cell \((a, v)\) such that \( S^*(a, v) = \{x\} \)
   4. then remove \( x \) from all cells.
5. Repeat steps 2-4 until
   7. all cells are either empty or each contains at least two things;
6. Return \( X \) = the set of all things that occur in at least one non–empty cell.

Repeating the algorithm with the dual information matrix \((A, V, U \setminus X)\) we may eventually obtain further hyper–granules.

The complexity of the algorithm is \( \Theta(|A| \cdot |V| \cdot |U|) \) as we do take into account neither the cost of inserting a symbol into a cell nor deleting it from a cell.

We consider an information system TEST in Fig. 4. The dual information matrix TEST\(^*\) is shown in Fig. 5. We have to remove from all cells the things 2, 5, 9. As each remaining cell is either empty or some at least two–element set, the hyper–granule is \( X = \{1, 3, 4, 6, 7, 8, 10\} \). For the remaining subsystem TEST1=(\(A, V, \{2, 5, 9\}\)), we
obtain the dual information matrix TEST1∗ shown in Fig. 6. After steps 2–4 of the algorithm, all cells are empty so there is no second hyper–granule: things 2, 5, 9 are outliers.

We would like to observe in addition that it is easy to read off from the dual information matrix the coordinates of a thing in the representation space; e.g., the thing 4 can be represented as the vector \([0, 0, 0, 0, 0, 0, 1, 0, 0, 0]_10\) in the simplex spanned on unit vectors representing, respectively, things 1, 2, ..., 10 in the vector space \(R^{10}\). Hence, \(N(4) = \{8, 10\}\) is an irreducible neighborhood of the thing 4; another irreducible neighborhood of 4 is \(\{1, 10\}\) with coordinates \([\frac{1}{2}, 0, ..., 0, \frac{1}{2}]\).

### 7.1 A Decision Assigning Algorithm

For a new test thing \(y\), we consider in the universe \(U\) (the training set) the hyper–granule \(H(y) \subseteq U \cup \{y\}\) along with irreducible neighborhoods \(N_1(y), \ldots, N_k(y)\) (if there are any). Let \(D(y) \subseteq V_d\) be the least set of decision values with the property that

\[
\text{If } x \in \bigcup_j N_j(y) \text{ then } d(x) \in D(y). \tag{29}
\]

By (22), the hypothetical \(d(y)\) belongs in \(D(y)\). Now, given \(j \leq k\), the neighborhood \(N_j(y)\) votes for decision value \(d_j(y)\) in the manner as follows. For \(z \in N_j(y)\), we denote with \(q_j(z)\) the coordinate with which \(z\) enters the vector representing \(y\) with respect to \(N(y)\). Then

\[
d_j(y) = \arg\min_{v \in D(y)} ||v - \sum_z q_j(z) \cdot d(z)||, \tag{30}
\]

where \(||-||\) is a metric chosen for Euclidean space containing \(V_d\). Given a parameter \(p \leq k\), we select \(p\) neighborhoods from among \(N_1(y), N_2(y), \ldots, N_k(y)\) with greatest values of \(q_j = \max_z q_j(z)\) and let

\[
d(y) = \arg\min_{v \in D(y)} ||v - \sum_{j \leq p} q_j \cdot d_j(y)||. \tag{31}
\]

This approach may be regarded as a two–step variant of nearest neighbor classifier: neighborhoods which are irreducible are at the same time closest to the thing; an important factor not present in usual nearest neighbor classifiers is that neighborhoods guarantee also that attribute values in the thing come from neighborhood members which double stresses the similarity among the thing and neighborhood members.

To give a procedure for computing neighbors of things, we define some useful notions,

\[
I(x, y) = \{a \in A : a(x) = a(y)\}, \tag{32}
\]

and,

\[
f(x, y) = \frac{\text{card}(I(x, y))}{\text{card}(A)}. \tag{33}
\]
<table>
<thead>
<tr>
<th>thing</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Fig. 4. Information system TEST**

<table>
<thead>
<tr>
<th>value</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{2, 4, 5, 7, 10}</td>
<td>{1, 3, 4, 5, 8, 10}</td>
<td>{1, 3, 6, 7, 9}</td>
<td>{2, 6, 8, 9}</td>
<td>{3, 5, 7, 10}</td>
<td>{1, 4, 5, 8, 10}</td>
</tr>
<tr>
<td>1</td>
<td>{1, 3, 6, 8, 9}</td>
<td>{2, 6, 7}</td>
<td>{2, 4, 8, 10}</td>
<td>{1, 3, 4, 5, 7, 10}</td>
<td>{1, 2, 4, 6, 8}</td>
<td>{3, 6, 7, 9}</td>
</tr>
<tr>
<td>2</td>
<td>∅</td>
<td>{9}</td>
<td>∅</td>
<td>{9}</td>
<td>{9}</td>
<td>{2}</td>
</tr>
</tbody>
</table>

**Fig. 5. The dual information matrix TEST**

<table>
<thead>
<tr>
<th>value</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{2, 5}</td>
<td>{5}</td>
<td>{9}</td>
<td>{2, 9}</td>
<td>{5}</td>
<td>{5}</td>
</tr>
<tr>
<td>1</td>
<td>{9}</td>
<td>{2}</td>
<td>{2}</td>
<td>{5}</td>
<td>{2}</td>
<td>{9}</td>
</tr>
<tr>
<td>2</td>
<td>∅</td>
<td>{9}</td>
<td>{5}</td>
<td>∅</td>
<td>{9}</td>
<td>{2}</td>
</tr>
</tbody>
</table>

**Fig. 6. The dual information matrix TEST1**

For $X \subseteq A$ and $x \in U$, we let $V(x, X) =$ the sequence $a(x) : a \in X$ in the assigned order of attributes; the symbol $0^n$ denotes the sequence of length $n$ of 0.

**Procedure Irreducible Neighborhood(x)**

Input: the thing x

Output List of irreducible neighborhoods of x

variable sets $\pi$, $A_\pi(x, y)$, $n_\pi(x, y)$, $N(x, y)$, $v_\pi(x, y)$

Initialization: $N(x, y) \leftarrow 0$,

1. order coefficients $f(x, y)$ in descending order

2. in descending order of $f(x, y)$ do
Now, isducible neighborhoods of 10.

As an example let us assign decision values to things 1, 2, \ldots, 10 in TEST, in Fig. 7.1. For simplicity, the thing 4 is regarded as a test thing. The set \(D(4) = \{0, 1\}\). Irreducible neighborhoods of 4 are of cardinality 2 and they are: \(N_1(4) = \{1, 10\}\) with coordinates \(\left[\frac{1}{2}, \frac{9}{6}\right]\); \(N_2(4) = \{1, 10\}\) with coordinates \(\left[\frac{2}{2}, \frac{7}{6}\right]\); \(N_3(4) = \{7, 8\}\) with coordinates \(\left[\frac{2}{6}, \frac{5}{6}\right]\); \(N_4(4) = \{8, 10\}\) with coordinates \(\left[\frac{1}{6}, \frac{5}{6}\right]\).

Now, \(N_1(4)\) votes for \(\frac{1}{4} \cdot 1 + \frac{2}{3} \cdot 1 = 1 = d_1(4)\); \(N_2(4)\) votes for \(\frac{2}{6} \cdot 0 + \frac{4}{6} \cdot 1 = 0 = d_2(4)\); \(N_3(4)\) votes for \(\frac{2}{6} \cdot 1 + \frac{1}{6} \cdot 1 = 1 = d_3(4)\); \(N_4(4)\) votes for \(\frac{1}{4} \cdot 1 + \frac{2}{3} \cdot 1 = 1 = d_4(4)\).

The final decision value is the nearest of 0, 1 to the value \(\frac{\frac{1}{4} + \frac{2}{3} + \frac{1}{4} + \frac{1}{4} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6}}{\frac{2}{6} + \frac{3}{6} + \frac{2}{6} + \frac{2}{6}} = \frac{13}{17}\) which is \(d(4) = 1\).
<table>
<thead>
<tr>
<th>value</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${4, 7, 10}$</td>
<td>${1, 3, 4, 8, 10}$</td>
<td>${1, 3, 6, 7}$</td>
<td>${6, 8}$</td>
<td>${3, 7, 10}$</td>
<td>${1, 4, 8, 10}$</td>
</tr>
<tr>
<td>1</td>
<td>${1, 3, 6, 8}$</td>
<td>${6, 7}$</td>
<td>${4, 8, 10}$</td>
<td>${1, 3, 4, 7, 10}$</td>
<td>${1, 4, 6, 8}$</td>
<td>${3, 6, 7}$</td>
</tr>
</tbody>
</table>

**Fig. 7.** The dual information matrix $TEST^*$ after removing outliers

```
1 2 3 4 5 6 7 8 9 10
0 2 1 new thing 1 2 1 0 1
```

**Fig. 8.** Decision values for TEST; thing 4 regarded as a test thing

**References**

Fuzzy Systems of Logical Inference and Their Applications

Oleksandr Provotar

University of Rzeszow, Rzeszow, 35-310, Poland
aprowatal@bigmir.net

Abstract. The approaches to the solution of various problems of artificial intelligence methods are proposed. In particular, the problem of knowledge representation by means of fuzzy specifications in expert systems, the problem of recognizing the structures of the proteins of different organization levels and the problem of building linguistic models in fuzzy Boolean variables logic are considered. All methods are based on the ideas of inductive mathematics. To investigate a reliability of these methods is possible only with the help of the theory of probability or possibility theory.

Key words: fuzzy sets, problem of recognizing, fuzzy Boolean variable

1 Introduction

It is known that the fuzzy sets [2], [5], [6], [7], [8] are the convenient tool to present knowledge in information systems. Using the fuzzy sets it is possible to outline, for instance, the picture of symptoms of the patient in expert diagnostics systems. Determination of the diagnosis in such systems requires using the mechanisms of logical inference. In particular, in case of the fuzzy specifications as symptoms as diagnostics such mechanisms can be so called the fuzzy systems of logical inference which are built on the basis of ideas and methods of inductive mathematics [1].

The fuzzy specification of problem means ordered set of fuzzy instructions (fuzzy rules, linguistic rules). The fuzzy specification of the problem with the algorithm during fulfilling which the approximate (fuzzy) solution of the problem is received will be called as fuzzy system of logical inference.

Let \( x_1, \ldots, x_n \) are input linguistic variables and \( y \) – output linguistic variable [6]. The ordered set of fuzzy instructions looks like as following

\[
\text{if } x_1 \text{ is } A_{11} \land \cdots \land x_n \in A_{1n} \text{ then } y \text{ is } B_1,
\]
\[
\text{if } x_1 \text{ is } A_{21} \land \cdots \land x_n \in A_{2n} \text{ then } y \text{ is } B_2,
\]

\[
\ldots \ldots \ldots \ldots \ldots \ldots \ldots
\]
\[
\text{if } x_1 \text{ is } A_{m1} \land \cdots \land x_n \in A_{mn} \text{ then } y \text{ is } B_m,
\]

where \( A_{ij} \) and \( B_i \) – fuzzy sets, symbol \( \land \) is interpreted as \( t \)-norm of fuzzy sets.

The algorithm of calculating the output of such specification under the inputs \( A_1^\prime, \ldots, A_n^\prime \) consists in performing such steps:

1. Calculate the \( t \)-norm of the premises for each rule:

\[
A_{ij}^\prime \land \cdots \land A_{ij}^\prime \in A_{ij}.
\]

2. If the \( t \)-norm of all premises of the rule is non-zero, then the conclusion of the rule is:

\[
y = B_i.
\]

3. If the \( t \)-norm of at least one premise of the rule is zero, then the conclusion of the rule is:

\[
y = \text{null}.
\]
1. Calculate the truth level of the rules:

\[ \alpha_i = \min \left[ \max \left( A_1'(x_1) \land A_{i1}(x_1) \right), \ldots, \max \left( A_n'(x_n) \land A_{in}(x_n) \right) \right] ; \]

2. Calculate outputs of each rule:

\[ B_i'(y) = \min(\alpha_i, B_i(y)) ; \]

3. Calculate aggregated output:

\[ B(y) = \max \left( B_1'(y), \ldots, B_m'(y) \right) . \]

2 Expert Diagnostics Systems

Let \( X_1 = \{5, 10, 15, 20\}, X_2 = \{5, 10, 15, 20\}, X_3 = \{35, 36, 38, 39, 40\} \) – spaces for determining the values of linguistic variables:

\[ x_1 = \text{"Coughing"} = \{\text{"weak"}, \text{"moderate"}, \text{"strong"}\}, \]
\[ x_2 = \text{"Runningnose"} = \{\text{"weak"}, \text{"moderate"}, \text{"strong"}\}, \]
\[ x_3 = \text{"Temperature"} = \{\text{"normal"}, \text{"raised"}, \text{"high"}, \text{"veryhigh"}\}, \]

accordingly.

Determine the elements of these sets:

\[
\text{"Coughing"} : \text{"weak"} = 1/5 + 0.5/10;  
\text{"moderate"} = 0.5/5 + 0.7/10 + 1/15; 
\text{"strong"} = 0.5/10 + 0.7/15 + 1/20. 
\]

\[
\text{"Runningnose"} : \text{"weak"} = 1/5 + 0.5/10;  
\text{"moderate"} = 0.5/10 + 1/15; 
\text{"strong"} = 0.7/15 + 1/20. 
\]

\[
\text{"Temperature"} : \text{"normal"} = 0.5/35 + 0.8/36 + 0.9/37 + 0.5/38;  
\text{"raised"} = 0.5/37 + 1/38; 
\text{"high"} = 0.5/38 + 1/39; 
\text{"veryhigh"} = 0.8/39 + 1/40. 
\]

Let \( Y = \{\text{influenza, sharp respiratory disease, angina, pneumonia}\} \) – space for determining the value of linguistic variable \( y \). Then the dependence of the patient’s disease on his symptoms can be described by the following systems of speci-
fication:

if \( x_1 \) is “weak” \( \land \) \( x_2 \) is “weak” \( \land \) \( x_3 \) is “raised” then
\( y \) is “0.5/influenza + 0.5/sharp respiratory disease + 0.4/angina + 0.8/pneumonia”;

if \( x_1 \) is “weak” \( \land \) \( x_2 \) is “moderate” \( \land \) \( x_3 \) is “high” then
\( y \) is “0.8/influenza + 0.7/sharp respiratory disease + 0.8/angina + 0.3/pneumonia”;

if \( x_1 \) is “weak” \( \land \) \( x_2 \) is “moderate” \( \land \) \( x_3 \) is “very high” then
\( y \) is “0.9/influenza + 0.7/sharp respiratory disease + 0.8/angina + 0.2/pneumonia”;

If to the input \( x_1 \) of this algorithm to supply value \( A'_1 = 1/5 + 0.7/10 \), to the input \( x_2 \) – value \( A'_2 = 1/5 + 0.5/10 \), to the input \( x_3 \) – value \( A'_3 = 1/36 + 0.9/37 \), then in accordance with procedure of fulfilling the algorithm of the fuzzy system of logical inference the fuzzy solution of the problem will be received
\( B = 0.5/influenza + 0.5/sharp respiratory disease + 0.4/angina + 0.5/pneumonia. \)

The problem of searching the symptoms using fuzzy diagnosis can be inverse to this problem. Specifically, let output of the fuzzy system of logical inference with inputs
\( A'_1 = x_1/5 + x_2/10 + x_3/15 + x_4/20, A'_2 = 1/5 + 0.5/10, A'_3 = 1/36 + 0.9/37 \)
is fuzzy set \( B = 0.5/influenza + 0.5/sharp respiratory disease + 0.4/angina + 0.5/pneumonia. \)Aggregation of the individual outputs leads to the next system of fuzzy relation equations:
\[
\min \left[ \max(x_1 \land 1, x_2 \land 0.5), 0.4 \right] = 0.4,
\]
\[
\min \left[ \max(x_1 \land 1, x_2 \land 0.5), 0.5 \right] = 0.5.
\]
Solving it the value of the symptom “Coughing” will be received, which is described by the fuzzy set “Coughing” = \(0.5/5 + 1/10\).

To determine the probability of event \( A \) in the space of elementary events \( X \), the notion of probable measure is introduces. It is numerical function \( P \), which puts number \( P(A) \) to the elementary event \( A \), besides that
\[
0 \leq P(A) \leq 1, \ P(X) = 1, \ P \left( \bigcup_{i=1}^{\infty} A_i \right) = \sum_{i=1}^{\infty} P(A_i)
\]
for any \( A_1, A_2, \ldots \) such, as \( A_i \cap A_j = 0 \), with \( i \neq j \).

The fuzzy event \( A \) in space \( X \) will be called the fuzzy set
\[
A = \{(x, \mu_A(x)), x \in X \},
\]
where $\mu_A : X \rightarrow [0, 1]$ – membership function of fuzzy set $A$. Then probability of event $A$ can be calculated using the formula

$$P(A) = \sum_{x \in A} \mu_A(x)P(x).$$

Considering this, it is possible to calculate the probabilities of any fuzzy event with provided probability measure. In particular, the probability of symptom “Coughing” at probabilities distribution $P(5) = 0.4, P(10) = 0.4, P(15) = 0.1, P(20) = 0.1$

can be calculated in the following way:

$$P(“Coughing”) = 0.5 \cdot 0.4 + 1 \cdot 0.4.$$

### 3 Bioinformatics

It is known [1], [4] that the problem of recognizing the structures of the proteins of different organization levels is rather complicated. To solve it the different methods and approaches, including experimental (based on physics of chemical relations creation), machine teaching (used the data bases of experimentally found secondary structures as learning samples), probabilistic (on the basis of the Bayes procedures and Markov chains) are used.

The method of recognition of the secondary structure of DNA using fuzzy systems of logical inference is proposed. The problem is the following: it is necessary to build the fuzzy system of logical inference which using random amino acid sequence would define (as an fuzzy set) the secondary structure of central remainder (of the amino acid) of the input sequence.

To solve this problem at first it is necessary to design the fuzzy specification of the problem according to learning samples. One of the methods to build the system of fuzzy instructions according to numerical data consists of the following. Let’s the rules base with $n$ inputs and one output is created. There are learning dates (samples) as the sets of pairs for that

$$(x_1(i), x_2(i), \ldots, x_n(i); d(i)), \ i = 1, 2, \ldots, m,$$

where $x_j(i)$ – inputs and $d(i)$ – output, at that $x_j(i) \in \{a_1, a_2, \ldots, a_k\}$, $d(i) \in \{b_1, b_2, \ldots, b_l\}$. It is necessary to build the fuzzy system of logical inference which would generate the correct output data according to random input values. The algorithm of solving of the provided problem consists in the following sequence of steps:

1. Dividing the space of inputs and outputs for areas (dividing learning data for groups on $m_1, \ldots, m_k$ lines, which means, each input and output is divided for $2N + 1$ cuts where $N$ for each input is selected individually. Separate areas (segments) will be called in the following way:

$M_N(left\ N), \ldots, M_1(left\ 1), S(medium), D_1(right\ 1), \ldots, D_N(right\ N).$

Determination membership function for each areas.
2. Building fuzzy sets on the basis of learning samples (Each group \( m_i \) learning data

\[(x_1(1), x_2(1), \ldots, x_n(1); d(1))\]
\[(x_1(2), x_2(2), \ldots, x_n(2); d(2))\]
\[(x_1(m_i), x_2(m_i), \ldots, x_n(m_i); d(m_i))\]

is compared with the fuzzy sets of the form:

\[A^{(m_i)}_1 = \frac{|a^{(1)}_1|}{m_i}/a_1 + \cdots + \frac{|a^{(1)}_k|}{m_i}/a_k,\]

\[A^{(m_i)}_n = \frac{|a^{(n)}_1|}{m_i}/a_1 + \cdots + \frac{|a^{(n)}_k|}{m_i}/a_k,\]

\[B^{(m_i)} = \frac{|b_1|}{m_i}/b_1 + \cdots + \frac{|b_l|}{m_i}/b_l,\]

where \(|a^{(j)}_1|\) – number of symbols \(a_1\) in column \(j\) of the learning data group, \(|b_j|\) – number of symbols \(b_j\) in the last column of the learning data group.

3. Building fuzzy rules on the basis of fuzzy sets from the previous step on the following scheme:

\[(x_1(1), x_2(1), \ldots, x_n(1); d(1))\]
\[(x_1(2), x_2(2), \ldots, x_n(2); d(2))\]

\[\Rightarrow (x_1(m_1), x_2(m_1), \ldots, x_n(m_1); d(m_1))\]

\[\Rightarrow R^{(1)}: \text{if } x_1 \text{ is } A^{(m_i)}_1 \text{ and } x_2 \text{ is } A^{(m_i)}_2 \text{ and } \ldots \text{ and } x_n \text{ is } A^{(m_i)}_n \text{ then } y \text{ is } B^{(m_i)}\]

4. Elimination of contradictions.

This algorithm puts in accordance to each set of learning data the fuzzy rule of the logical inference.

It will be shown how to use the suggested algorithm of building the fuzzy sets for recognizing the secondary structure of DNA.

It is known [1], [4], that the secondary structure of the pieces of polypeptide sequence is determined mainly by the interactions of neighbor amino acids within these pieces. To be more exact, the type of secondary structure of the exact remain is determined by its surrounding.
To build the fuzzy rules of logical inference the learning samples from 15 remains of the protein MutS [2] are used which look like the following:

\[
\begin{align*}
K & \ V \ S \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \\
V & \ S \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ L \\
S & \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ L \ D \\
E & \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ D \ A \\
G & \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ A \ A \\
\end{align*}
\]

The prediction belongs to the central remain, besides the following denotations are used: \(h\) – for spiral, \(e\) – for cylinder, "-" – other.

According to the algorithm, the teaching data is divided, for example, for 3 groups:

\[
\begin{align*}
K & \ V \ S \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \\
V & \ S \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ L \\
S & \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ L \ D \\
E & \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ D \ A \\
G & \ G \ L \ I \ R \ E \ G \ Y \ D \ P \ D \ A \ A \\
\end{align*}
\]

and compared to each group with according fuzzy sets

\[
A_i^{(m_1)}, A_i^{(m_2)}, A_i^{(m_3)}, B_i^{(m_1)}, B_i^{(m_2)}, B_i^{(m_3)}.
\]

Then fuzzy specification of the recognition problem will look like:

\[
\begin{align*}
R^{(1)} : & \text{ if } x_1 \text{ is } A_1^{(m_1)} \text{ and } x_2 \text{ is } A_2^{(m_1)} \text{ and } \ldots \text{ and } x_{14} \text{ is } A_{14}^{(m_1)} \text{ then } y \text{ is } B_i^{(m_1)} \\
R^{(2)} : & \text{ if } x_1 \text{ is } A_1^{(m_2)} \text{ and } x_2 \text{ is } A_2^{(m_2)} \text{ and } \ldots \text{ and } x_{14} \text{ is } A_{14}^{(m_2)} \text{ then } y \text{ is } B_i^{(m_2)} \\
R^{(3)} : & \text{ if } x_1 \text{ is } A_1^{(m_3)} \text{ and } x_2 \text{ is } A_2^{(m_3)} \text{ and } \ldots \text{ and } x_{14} \text{ is } A_{14}^{(m_3)} \text{ then } y \text{ is } B_i^{(m_3)}
\end{align*}
\]

Using the algorithm of solving the specification, we will find the output received system of fuzzy instructions, if to the input the following amino acid sequence is supplied:

\[
L \ K \ V \ S \ E \ G \ G \ L \ I \ R \ E \ G \ Y \ D \ P.
\]

In accordance with the procedure of executing the algorithm we will get that the secondary structure of the remainder \(L\) is \(h\).

### 4 Fuzzy Boolean Variables

The algebra of statement is one of the chapters of classic mathematical logic. The statement means the variable which can be of two possible values – 0 or 1. Such variable is
called Boolean. In some cases generalizing the notion of Boolean variable to the notion of fuzzy Boolean variable is useful [3]. The fuzzy Boolean variable is called a variable \( p \), which takes a value from the interval \([0, 1]\).

Let \( p \) and \( q \) – fuzzy Boolean variables. Logical operations with such variables are determined like that:
\[
\bar{p} = 1 - p,
\]
\[
p \land q = \min(p, q),
\]
\[
p \lor q = \max(p, q).
\]

From determination of the operations we get, that the following laws
\[
p \lor \bar{p} = 1,
\]
\[
p \land \bar{p} = 0
\]
are broken.

Let \( p_1, p_2, \ldots, p_n \) – fuzzy Boolean variables. Function \( f(p_1, \ldots, p_n) \) is called a function of fuzzy Boolean variables if it takes value on the interval \([0, 1]\).

Function \( f \) of fuzzy Boolean variables is called analytical, if it can be presented by the formula, which includes operations \( \neg \), \( \land \), \( \lor \).

Since in the fuzzy logic above-mentioned laws are violated, function
\[
p \rightarrow q = \bar{p} \cdot \bar{q} \lor \bar{p} \cdot q \lor p \cdot q
\]
cannot be simplified.

One of the tasks Boolean variables function analysis consists in the following. It is necessary to find out under the which conditions the values of analytical function, for example \( f(p, q) = p \land q \), includes in a given interval \([\alpha, \beta]\) of the segment \([0, 1]\) under the condition, that \( p \in [a_1, a_2] \subseteq [0, 1] \) and \( q \in [b_1, b_2] \subseteq [0, 1] \). Solution to this problem gives the possibility to calculate probabilities of varying weather in the following way.

It is know, that weather forecasters evaluate their forecasts using probability theory point out their forecasts with expressions:

“sunny” with probability \( p \in [0.7, 0.8] \);
“windy” with probability \( q \in [0.3, 0.5] \);
“cloudy” with probability \( h \in [0.8, 0.9] \).

Let us consider Boolean analytical function
\[
f(p, q) = p \rightarrow q = \bar{p} \cdot \bar{q} \lor \bar{p} \cdot q \lor p \cdot q.
\]
Let us suppose, that \( p \in [0.7, 0.8], q \in [0.3, 0.5] \). It is necessary to find out in which intervals of the segment \([0, 1]\) values of the function includes. In other words, it is necessary to find which interval contains the probability that the weather will be “sunny“ and “windy“, or “not sunny“ and “windy“, or “not sunny“ and “not windy“.

Considering, that \( p \rightarrow q = \bar{p} \cdot \bar{q} \lor \bar{p} \cdot q \lor p \cdot q \) and that
\[
p \in [0.7, 0.8], \ q \in [0.3, 0.5]
\]
we find that
\[ \tilde{p} \in [0.2, 0.3], \; \tilde{q} \in [0.5, 0.7]. \]
So
\[ \tilde{p} \cdot \tilde{q} = \min(\tilde{p}, \tilde{q}) \in [0.2, 0.3], \]
\[ \hat{p} \cdot q = \min(\hat{p}, q) \in [0.2, 0.3], \]
\[ p \cdot q = \min(p, q) \in [0.3, 0.5]. \]
Then
\[ \tilde{p} \cdot \tilde{q} \lor \hat{p} \cdot q \in [0.2, 0.3], \]
\[ \hat{p} \cdot q \lor \tilde{p} \cdot \tilde{q} \lor p \cdot q \in [0.3, 0.5]. \]
That is why value of the function \( f(p, q) \) will includes to the interval \([0.3, 0.5]\).

Let us consider another option of the problem. Let us suppose, that as earlier, \( p \in [0.7, 0.8], q \in [0.3, 0.5] \) analytical Boolean function is unknown, but is known the interval of the segment \([0, 1]\), in which the values of this function are includes. In this case – this is the interval \([0.3, 0.5]\). It is necessary to find out in which interval of the segment \([0, 1]\) the values of this unknown function included, if for example \( p \in [0.5, 0.6], q \in [0.3, 0.5] \).

One of the approaches to solve this problem consists in building and researching it is so called linguistic model [2], [8]. In this model the variables \( p, q, f \) will be considered as linguistic variables, and appropriate intervals will be described by the fuzzy sets of the space \([0, 1]\). So, our fuzzy model will look like the following:

\[ R: \text{if } p \in A_1 \land q \in A_2 \text{ then } f \in B \]

where \( A_1 = 1/[0.7, 0.8], A_2 = 1/[0.3, 0.5], B = 1/[0.3, 0.5] \). It is necessary to find the output \( B' \) of this fuzzy rule with inputs \( A_1 = 1/[0.6, 0.8], A_2 = 1/[0.3, 0.5] \). After calculating according to the procedure we will have:

\[ B'(y) = 1/[0.3, 0.5]. \]

One more position of the task consists in the following. Let \( p \in [0.5, 0.6] \),
Let us calculate the probability of windy under the condition that \( p \rightarrow q \in [0.3, 0.5] \).
Let us suppose, that \( q \in [b_1, b_2] \subseteq [0, 1] \). Then
\[ p \land q \in [\min(0.5, b_1), \min(0.6, b_2)], \]
\[ \tilde{p} \land q \in [\min(0.4, b_1), \min(0.5, b_2)], \]
\[ \tilde{p} \land \tilde{q} \in [\min(0.4, 1 - b_2), \min(0.5, 1 - b_1)], \]
\[ p \lor \tilde{q} \in [\max(\min(0.5, b_1), \min(0.4, b_1)), \max(\min(0.6, b_2), \min(0.5, b_2))], \]
\[ p \lor \tilde{q} \lor \tilde{p} \lor \tilde{q} \lor \tilde{p} \lor \tilde{q} \in [\max(\min(0.5, b_1), \min(0.5, b_1), \min(0.5, 1 - b_1)), \max(\min(0.6, b_2), \min(0.4, b_2), \min(0.4, 1 - b_2))]. \]
Here we get:

\[
\max(\min(0.5, b_1), \min(0.5, 1 - b_1)) > 0.3, \\
\max(\min(0.6, b_2), \min(0.4, b_2), \min(0.4, 1 - b_2)) \leq 0.5.
\]

From first correlation we find \(b_1\):

\[
0.5 \leq b_1 \Rightarrow \quad b_1 \leq 0.5 \Rightarrow \quad b_1 \text{ - arbitrary}
\]

From second correlation we find \(b_2\):

\[
0.4 \leq 0.6 \leq b_2 \Rightarrow \quad 0.4 \leq b_2 \leq 0.6 \Rightarrow 0 \leq b_2 \leq 0.5, \\
b_2 \leq 0.4 \leq 0.6
\]

So, the probability of windy weather \(q \in [0, 0.5]\).

## 5 Conclusions

The proposed approach to solving problems (based on fuzzy models) allows to simplify the methods of solving problems. Other approaches may be based on possibility theory. But there is a necessity for additional studies of the reliability issues in the first and in the second case.

## References

Enhanced Error Correction Algorithm for RBF Neural Networks

Pawel Rozycki and Janusz Kolbusz

University of Information Technology and Management in Rzeszow,
{prozycki,jkolbusz}@wsiz.rzeszow.pl

Abstract. Using RBF units in neural networks are very interesting option that make network more powerful. The paper presents new training algorithm based on second order ErrCor algorithm. The effectiveness of proposed algorithm has been confirmed by several experiments.

Key words: Error Correction, ErrCor, RBF networks, training algorithms

1 Introduction

The rapid development of intelligent computational systems allowed to solve thousands of practical problems using neural networks. Major achievements have been made mainly using architecture MLP (Multi-Layer Perception), but it turns out that it is also possible with other neural network architectures. Although EBP (Error Back Propagation) [1] caused a real breakthrough, it turned out to be a very slow algorithm, not capable of learning other than MLP, compact network architectures. Most visible progress in this field was develop the LM (Levenberg-Marquardt) algorithm to train the neural network. This algorithm is able to teach the network by 100 to 1000 times less iterations, but its usage to more complex problems is significantly limited, since the size of the Jacobian matrix is proportional to the number of patterns.

In order to solve more and more complex problems with the use of neuron networks we should thoroughly understand the neural network architecture and its impact on the operation of the system and finally develop appropriate processes of learning these networks. Modification of existing algorithms and development of new algorithms for network learning will allow for faster and more effective network teaching.

Often used networks MLP have limited capabilities[1], but new neural network architectures like BMLP (Bridged MLP) [1,2] or DNN (Dual Neutral Networks) [2] with the same number of neurons can solve problems up to 100 times more complex [2,3]. Therefore, it can be concluded that the way neurons interconnections in the network is fundamental.

The use of appropriate architecture has a significant impact on the solution of given problem. An example can be FCC (Fully Connected Cascade) network architecture. Such a network with 10 neurons can solve the Parity-1023 problem, while the most widely used the MLP architecture network with 10 neurons in the three-tiered, one hidden layer, architecture, is able to solve Parity-9 problem. Thus, moving away from the commonly used architecture MLP, while maintaining the same number of neurons
can increase network capacity, even a hundred times. [2-4]. However, a problem arises in that the currently known network learning algorithms, such as EBP [5], or LM do not deal with such network architectures. LM algorithm is not able to teach other architectures than the MLP, because the size of Jacobian, which must be processed is proportional to the number patterns of learning, which limits LM algorithm for solving network learning a relatively small problems. The only known algorithm that can learn these new architectures is NBN algorithm (Neuron-by-Neuron) [6-8]. It is faster than LM and can be used for all architectures, including BMLP, FCC DNN and MLP, of course, and gives good learning results. However, published in 2012 ISO algorithm [9] and published in 2014 ErrCor (Error Correction) algorithm [10], allow to get even better results.

2 Enhanced Error Correction Algorithm

2.1 Error Correction Fundamentals

Error Correction is second order LM based algorithm that has been designed for RBF networks where as neurons RBF units with Gaussian activation function defined by (1) are used.

$$\varphi_h(x_p) = \exp\left(-\frac{\|x_p - c_h\|^2}{\sigma_h}\right)$$

where: $c_h$ and $\sigma_h$ are the center and width of RBF unit $h$, respectively. $\| \cdot \|$ represents the computation of Euclidean Norm. The output of such network is given by:

$$O_p = \sum_{h=1}^{H} w_h \varphi_h(x_p) + w_o$$

where: $w_h$ presents the weight on the connection between RBF unit $h$ and network output. $w_o$ is the bias weight of output unit. Note that the RBF networks can be implemented using neurons with sigmoid activation function in MLP architecture [11,12]. The main idea of the ErrCor algorithm is increasing the number of RBF units one by one and adjusting all RBF units in network by training after adding of each unit. New unit is initially set to compensate largest error in the current error surface and after that all units are trained changing both centers and widths as well as output weights. Details of algorithm can be found in [10].

As can be found in [10, 13] ErrCor algorithm had been successfully used to solve several problems like function approximation, classification or forecasting. The main disadvantage of ErrCor algorithm is long computation time caused mainly by requirement of training of whole network at each iteration.

2.2 Enhanced ErrCor

Long computation time depends on many factors. One of the most important is number of patterns used in training and long training of whole network after adding of next
RBF unit. In order of improve this process we suggest the following modifications of ErrCor algorithm:

- after adding new RBF unit only this new unit is trained using LM-based method used in ErrCor algorithm [10] and after that all output weights are justified using regression;

- after added N new RBF units whole network is trained using the same LM-based method used in ErrCor algorithm where N is arbitrary assigned value.

Such modification allow to shortened training process because critical whole training process is limited to cases when N new units are added to network. In the other cases the training is much faster because in fact trained is only one RBF unit and regression is quite small absorbing process.

Pseudo code of the enhanced ErrCor algorithm is shown below. Changes to original ErrCor algorithm [10] are bolded.

**Enhanced ErrCor pseudo code**

evaluate error of each pattern;
while 1
  C = pattern with biggest error;
  add a new RBF unit with center = C;
  if N new RBF units are added
    train the whole network using LM-based method;
  else
    train only one new added RBF unit using LM-based method;
    adjust output weights for whole network by regression
  end
evaluate error of each pattern;
calculate SSE = Sum of Squared Errors;
if SSE < desired SSE
    break;
end;
end

In the next section some experimental results for this approach is presented.

3 Experiments Results

To confirm suggested approach several experiments for different approximation benchmark functions and training parameters have been prepared. The following functions have been selected: Peaks function, Second Shaffer function and Shwefel function. In the next three subsections the ErrCor algorithm and the Enhanced ErrCor algorithm have been used to solve approximation problem of mentioned functions. In all experiments 900 training patterns and 3481 testing patterns have been generated. For such prepared data series experiments have been done with different values of parameter N and compared to results achieved using original ErrCor algorithm. To prepare experiments Matlab 2009b software with Windows 7 64 on Intel Core i5-M560 CPU and 8GB platform was used.

3.1 Shwefel Function

First experiment was prepared for Shwefel function given by

\[ z(x, y) = 2 \times 418.9829 - x \sin(\sqrt{|x|}) - y \sin(\sqrt{|y|}) \]  

(3)

shown in Figure 2.

![Figure 2. Surface of normalized Shwefel function.](image)
Results achieved for Shwefel function are shown in Table 3. Result for original ErrCor that can be treated as a reference is denoted as OrgErrCor. Parameter N means the number of units that are added to network between full training. The case when training process is done without full network training is denoted as X in column N. The RMSE is Root Mean Square Error given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (out_T - out_E)^2}{n}}$$

where $out_T$ is the output of trained network and $out_E$ is expected value and $n$ is the number of patterns.

<table>
<thead>
<tr>
<th>N</th>
<th>Training time</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrgErrCor</td>
<td>246.6722</td>
<td>0.0038543</td>
<td>0.0038299</td>
</tr>
<tr>
<td>2</td>
<td>215.7920</td>
<td>0.0032388</td>
<td>0.0031717</td>
</tr>
<tr>
<td>3</td>
<td>148.7899</td>
<td>0.0030517</td>
<td>0.0029844</td>
</tr>
<tr>
<td>4</td>
<td>88.0451</td>
<td>0.0052154</td>
<td>0.0051204</td>
</tr>
<tr>
<td>5</td>
<td>75.5889</td>
<td>0.0053222</td>
<td>0.0051949</td>
</tr>
<tr>
<td>6</td>
<td>59.7885</td>
<td>0.0052533</td>
<td>0.0051889</td>
</tr>
<tr>
<td>7</td>
<td>51.9710</td>
<td>0.0057257</td>
<td>0.0057030</td>
</tr>
<tr>
<td>8</td>
<td>66.9600</td>
<td>0.0079042</td>
<td>0.0077081</td>
</tr>
<tr>
<td>9</td>
<td>60.3580</td>
<td>0.0054597</td>
<td>0.0053507</td>
</tr>
<tr>
<td>10</td>
<td>51.2228</td>
<td>0.0059679</td>
<td>0.0058742</td>
</tr>
<tr>
<td>15</td>
<td>43.1061</td>
<td>0.0105723</td>
<td>0.0104105</td>
</tr>
<tr>
<td>30</td>
<td>36.2405</td>
<td>0.0497498</td>
<td>0.0498103</td>
</tr>
<tr>
<td>X</td>
<td>21.1715</td>
<td>0.0747471</td>
<td>0.0732735</td>
</tr>
</tbody>
</table>

As shown in Table 1 training time decreases with increased value of N. This is obvious because frequency of full training, that is the most time consuming part of training process is smaller for higher N. More important is that values of testing and training RMSE for small values of N (2 and 3) are better than these achieved with original ErrCor, and for higher value of N are only slightly worse. Note that results for N=10 are only 53% worse but achieved almost 5 times faster.

### 3.2 Second Shaffer Function

The second experiment have been done for Second Shaffer function. This function is given by

$$z(x, y) = 0.5 + \frac{sin^2(x^2 - y^2) - 0.5}{[1 + 0.001(x^2 - y^2)]^2}$$

shown in Figure 3.
Fig. 3. Surface of normalized Second Shaffer function.

Results achieved with Enhanced Error Correction algorithm is shown in Table 2. Similarly like for Shwefel function training time decreases with N while RMSE is relatively are close to or even lower than for original ErrCor.

Figure 4 show the training process that show changes of training and testing RMSE during training process. Training RMSE is drown as blue circles and testing RMSE is drawn as a red stars. As can be observed in the case of Enhanced ErrCor with N = 2 reaches similar result as ErrCor but is able to obtain same results faster and using less neurons.

Table 2. Results for Second Shaffer function.

<table>
<thead>
<tr>
<th>N</th>
<th>Training time</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrgErrCor</td>
<td>262.1465</td>
<td>0.0008487</td>
<td>0.0046744</td>
</tr>
<tr>
<td>2</td>
<td>160.7057</td>
<td>0.0005260</td>
<td>0.0046379</td>
</tr>
<tr>
<td>3</td>
<td>140.3324</td>
<td>0.0005305</td>
<td>0.0046374</td>
</tr>
<tr>
<td>4</td>
<td>123.2476</td>
<td>0.0008320</td>
<td>0.0051382</td>
</tr>
<tr>
<td>5</td>
<td>106.8903</td>
<td>0.0014932</td>
<td>0.0047826</td>
</tr>
<tr>
<td>6</td>
<td>50.9329</td>
<td>0.0018604</td>
<td>0.0049701</td>
</tr>
<tr>
<td>7</td>
<td>44.1457</td>
<td>0.0011056</td>
<td>0.0047119</td>
</tr>
<tr>
<td>8</td>
<td>43.9627</td>
<td>0.0010136</td>
<td>0.0046913</td>
</tr>
<tr>
<td>9</td>
<td>28.2079</td>
<td>0.0018110</td>
<td>0.0049119</td>
</tr>
<tr>
<td>10</td>
<td>57.9185</td>
<td>0.0010418</td>
<td>0.0047068</td>
</tr>
<tr>
<td>15</td>
<td>21.7188</td>
<td>0.0017827</td>
<td>0.0049438</td>
</tr>
<tr>
<td>30</td>
<td>17.5358</td>
<td>0.0093627</td>
<td>0.0103736</td>
</tr>
<tr>
<td>X</td>
<td>14.4285</td>
<td>0.0093627</td>
<td>0.0103736</td>
</tr>
</tbody>
</table>
3.3 Peaks Function

The last experiment with described Enhanced Error Correction algorithm has been used for approximation of Peaks function given by:

\[
z(x, y) = -\frac{1}{30} e^{(-1-6x-9x^2-9y^2)} + \\
- (0.6x - 27x^3 - 243y^5) e^{(-9x^2-9y^2)} + \\
(0.3 - 1.8x + 2.7x^2) e^{(-1-6y-9x^2-9x^2)}
\]

and shown in Figure 5.

Results achieved for this function in the same way like for previous functions are shown in Table 3. Unfortunately, they are not so clear like for previous functions. While training time decreases with N in the same time RMSE values increase. Examples of training process by original ErrCor and Enhanced ErrCor with N=3 is presented in Figure 6. As can be observed the full network training is seen as a rapid RMSE decreasing while adding and training of one RBF unit initially produces similar effect but later does not decrease RMSE. In the case when N value is higher than maximal number of units in the network the training is limited to adding new units and training then one-by-one without full network training. Such training process is shown in Figure 7. Note that starting from 14th unit added to network RMSE values are not decreasing. This is because each new unit added to network is localized according to the pattern with the highest error and in these case each new unit, starting from 14th, is initially localized in the same place.

4 Conclusions

Achieved results confirm effectiveness of suggested method for improvement Error Correction algorithm that is currently one of the most powerful for training RBF networks. Proposed modification allows to reduce training time in most cases without
Fig. 5. Surface of Peaks function.

Table 3. Results for Peaks function.

<table>
<thead>
<tr>
<th>N</th>
<th>Training time</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrgErrCor</td>
<td>710.2437</td>
<td>0.0000071</td>
<td>0.0000071</td>
</tr>
<tr>
<td>2</td>
<td>418.5150</td>
<td>0.0000234</td>
<td>0.0000234</td>
</tr>
<tr>
<td>3</td>
<td>255.5103</td>
<td>0.0000580</td>
<td>0.0000580</td>
</tr>
<tr>
<td>4</td>
<td>137.8621</td>
<td>0.0001049</td>
<td>0.0024100</td>
</tr>
<tr>
<td>5</td>
<td>75.8098</td>
<td>0.0015600</td>
<td>0.0018230</td>
</tr>
<tr>
<td>6</td>
<td>72.7661</td>
<td>0.0079510</td>
<td>0.0080396</td>
</tr>
<tr>
<td>7</td>
<td>136.9426</td>
<td>0.0001695</td>
<td>0.0001710</td>
</tr>
<tr>
<td>8</td>
<td>98.7397</td>
<td>0.0024247</td>
<td>0.0024510</td>
</tr>
<tr>
<td>9</td>
<td>28.9673</td>
<td>0.0191530</td>
<td>0.0231557</td>
</tr>
<tr>
<td>10</td>
<td>25.4868</td>
<td>0.0107400</td>
<td>0.0108643</td>
</tr>
<tr>
<td>15</td>
<td>54.5841</td>
<td>0.0014544</td>
<td>0.0014681</td>
</tr>
<tr>
<td>30</td>
<td>29.8234</td>
<td>0.0464481</td>
<td>0.0470313</td>
</tr>
<tr>
<td>X</td>
<td>17.1136</td>
<td>0.0464481</td>
<td>0.0470313</td>
</tr>
</tbody>
</table>
Fig. 6. Training process for approximation of Peaks function with: (a) original ErrCor algorithm, (b) Enhanced ErrCor (N=3).

Fig. 7. Training process for approximation of Peaks function with Enhanced ErrCor without full network training.
losses of low training and testing errors. Further work will be focused on improve-
ment of proposed algorithm by correction of method for selection of initial localization
for new RBF units and on applying described algorithm for wider spectrum of functions
and real world classification datasets from UCI Machine Learning Repository.

References

1. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-
2. S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture". In D. S. Touret-
Mateo, CA, 1990, pp. 524-532
Connectionists Models Summer School, Morgan Kaufman.
4. B. M.Wilamowski, "Challenges in Applications of Computational Intelligence in Industrial
Electronics", IEEE International Symposium on Industrial Electronics (ISIE 2010), Jul 04-07,
5. Y. Bengio, "Learning deep architectures for AI". Foundations and Trends in Machine Learn-
Trans. on Neural Networks, vol. 21, no.11, pp1793-1803, Nov. 2010.
Method for Training Radial Basis Function Networks", IEEE Transactions on Neural Net-
Basis Function Networks", IEEE Transactions on Neural Networks and Learning Systems,
11. B. M. Wilamowski, R. C. Jaeger, "Implementation of RBF type networks by MLP net-
works", IEEE International Conference on Neural Networks (ICNN 96), pp. 1670-1675
12. X. Wu, B.M. Wilamowski "Advantage analysis of sigmoid based RBF networks". In: Pro-
ceedings of the 17th IEEE International Conference on Intelligent Engineering Sys-
13. C. Cecati, J. Kolbusz, P. Rozycki, P. Siano, B. Wilamowski, "A Novel RBF Training Al-
gorithm for Short-Term Electric Load Forecasting and Comparative Studies", IEEE Trans. on
Ind. Electronics, Early Access, 2015.
The Minimization Method of Boolean Functions in Polynomial Set-theoretical Format

Bohdan Rytsar
National University “Lviv Polytechnic”
S. Bandera Str., 12, Lviv, 79646, Ukraine
and
University of Rzeszow, Chair of Computer Science
Prof. S. Pigonia Str. 1, 35-310 Rzeszow, Poland
bohdanrytsar@gmail.com

Abstract. A generalized of conjuncterms simplification rules in polynomial set-theoretical format has been considered. These rules are based on the proposed theorems for different initial conditions transform of pair conjuncterms, Hamming distance between them can be arbitrary. These rules may be useful to minimize in polynomial set-theoretical format of arbitrary logic functions with \( n \) variables. Advantages of the proposed rules of simplification are illustrated by several examples.

Key words: Boolean function, polynomial set-theoretical format, simplification of conjuncterms, Hamming distance

1 Introduction

Investigations [1–8] have shown that it is economically profitable to build digital devices such as arithmetic units, coding-error detectors as well as devices with programmed logic, etc. on logical elements AND-EXOR, which realize polynomial basis \{\&, \oplus, 1\}, that is AND, EXCLUSIVE OR (EXOR) logical operations and constant 1. It is easier to test and diagnose [9–11] digital devices on AND-EXOR if compared to the devices built on AND-OR. However, in spite of the mentioned advantages it is more difficult to minimize a function in polynomial format, i.e. in ESOP (EXOR Sum-Of-Product), than in disjunctive format, i.e. in SOP (Sum-Of-Product). If the merge operation of adjacent conjuncterms (conjunction of literals) is only applied in the SOP minimization, than, in addition, the same operations can be applied in the ESOP minimization [1, 2].

A precise solutions of a minimization problem in ESOP generally are based on analytical [2] or on visual transformations [1–3]. Respectively, such methods are suitable only for functions from small amount of variables [5, 7, 10–13] and only for special classes for functions with up to 10 variables [14]. Heuristic methods have comparatively wider practical application [1, 8, 16–23]. Among them there are minimization method based on a coefficient of generalized canonical Reed-Muller forms using of matrix transformations [1, 8, 11, 16] and the method based on iterative execution of operations with conjuncterms of different ranks of the given function. To the last belongs
the algorithm [17], which after transformation of the given function in Positive Polarity Reed-Muller expression minimizes it on the basis of three operations with conjuncterms. Better results have been shown by algorithm based on the procedure of so-called linked product terms [18, 19]. Later, on the basis of this procedure, the algorithms have been developed and completed with more perfect operations (that is primary xlink, secondary xlink, unlink, exorlink), which can be used for minimization of a system of complete and incomplete functions [20, 21].

However, the mentioned above algorithms have one drawback in common. They involve the procedure of linking in pairs only conjunctermsof the same rank \( r \in \{1, 2, \ldots, n\} \), which differs between each other by binary positions. Correspondingly, this limits the use of such algorithms to functions given in SOP or ESOP, which can have triple conjuncterms in the different part. In these cases to conjuncterms that differ in ranks certain procedures of transformation are applied leading to an increase of procedural steps and processing time. Besides, the above mentioned operations of conjuncterms linking and other rules of simplification [24–26] do not have generalized character as to Hamming distance between any two conjuncterms of different ranks of a given function that does not guarantee the final minimized result.

In this paper we consider a new method of minimization of Boolean functions with \( n \) variables in polynomial set-theoretical format (PSTF), based on a procedure of splitting of conjuncterms [27–29] and on usage of generalized set-theoretical rules of conjuncterms simplification [30]. The suggested rules guarantee better (as to costs of realization and number of procedure steps) results of minimization of logic functions proved by the numerous examples that are borrowed from publications of other authors for comparison purposes.

2 Problem Formulation

Boolean function \( f(x_1, x_2, \ldots, x_n) \) that undergoes minimization, will be given by a set of binary minterms or in perfect set-theoretical form (STF) as a \( Y_1 = \{m_1, m_2, \ldots, m_k\} \), or in perfect polynomial set-theoretical form (PSTF) as a \( Y_\oplus = \{m_1, m_2, \ldots, m_k\}_\oplus \) [30, 31].

The generalized set-theoretical rules of simplification [30] of a conjuncterm set of any function \( f \), given in PSTF \( Y_\oplus \), are based on iterative process of simplification of two conjuncterms \( \theta_1^1 = (\sigma_1 \sigma_2 \cdots \sigma_n) \) and \( \theta_2^2 = (\sigma_1 \sigma_2 \cdots \sigma_n) \), \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1\} \), which differ in Hamming difference \( d = 1, 2, \ldots, n \) it is number of different in value \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1\} \) of onename positions. Here the different part \( \alpha, \beta, \gamma, \delta, \ldots \) of these conjuncterms may have a different total number of literals \( k_\ell \). For example, two pairs of conjuncterms \( \begin{pmatrix} 11-01 \\ 01-10 \end{pmatrix} \) and \( \begin{pmatrix} 11---1 \\ 01---0 \end{pmatrix} \) have \( d = 3 \), but the first has \( k_\ell = 6 \), and the second \( k_\ell = 5 \). Therefore, different initial conditions of transformation of two conjuncterms are possible.

We will consider the following conditions:
• when \( k_\ell = 2d \), here two conjuncterms are of the same \( r \)-rank \( \theta_1^r \) and \( \theta_2^r \) but differ in \( d \) onename binary positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1\} \);
• when \( k_l = 2d - 1 \), here one conjuncterm of \((r-1)\)-rank \( \theta_1^{r-1} \) and the second of \( r \)-rank \( \theta_2^r \) differ in \( d \) onename positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1, \} \), where dash \((-\)\) belongs to \( \theta_1^{r-1} \);

• when \( k_l = 2(d - 1) \), here two conjuncterms are of the same \((r-1)\)-rank \( \theta_1^{r-1} \) and \( \theta_2^{r-1} \) differ in \( d \) onename positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1, \} \) and each of them has one dash \((-\)\).

As a result of transformation of two conjuncterms in PSTF a transformed PSTF \( Y^\oplus \) will be formed, where power \( k_Y \) will depend on distance \( d \). Efficiency of simplification of two different conjuncterms for the mentioned above initial conditions will be estimated on the basis of comparison of interrelation \( k_\theta/k_l \), obtained on the ground of data of transformed PSTF \( Y^\oplus \), where \( k_\theta^\oplus \) is number of transformed conjuncterms and \( k_l^\oplus \) is number of their literals, with initial interrelation \( k_\theta/k_l \), where in this case \( k_\theta = 2 \).

### 3 Three Theorems about Conjuncterms Transformation

**Theorem 1.** Two conjuncterms of \( r \)-rank \( \theta_1^r \) and \( \theta_2^r \), \( r \in \{1, 2, \ldots, n\} \), of the function \( f(x_1, x_2, \ldots, x_n) \), that differ in values \( d \) of onename binary positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1, \} \), in polynomial set-theoretical format form a set of transformed PSTF \( Y^\oplus \) of power \( k_Y = d \), each of them consists of \( k_\theta^\oplus = d \) conjuncterms of \((r-1)\)-rank and has in different part the total number of literals \( k_l^\oplus = d(d - 1) \).

**Proof.** To determine \( k_\theta^\oplus/k_l^\oplus \) and \( k_Y \) let us consider the transformation of conjuncterms \( \theta_1^r \) and \( \theta_2^r \) for \( d = 0, 1, 2, 3, 4 \). Here it should be mentioned that initial interrelation \( k_\theta/k_l = 2/2d \).

- If \( d = 0 \), then \( \theta_1^r = \theta_2^r \). So, transformed PSTF \( Y^\oplus = \{\theta_1^r, \theta_2^r\}^\oplus = \emptyset \), that corresponds to analytical expression \( a \oplus a = 0 \). In this case \( k_\theta^\oplus/k_l^\oplus = 0/0; k_Y = 1 \).
- Let \( d = 1 \). Then \( \theta_1^r = (\sigma_1 \cdots \alpha_i \cdots \sigma_n) \) and \( \theta_2^r = (\sigma_1 \cdots \alpha_i \cdots \sigma_n) \), \( \alpha_i \in \{0, 1\} \).

   Respectively, for analytical expression \( \bar{a} \oplus a = 1 \) we can write:

   \[
   Y^\oplus = \{(\sigma_1 \cdots \bar{\alpha_i} \cdots \sigma_n), (\sigma_1 \cdots \alpha_i \cdots \sigma_n)\}^\oplus = (\sigma_1 \cdots \bar{\alpha_i} \cdots \sigma_n),
   \]

   where the transformed PSTF \( Y^\oplus = \{(\sigma_1 \cdots \bar{\alpha_i} \cdots \sigma_n)\}^\oplus = \theta^{r-1} \) is a triple conjuncterm of \((r-1)\)-rank.

   For (1) interrelation \( k_\theta^\oplus/k_l^\oplus = 1/0 \), and as initial interrelation \( k_\theta/k_l = 2/2 \), then it indicates on a result of transformation (1) the simplification took place due to the replacement of two conjuncterms of \( r \)-rank by one conjuncterm of \((r-1)\)-rank; \( k_Y = 1 \).

   To simplify the writing of the conjuncterms of the given and transformed PSTF \( Y^\oplus \) will be considered only for their different positions which will be written down in a column. In (1) such position is \( \alpha_i \), so, simplified writing down (1) with taking into account \( \alpha_i \equiv \alpha \in \{0, 1\} \), will look like:

   \[
   \binom{\alpha}{\bar{\alpha}} \Rightarrow (-),
   \]

   where \( \Rightarrow \) – operator of transformation of the conjuncterms \( \theta_1^r \) and \( \theta_2^r \) in polynomial format of the function \( f \). In examples of transformation, the same in meaning onename positions of conjuncterms will be rewritten without any change. For
example, \( \begin{pmatrix} 1-01 \\ 1-11 \end{pmatrix} \mapsto (1--1) \), that in decimal format corresponds to \( \begin{pmatrix} 9, 13 \\ 11, 15 \end{pmatrix} \mapsto (9, 11, 13, 15) \) and in analytical form is \( x_1x_3x_4 + x_1x_3x_4 = x_1x_4 \).

- Let \( d=2 \). Then \( \theta_1' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \sigma_n), \theta_2' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \sigma_n), \alpha_i, \beta_j \in \{0,1\} \), and respectively to analytical expressions \( \overline{a\oplus b} \oplus ab = \begin{cases} \overline{a \oplus b} & \text{if } \alpha_i = \beta_j \\ a \oplus b & \text{if } \alpha_i \neq \beta_j \end{cases} \) and \( \overline{a\oplus b} \oplus ab = \begin{cases} \overline{a \oplus b} & \text{if } \alpha_i = \beta_j \\ a \oplus b & \text{if } \alpha_i \neq \beta_j \end{cases} \), in simplified way (for \( \alpha_i \equiv \alpha, \beta_i \equiv \beta, \alpha, \beta, \in \{0,1\} \)) we will obtain

\[
\begin{pmatrix} \bar{\alpha} \bar{\beta} \\ \alpha \beta \end{pmatrix} = \left\{ \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix}, \begin{pmatrix} \alpha \\ \bar{\beta} \end{pmatrix}, \begin{pmatrix} \bar{\alpha} \\ \beta \end{pmatrix}, \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \right\}.
\]

(3)

For (3) we have \( k_0^*/k_1^* = 2/2 \), that is indicative of simplification of the given conjuncters due to reduction of their rank from \( r \) to \( (r-1) \), as initial interrelation \( k_0/k_1 = 2/4; k_Y = 2 \).

- Let \( d=3 \). Then \( \theta_1'' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \bar{\gamma}_k \cdots \sigma_n), \theta_2'' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \bar{\gamma}_k \cdots \sigma_n), \alpha_i, \beta_j, \gamma_k \in \{0,1\} \) for \( \alpha_i \equiv \alpha, \beta_i \equiv \beta, \gamma_i \equiv \gamma, \alpha, \beta, \gamma \in \{0,1\} \) we have

\[
\begin{pmatrix} \bar{\alpha} \bar{\beta} \bar{\gamma} \\ \alpha \beta \gamma \end{pmatrix} = \left\{ \begin{pmatrix} \bar{\alpha} \bar{\beta} \\ \bar{\alpha} \beta \\ \bar{\alpha} \gamma \\ \alpha \beta \\ \alpha \gamma \\ \beta \gamma \end{pmatrix} \right\}.
\]

(4)

For (4) \( k_0^*/k_1^* = 3/6 \) indicates on an increase of power of each transformed PSTF \( Y^{\oplus} \) and unchangeability of number of literals of their conjuncters as initial interrelation \( k_0/k_1 = 2/6; k_Y = 6 \).

- Let \( d=4 \). Then \( \theta_1''' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \bar{\gamma}_k \cdots \bar{\delta}_l \cdots \sigma_n), \theta_2''' = (\sigma_1 \cdots \bar{\alpha}_i \cdots \bar{\beta}_j \cdots \bar{\gamma}_k \cdots \bar{\delta}_l \cdots \sigma_n), \alpha_i, \beta_j, \gamma_k, \delta_l \in \{0,1\} \) for \( \alpha_i \equiv \alpha, \beta_i \equiv \beta, \gamma_i \equiv \gamma, \delta_i \equiv \delta, \alpha, \beta, \gamma, \delta \in \{0,1\} \) we have:

\[
\begin{pmatrix} \bar{\alpha} \bar{\beta} \bar{\gamma} \bar{\delta} \\ \alpha \beta \gamma \delta \end{pmatrix} = \left\{ \begin{pmatrix} \bar{\alpha} \bar{\beta} \bar{\gamma} \\ \bar{\alpha} \beta \bar{\gamma} \\ \bar{\alpha} \gamma \bar{\delta} \\ \alpha \beta \bar{\gamma} \\ \alpha \gamma \delta \\ \beta \gamma \delta \end{pmatrix} \right\}.
\]

(5)

So, for (5) \( k_0^*/k_1^* = 4/12 \) indicates on an increase of power of transformed PSTF \( Y^{\oplus} \) and the number of literals, as the initial interrelation \( k_0/k_1 = 2/8; k_Y = 24 \).

In the case of necessity for any pair of conjuncters of \( r \)-rank of a function \( f \), that have distance \( d > 4 \), one can in similar way form a set of \( \frac{d!}{d!} \) of transformed PSTF \( Y^{\oplus} \); \( d = 1, 2, \ldots, n \).
Based on the considered above, one can state that two conjuncters of \( r \)-rank \( \theta_1^d \) and \( \theta_2^d \) function \( f \), that differ \( d = 1, 2, \ldots, n \) in different by values onename binary positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1\} \), form in polynomial format a set with \( k_Y = d! \) of transformed PSTF \( Y^\oplus \), each of them consisting of different conjuncters of \((r-1)\)-rank with \( k_0^* / k_1^* = d/(d-1) \), that is the proof of Theorem 1.

**Theorem 2.** Two conjuncters of the function \( f(x_1, x_2, \ldots, x_n) \), one of which of \((r-1)\)-rank \( \theta_1^{d-1} \) differs from another \( r \)-rank \( \theta_2^d \) in the number of \( d \) different in values onename positions \( \alpha, \beta, \gamma, \delta, \ldots \in \{0, 1, -\} \), among which the dash (−) belongs to \( \theta_1^{d-1} \), \( r \in \{1, 2, \ldots, n\} \), in polynomial set-theoretical format create \( k_Y = (d-1)! \) of sets of transformed PSTF \( Y^\oplus \), each of them has power \( k_0^* = d \) and the total number of literals in different part \( k_1^* = d(d-1) - (d-2) \), here:

- if \( d = 1 \), then \(
\begin{align*}
\theta &\Rightarrow \overline{\theta} \\
\alpha &\Rightarrow \overline{\alpha}
\end{align*}
\)

- if \( d = 2 \), then \(
\begin{align*}
\alpha, \beta &\Rightarrow \overline{\alpha, \beta} \\
\alpha, \beta &\Rightarrow \overline{\alpha, \beta}
\end{align*}
\)

- if \( d = 3 \), then \(
\begin{align*}
\alpha, \beta, \gamma &\Rightarrow \overline{\alpha, \beta, \gamma} \\
\alpha, \beta, \gamma &\Rightarrow \overline{\alpha, \beta, \gamma}
\end{align*}
\)

- if \( d = 4 \), then \(
\begin{align*}
\alpha, \beta, \gamma, \delta &\Rightarrow \overline{\alpha, \beta, \gamma, \delta} \\
\alpha, \beta, \gamma, \delta &\Rightarrow \overline{\alpha, \beta, \gamma, \delta}
\end{align*}
\)

where \( \alpha, \beta, \gamma, \delta \) are binary positions of any value 0 or 1.

**Proof.** In this case the given PSTF \( Y^\oplus \) has interrelation \( k_0^*/k_1^* = 2/(2d - 1) \).
• Let $d = 1$. Then $\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$, $\theta_{2} = (\sigma_{1} \cdots \sigma_{n})$, $\alpha_{i} \in \{0, 1\}$, and respectively for the expression $1 \oplus \alpha = \bar{\alpha}$, $\alpha \in \{a, \bar{a}\}$, we can write down such PSTF $Y^\oplus$:

$$Y^\oplus = \{(\sigma_{1} \cdots \sigma_{n}), (\sigma_{1} \cdots \sigma_{n})\}^\oplus = \{(\sigma_{1} \cdots \sigma_{n})\}^\oplus.$$ 

As interrelation $k_{\theta}/k_{\beta} = 1/1$, then compared with $k_{\theta}/k_{\beta} = 2/1$ we have simplification of the given PSTF $Y^\oplus$ due to removal of one conjuncterm; $k_{Y} = 1$.

• Let $d = 2$. Then for $\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$, $\theta_{2} = (\sigma_{1} \cdots \sigma_{n})$ and $\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$, $\alpha_{i} \in \{0, 1\}$, we will obtain $Y^\oplus = \{(\sigma_{1} \cdots \sigma_{n})\}$ and $Y_{2} = \{(\sigma_{1} \cdots \sigma_{n})\}$ and $Y_{1} = \{(\sigma_{1} \cdots \sigma_{n})\}$ and $Y_{\oplus} = \{(\sigma_{1} \cdots \sigma_{n})\}$.

Comparing the obtained interrelation $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 2/2$ with $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 2/3$, we see that transformed PSTF $Y^\oplus$ is simpler than the given PSTF $Y^\oplus$ for one literal. It should be noted, that a number of transformed PSTF $Y^\oplus$ is determined by a number of binary positions in different part $\theta_{1}^{-1}$ and $\theta_{2}$, that conforms to Theorem 1. Therefore, for $d = 2$ we have $k_{Y} = 1$.

• Let $d = 3$. Then for $\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$, $\theta_{2} = (\sigma_{1} \cdots \sigma_{n})$ and $\theta_{3} = (\sigma_{1} \cdots \sigma_{n})$, $\alpha_{i} \in \{0, 1\}$, we have:

$$Y^\oplus = \{(\sigma_{1} \cdots \sigma_{n})\}^\oplus = \{(\sigma_{1} \cdots \sigma_{n})\}^\oplus,$$

$$Y_{\oplus} = \{(\sigma_{1} \cdots \sigma_{n})\}^\oplus.$$ 

The obtained $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 3/5$ indicates on an increase by one conjuncterm, since $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 2/5$; $k_{Y} = 2$.

• Let $d = 4$. Based on the considered above and taking into account the rule (4) of Theorem 1 for $d = 3$ (three positions are common), one can state that for

$$\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$$

$$\theta_{2} = (\sigma_{1} \cdots \sigma_{n}),$$

$$\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$$

$$\theta_{2} = (\sigma_{1} \cdots \sigma_{n}),$$

$$\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$$

$$\theta_{2} = (\sigma_{1} \cdots \sigma_{n}),$$

$$\theta_{1}^{-1} = (\sigma_{1} \cdots \sigma_{n})$$

$$\theta_{2} = (\sigma_{1} \cdots \sigma_{n}),$$

the interrelation $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 4/10$ which, compared with $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = 2/7$, indicates on an increase of a number of conjunctermas as well as their literals; $k_{Y} = 6$.

Thus, if a conjuncterm of $(r - 1)$-rank $\theta_{r}^{-1}$ differs from a conjuncterm of $r$-rank $\theta_{r}^{q}$ in the number $d$ of different by value onename positions $\alpha, \beta, \gamma, \delta, \ldots \in \{0, 1\}$, and respectively dash (-) belongs to $\theta_{r}^{-1}$, $r \in \{1, 2, \ldots, n\}$, then $(d - 1)!$ of sets transformed PSTF $Y^\oplus$ will be formed, each of which having the interrelation $k_{\theta_{1}^{-1}}/k_{\theta_{2}} = d/(d(d - 1) - (d - 2))$, that is the proof of Theorem 2. □
Theorem 3. Two conjuncterms of \((r - 1)\)-rank \(\theta_1^{r-1}\) and \(\theta_2^{r-1}\), \(r \in \{1, 2, \ldots, n\}\), of the function \(f(x_1, x_2, \ldots, x_n)\) differ in \(d\) onename binary positions \(\alpha, \beta, \gamma, \delta, \ldots \in \{0, 1, -\}\), where each conjuncterm has one \((-\)\), in polynomial set-theoretical format starting with \(d = 2\), create \(k_Y = (d - 2)!\) of sets transformed PSTF \(Y^\oplus\), each of which has power \(k_\theta = d\) and the total number of literals in the different part \(k_\Gamma = d(d - 1) - 2(d - 2)\) and :

- if \(d = 2\), then 
  \[
  \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right) \quad \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right),
  \]
  (15)

- if \(d = 3\), then 
  \[
  \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right) \quad \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right),
  \]
  (16),(17),(18)

- if \(d = 4\), then 
  \[
  \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right) \quad \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right),
  \]
  (19),(20)

- if \(d = 4\) then 
  \[
  \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right) \quad \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right),
  \]
  (21),(22)

- if \(d = 4\) then 
  \[
  \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right) \quad \left(\begin{array}{cc}
  \alpha & -
  \\
  -\beta & \gamma
  \end{array}\right),
  \]
  (23),(24)

where \(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}\) are binary positions of any value 0 or 1.

Proof. Given PSTF \(Y^\oplus\) has the initial interrelation \(k_\theta/k_\Gamma = 2/2(d - 1)\).

- Let \(d = 2\). Then \(\theta_1 = (\alpha_1 \cdots \hat{\alpha}_i \cdots \beta_j \cdots \sigma_n)\) and \(\theta_2 = (\sigma_1 \cdots \sigma_n)\), \(\hat{\alpha}_i, \hat{\beta}_j \in \{0, 1\}\). For \(f(a, b)\) respectively to (15) we have \(\hat{a} \oplus \hat{b} = (\hat{a} \oplus \hat{1}) \oplus (\hat{b} \oplus \hat{1}) = \hat{a} \oplus \hat{b}\), that corresponds to PSTF \(Y^\oplus = \{\gamma_1 \cdots \hat{\alpha}_i \cdots \beta_j \cdots \sigma_n\}(\sigma_1 \cdots \sigma_n) = \{\gamma_1 \cdots \hat{\alpha}_i \cdots \beta_j \cdots \sigma_n\}(\sigma_1 \cdots \sigma_n)\}^\oplus\).

Here the interrelation \(k_\theta/k_\Gamma = k_\theta/k_\Gamma = 2/2\) that indicates on unchangeability of parameters of the transformed PSTF \(Y^\oplus\), in which only inversion of different positions took place; \(k_Y = 1\).

- Let \(d = 3\). Then \(\theta_1 = (\alpha_1 \cdots \hat{\alpha}_i \cdots \hat{\beta}_j \cdots \cdots \cdots \sigma_n)\) and \(\theta_2 = (\sigma_1 \cdots \sigma_n)\), \(\theta_1 = (\alpha_1 \cdots \hat{\alpha}_i \cdots \hat{\beta}_j \cdots \cdots \cdots \sigma_n)\) and \(\theta_2 = (\sigma_1 \cdots \sigma_n)\), and \(\theta_1 = (\alpha_1 \cdots \hat{\alpha}_i \cdots \hat{\beta}_j \cdots \cdots \cdots \sigma_n)\) and \(\theta_2 = (\sigma_1 \cdots \cdots \cdots \cdots \cdots \sigma_n)\), \(\hat{\alpha}_i, \hat{\beta}_j, \hat{\gamma}_k \in \{0, 1\}\).

So, transformed PSTF \(Y^\oplus\) will look like:

\[
Y^\oplus = \{\gamma_1 \cdots \hat{\alpha}_i \cdots \hat{\beta}_j \cdots \cdots \cdots \sigma_n\}, \{\sigma_1 \cdots \sigma_n\}\}^\oplus =
\]
Example 1. To apply Theorems 1, 2 and 3 to the function that is the proof of Theorem 3.

\[ Y = \{ (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n) \} \]

\[ Y^\oplus = \{ (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n) \} \]

\[ Y^\oplus = \{ (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n) \} \]

\[ Y^\oplus = \{ (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n) \} \]

\[ Y^\oplus = \{ (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n), (\sigma_1 \cdots \tau \cdots \rho \cdots \sigma_n) \} \]

\[ \text{Solution.} \] The function has PSTF $Y^\oplus = \{ (1^{-1}), (1^{-1}), (1^{-1}), (1^{-1}) \}$. To the pair $(1111)$ and $(0000)$, that has $d = 4$, we will apply, for example, the fourth PSTF from the rule (5):

\[ Y^\oplus = \left\{ (1^{-1}), (1^{-1}), (1^{-1}), (1^{-1}), \begin{array}{c} (000-1) \\ 0-01 \\ 01-1 \\ -1-11 \end{array} \right\} \]

Applying the rule (6) of Theorem 2 to the underlined pairs that have $d = 1$, namely

\[ \begin{pmatrix} -1 \\ 01 \end{pmatrix} \oplus \begin{pmatrix} 11 \end{pmatrix}, \begin{pmatrix} -11 \\ -11 \end{pmatrix} \oplus \begin{pmatrix} -01 \end{pmatrix}, \begin{pmatrix} -011 \\ -11 \end{pmatrix} \oplus \begin{pmatrix} -010 \end{pmatrix} \]

we will obtain PSTF $Y^\oplus = \{ (1^{-1}), (1^{-1}), (1^{-1}), (1^{-1}), (000-1), (11-1), (00-1), (0-01) \}$. Doing further transformations and according to the rules (16) and (17) of
Theorem 3, namely \( \begin{pmatrix} 010 \\ 000 \end{pmatrix} \Rightarrow \begin{pmatrix} 00- \\ 100- \\ -011 \end{pmatrix} \), \( \begin{pmatrix} 00- \\ 0-00 \\ 110- \end{pmatrix} \), we’ll obtain the final minimal PSTF \( Y^\oplus = \{ (1-), (-1-), (-0-), (0-), (011), (0-00), (110-), (1-1) \}^\circ \Rightarrow \Rightarrow \{ (1-), (-1-), (-011), (0-00), (110-), (1-1) \}^\circ \). Here the cost of realization of the minimized function \( f = x_1 \oplus x_2 \oplus \bar{x_2}x_3x_4 \oplus \bar{x_1}x_3\bar{x}_4 \oplus x_1x_2x_3 \oplus x_1x_2x_4 \) is equal to \( k_n^s/k_s^t = 6/14 \) that is a better result if compared to \([32]\), where \( k_\theta /k_l = 6/15 \).

4 Minimization of Complete and Incomplete Functions

The proposed method of Boolean functions minimization in the polynomial set-theoretical format is based on the idea of minterms splitting of a given function \( f(x_1, x_2, \ldots, x_n) \) in the disjunctive format \([27–29]\).

The algorithm of minimization of a function \( f \) in the polynomial set-theoretical format is realized on two stages:

1. **1-st stage:** the procedure of splitting of minterms of a given function \( f \) and the obtaining of a set of covering of a matrix of splitting;

2. **2-nd stage:** the procedure of iterative simplification of conjuncterms of a set of covering (obtained on the 1-st stage) on the basis of generalized rules of Theorems 1, 2 and 3 and formation of a minimal PSTF \( Y^\oplus \) of a given function \( f \).

The 1-st stage is realized by sequence of such steps:

**Step 1:** the given binary minterms \( m_1, m_2, \ldots, m_k \) of the perfect PSTF \( Y^\oplus = \{ m_1, m_2, \ldots, m_k \}^\circ \) of the function \( f \) are split (operator \( \subseteq \)) by using the matrix-column of the masks of literals of rank \( r \geq n - \log_2 k \), \( r = 1, 2, \ldots, n \), as a result of this a matrix of splitting \( M_n^r \) of \( C_n^r \times k \) dimension is formed, where \( C_n^r = \frac{n!}{(n-r)!r!} \), for example, let \( n = 5 \); if the number \( k \) of minterms is \( 8 \leq k < 16 \), then we use the matrix of masks of rank \( r = 2 \), and as a result the matrix \( M_n^2 \) of the dimension \( C_2^2 \times k \) is formed;

**Step 2:** in the matrix \( M_n^r \) (in our example \( M_2^r \)) for execution of the procedure of covering (operator \( \supseteq \)) the conjuncterms-copies of \( r \)-rank, the number of which \( 2^{n-r-1} < k_r \leq 2^{n-r} \) (8 < \( k_r \leq 8 \)) are highlighted by underlining; priority is given to conjuncterms-copies and their number is \( k_r = 2^{n-r} \) (\( k_r = 8 \)); if \( k = k_r \), then the matrix is covered with a conjuncterms-copy of \( r \)-rank; if \( k_r < 2^{n-r} \) (\( k_r < 8 \)), then the covering of the matrix will be made of the conjuncterms-copies the number of which \( 2^{n-r-1} < k_r \leq 2^{n-r} \), and if there are not enough of them, then together with generating minterms of the matrix \( M_n^r \); if \( k_r < 2^{n-r-1} \), then the transition to step 1 is done for realization of similar procedures with application of the matrix of masks of the rank \( r = 3 \) and etc. until to getting in the covering of the matrix \( M_n^r \) of the minterms splitting of which provides its covering, if such minterms > 2, then the transition to step 1 is done.

The 1-st stage of algorithm is completed when there are not only minterms in the set of covering of the matrix \( M_n^r \) or when the split elements do not provide its covering.

The 2-nd stage of the minimization algorithm is the procedure of iterative simplification. It is done with the conjuncterms of the set of the covering in sequence of the following steps:
Step 1: for every pair with \( d = 1 \) (pairs with \( d = 0 \) are not taken into account) either the rule (2) of Theorem 1, or the rule (6) of Theorem 2; are applied; after respective replacement the transition to the 1-st is done, if there are not such pairs, then go to the step 2;

Step 2: for every pair with \( d = 2 \) we apply either one from the sets of the rule (3) of Theorem 1, or the rule (7) of Theorem 2; after respective replacement the transition to the 1-st step is done and if there are not such pairs, then go to the 3-rd step;

Step 3: for every pair with \( d = 3 \) we apply either one from the sets of the rule (4) of Theorem 1, or one from the sets of the rules (8), (9) or (10) of Theorem 2, or one from the rules (15), (16) or (17) of Theorem 3; after respective replacement the transition to the 1-st step is done and if there are not such pairs, then go to the 4-th step;

Step 4: for every pair with \( d = 4 \) we apply one from the sets of the rule (5) of Theorem 1, or one from the rules (8), (9) or (10) of Theorem 2, or one from the sets of the rules (18)–(23) of Theorem 3; after respective replacement the transition to the 1-st step is done and if there are not such pairs, then go to the 5-th step;

Step 5: if further transformation does not lead to the simplification of the set of conjuncterms, then this set is the found minimal PSTF \( Y^\oplus \) of the function \( f \), the cost of realization of which is determined by the interrelation \( k^*_b/k^*_i \).

Example 2. To minimize the function \( f(x_1, x_2, x_3, x_4) \) in the polynomial format by using the splitting method. This function has perfect STF \( Y^1 = \{0, 6, 14, 15\}^1 \) (this function is borrowed from [21, p. 28]).

Solution.

\[
Y^\oplus = \{(0000), (0110), (1110), (1111)\}^\oplus \Rightarrow \begin{bmatrix}
\text{ll} & \text{l-l} & \text{l-l} & \text{l-l} \\
0-0 & 0-1 & 1-0 & 1-1 \\
0-0 & 0-0 & 1-0 & 1-1 \\
-00 & -11 & -11 & -11 \\
-00 & -10 & -10 & -11
\end{bmatrix}
\]

\[
\Rightarrow \{\text{l-l}\} = \{((-11), (0111), (0000))\}^\oplus.
\]

We apply the rule (4) of Theorem 1 to the minterms \((0000)\) and \((0111)\):

\[
\begin{pmatrix}
0000 \\
0111
\end{pmatrix} \Rightarrow \begin{pmatrix}
000- & 001- & 011- & 010- & 011- \\
00-1 & 01-1 & 00-0 & 00-0 & 00-0 \\
011- & 00-0 & 00-0 & 01-1 & 01-0 \\
01-0 & 01-0 & 01-0 & 00-0 & 00-0
\end{pmatrix}
\]

After replacement of minterms \((0000)\) and \((0111)\) by the underlined sets in the set of covering, we obtain two equal as to the realization cost of solutions of minimization of the given function which is reflected by the minimal PSTF:

\[
Y^\oplus = \{((-11), (0000), (0111))\}^\oplus \Rightarrow \begin{pmatrix}
1. (00-0), (01-0) \\
2. (01-0), (00-0)
\end{pmatrix}^\oplus
\]

\[
\Rightarrow \{111-\}, \{1. (00-0), (01-0)\}^\oplus \Rightarrow \{111-\}, \{2. (01-0), (00-0)\}^\oplus.
\]
Answer. The cost of realization of the minimized function determines the interrelation \(k_2^9/k_1^9 = 3/9\) that is a better result than in [21], where the PSTF \(Y^\oplus = \{(-1-), (0000), (0111)\}^\oplus\) that is equal to 3/10.

In [27, 28], the incomplete function \(f(x_1, x_2, \ldots, x_n)\) can be given by the perfect STF \(\{Y^1, Y^\sim\}\), where \(Y^1\) and \(Y^\sim\) there are subsets of the complete set \(E_n\), on which the function \(f\) takes the value respectively 1 and \(\sim\) (so-called “don’t-care”). In the polynomial set-theoretical format the sets \(Y^\oplus\) and \(Y^\oplus\) correspond to the sets \(Y^1\) and \(Y^\sim\), the elements of which are binary minterms. Thus, incomplete function \(f\) can be given by the perfect PSTF \(\{Y^\oplus, Y^\oplus\}\).

Similarly, [27, 29] the procedure of splitting of conjuncters of incomplete function \(f\) is realized by the matrix of splitting \(M_n^r\), which is designated as \(M_n^r:M_n^r\), where \(M_n^r\) that is basic submatrix, \(M_n^r\) that is additional submatrix, and \(\sim\) is a symbol of separation of the matrix \(M_n^r\). As a result of covering of the matrix \(M_n^r\) a set of the splitting conjuncters \(Y^\oplus:Y^\oplus\) is be obtained.

An algorithm of minimization of an incomplete function in the polynomial set-theoretical format is realized in the same way as for a complete function in two stages. On the 1-st stage the minterms of the perfect PSTF \(\{Y^\oplus, Y^\oplus\}\) are split by using of the matrix \(M_n^r\), where the main role in its cover is played by the conjuncters-copies of the basic submatrix \(M_n^r\). Whereas the 2-nd stage of the algorithm of minimization of an incomplete function is realized in similar way as for a complete function.

**Example 3.** To minimize incomplete function \(f(x_1, x_2, x_3, x_4)\) in the polynomial format by using splitting method. This function is has perfect STF \(\{Y^1 = \{3, 5, 6, 9, 12, 15\}\}^1\)
\(\{Y^\sim = \{1, 2, 8, 11\}\}^\sim\) (this function is borrowed from [33, p. 460]).

**Solution.**

\[
\begin{align*}
Y^\oplus &= \{(0011), (0101), (0110), (1001), (1100), (1111)\}^\oplus \Rightarrow \\
Y^\oplus &= \{(0001), (0110), (1000), (1011)\}^\oplus \\
\end{align*}
\]

\[
\begin{bmatrix}
[l l \leftarrow] & [s] \\
\hline
l \leftarrow & 00\leftarrow & 01\leftarrow & 01\leftarrow & 10\leftarrow & 11\leftarrow & 11\leftarrow & 00\leftarrow & 00\leftarrow & 10\leftarrow & 10\leftarrow \\
\hline
l \leftarrow & 00\leftarrow & 01\leftarrow & 10\leftarrow & 11\leftarrow & 11\leftarrow & 11\leftarrow & 00\leftarrow & 00\leftarrow & 10\leftarrow & 10\leftarrow \\
\hline
\hline
\sim \leftarrow & 00\leftarrow & 00\leftarrow & 11\leftarrow & 11\leftarrow & 11\leftarrow & 11\leftarrow & 00\leftarrow & 00\leftarrow & 10\leftarrow & 10\leftarrow \\
\hline
\sim \leftarrow & 00\leftarrow & 00\leftarrow & 11\leftarrow & 11\leftarrow & 11\leftarrow & 11\leftarrow & 00\leftarrow & 00\leftarrow & 10\leftarrow & 10\leftarrow \\
\hline
\hline
\end{bmatrix}
\]

\[
\begin{align*}
[l \leftarrow] & \Rightarrow \{(01\sim), (0111\sim), (1101\sim), (0011\sim), (1001\sim), (1111\sim)\}^\oplus \\
& \Rightarrow \{(01\sim), (0111\sim), (1101\sim), (0011\sim), (1001\sim), (1111\sim)\}^\oplus \\
\end{align*}
\]

After the transformation of the pair of highlighted conjuncters by the rule (3) of the Theorem 1, i. e. \((01\sim)^\oplus \Rightarrow (-1-\sim)^\oplus\), we will obtain the final minimal PSTF \(Y^\oplus = \{(-1\sim), (1---\sim), (-1\sim)^\oplus\}\), to which corresponds the minimal PSTF
\( Y^{\oplus} = \{ (2, 3, 6, 7, 10, 11, 14, 15), (8, 9, 10, 11, 12, 13, 14, 15) \} \), where the highlighted in bold font elements belong to set \( Y^{-} \).

**Answer:** The cost of realization of the given function is equal to \( k_i^{\oplus}/k_i^{\ominus} = 3/4 \). If compared with [33] it is a better result, where \( Y^{\oplus} = \{ (11\ldots), (11\ldots), (11\ldots) \} \) and the cost of realization is equal to \( k_i^{\oplus}/k_i^{\ominus} = 3/5 \).

## 5 Minimization of System of Complete and Incomplete Functions

In general, the system \( F(X) = \{ x_1, x_2, \ldots, x_n \} \), of Boolean functions, \( f_i(X) \), \( i = 1, 2, \ldots, s \) can be represented in the polynomial set-theoretical format as a perfect PSTF \( \{ Y_i^{\oplus}, Y_i^{\ominus} \} \) [27, 29]:

\[
F(X) = \left\{ Y_1^{\oplus}, \ldots, Y_s^{\ominus} \right\}
\]

where \( v_i < 2^n - k_i; m_{ij} \) are binary minterms of functions \( f_i(X) \), \( j = 1, 2, \ldots, k_i \); while if \( F(X) \) is a system of the completely specified functions, then \( Y_i^{\oplus} = \emptyset \) and we have perfect PSTF \( \{ Y_i^{\ominus} \} \), and if \( F(X) \) is a system of the incompletely specified functions, then \( Y_i^{\ominus} = Y_i^{\ominus} \) and we have perfect PSTF, where symbol \( \sim \) represents incomplete (“don’t care”) values of functions \( f_i \) of system \( F(X) \).

Similarly as in SOP form [27, 29] compatible minimization of the system of PSTF \( \{ Y_i^{\oplus}, Y_i^{\ominus} \} \) (25) is performed by splitting method with the system minterms \((m)_{1,2,\ldots,s'} \in \{1, 2, \ldots, s\}\), formed from minterms of the system \( F(X) \).

The algorithm of compatible minimization of the system \( F(X) \) of complete functions given by the perfect PSTF \( \{ Y_i^{\oplus}, Y_i^{\ominus} \} \), \( Y_i^{\ominus} = \emptyset \), is realized in the following way. On the first stage the system minterms \((m)_{1,2,\ldots,s'} \) of the set \( \{ Y_i^{\oplus} \} \), \( i = 1, 2, \ldots, s \), are split by using the matrix \( M_i^{\ominus} \) creating a system conjuncters of \((n-1)\)-rank \((\theta_{i}^{-1})_{1,2,\ldots,s'}\). The minimal covering of the matrix \( M_i^{\ominus} \) is done in similar way [27, 29] by the identical system conjuncters-copies of \((n-1)\)-rank. But among them, a decisive role for realization of compatible minimization of the given system will be played by those ones the indices of which contain the greatest quantity of numbers with the set \( \{ 1, 2, \ldots, s \} \). Here, if \( (\theta_{i}^{-1})_{1,2,\ldots,s'} \) and \((\theta_{j}^{-1})_{1,2,\ldots,s'} \), \( s' \in \{1, 2, \ldots, s\} \), are identical system conjuncters-copies of \((r-1)\)-rank of the matrix \( M_j^{\ominus} \), \( r = 1, 2, \ldots, n-1 \), then they can be elements of its covering if the indices of their generative form \((\theta_{i}^{-1})_{1,2,\ldots,s'} \) and \((\theta_{j}^{-1})_{1,2,\ldots,s'} \) will form a not empty intersection, i.e. \( \{ 1, 2, \ldots, s' \} \cap \{ 1, 2, \ldots, s'' \} \neq \emptyset \). For example, let the system minterms \((100)_{1,2,4}, (110)_{1,3}, (001)_{1,2,3} \) be generative elements of the matrix \( M_i^{\ominus} \). For the mask \( \{t-l\} \) the identical system conjuncters-copies will be \((1-0)_{1} \) and \((1-0)_{1} \), the index of which determines the intersection \( \{ 1, 2 \} \cap \{ 1, 3 \} = \{ 1 \} \), and for the mask \( \{t-l\} \) will be \((1-0)_{1,3} \) and \((-10)_{1,3} \) because \( \{ 1, 3 \} \cap \{ 1, 2, 3 \} = \{ 1, 3 \} \). So, in this case for covering of the matrix \( M_i^{\ominus} \) it is recommended to choose the pair of the mask \( \{-L\} \) because the power \( |\{ 1, 3 \}| > |\{ 1 \}| \).
Example 4. In the polynomial set-theoretical format to minimize the system \( F(X) \) of complete functions \( f_i(x_1, x_2, x_3), \ i = 1, 2, 3, \) by the splitting method. This has perfect STF \( Y_1^\oplus = \{(000), (010), (101), (110)\} \)

\( Y_2^\oplus = \{(001), (011), (101)\} \).

This example is borrowed from [21, p. 35] where the author illustrates efficiency of xlinking method.

Solution. Having transformed the given system of the perfect STF \( Y_{1,2,3}^\oplus \) into the system of the perfect PSTF \( Y_{1,2,3}^{\oplus \ominus} \) and, having formed from it a set of system minterms, we will execute the splitting procedure by using the matrix \( M_1 \) and its covering procedure:

\[
Y_{1,2,3}^{\oplus \ominus} = \{(000)_{1,3}, (001)_{2,3}, (010)_{1,3}, (011)_{2,3}, (101)_{1,2}, (110)_{1}\} \quad \Rightarrow \quad \frac{H-}{l-l} = \frac{0-1, 0-2, 0-1, 3 \ 0-2, 3 \ 1-1, 2 \ 1-1}{-0 \ -0 \ -1 \ -1 \ -0 \ -1 \ -1 \ -0} \quad \Rightarrow \quad \frac{C}{\{(0-\), (00)_{1,3}, (011)_{1,3}, (0-2)_{1,3}, (000)_{2,3}, (010)_{2,3}, (101)_{1,2}, (110)_{1}\}\oplus}.
\]

We will do the splitting procedure with system minterms of the formed set by using the matrix \( M_2 \) and its covering procedure:

\[
(00-1, 3, 011)_{1,3}, (000)_{2,3}, (010)_{2,3}, (101)_{1,2}, (110)_{1} \quad \Rightarrow \quad \frac{H-}{l-l} = \frac{00- \ 01- \ 00- \ 01- \ 10- \ 11-}{0-1, 3 \ 0-1, 3 \ 0-0, 3 \ 1-1 \ 1-0}{-01 \ -11 \ -00 \ -10 \ -01 \ -10} \quad \Rightarrow \quad \frac{C}{\{(0-1, 3, (00)_{2,3}, (101)_{1,2}, (110)_{1}\)\oplus}.
\]

Having distributed system conjuncters in the functions we obtain the system of the PSTF \( Y_{1,2,3}^{\oplus \ominus} \), with the underlined elements of which we will do step by step the transformations according to the rules (2), (3) and (7) of the theorems:

\[
Y_1^{\oplus \ominus} = \{(0-\), (011)_{1,3}, (101)_{1,3}\}\oplus = \{(0-\), (1)_{1,3}, (11)_{1,3}\}\oplus = \{(0-\), (01)_{1,3}, (11)_{1,3}\}\oplus.
\]

\[
Y_2^{\oplus \ominus} = \{(0-\), (00)_{2,3}, (010)_{2,3}\}\oplus = \{(0-\), (00)_{2,3}, (101)_{2,3}\}\oplus = \{(0-\), (101)_{2,3}\}\oplus = \{(01-), (111)\}\oplus .
\]

\[
Y_3^{\oplus \ominus} = \{(0-\), (00)_{1,2}, (101)_{1,2}\}\oplus = \{(0-\), (0-\), (0-\), (11)_{1,2}, (101)_{1,2}\}\oplus.
\]

Answer. Minimal system of the PSTF \( Y_{1,2,3}^{\oplus \ominus} \): \( Y_1^{\oplus \ominus} = \{(0-\), (0-\), (11)_{1,2}, (11)_{1,2}\}\oplus \). Cost of its realization reflects the interrelation \( k_1^*/k_1^* = 4/7 \). If compared with [21] it is a better result, where this system is compatibly minimized by xlinking method with \( k_1^*/k_1^* = 4/9 \), namely \( Y_2^{\oplus \ominus} = \{(0-\), (0-\), (0-\), (11)_{1,2}, (11)_{1,2}\}\oplus \).

\[
\begin{align*}
Y_1^{\oplus \ominus} = \{(0-\), (0-\), (11)_{1,2}, (11)_{1,2}\}\oplus \\
Y_2^{\oplus \ominus} = \{(0-\), (0-\), (0-\), (11)_{1,2}, (11)_{1,2}\}\oplus \\
Y_3^{\oplus \ominus} = \{(0-\), (0-\), (0-\), (11)_{1,2}, (11)_{1,2}\}\oplus.
\end{align*}
\]

In case of the system of incomplete functions (25) the set of system minterms will consist of two subsets separated by the symbol \( \oplus \) and reflected as \( \{Y_I^{\oplus \ominus}, Y_J^{\oplus \ominus}\} \).
Here the system minterms undergo the splitting procedure by using the matrix $M^*_n$, the elements of which are the conjuncters of $r$-rank. It should be noted, that in the course of covering the matrix $M^*_n$ two procedures are realized at a time: making the matrix compatible as its elements are used to maximum extent with higher capacity of the set $I$, and making it more predetermined in which the elements of the submatrix $Y^*_I$ are used. After distribution of the last ones in the functions of the system we obtain $\{Y^*_I, Y^*_I\}$, the elements of which for every function further undergo the simplification procedure according to the rules of the respective theorems in Section 3, selecting out of possible variants of transformation those which will provide the compatible minimization of the given system $F(X)$ in the best way.

**Example 5.** [37, p. 228, example 5.1] To minimize the system $F(X)$ of incomplete functions $f_1(a, b, c)$ and $f_2(a, b, c)$, given by the truth table (see right) with the help of splitting method in the polynomial set-theoretical format.

<table>
<thead>
<tr>
<th>$I$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$f_1$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>~</td>
<td>~</td>
</tr>
</tbody>
</table>

Solution. The given system $F(X)$ has the perfect PSTF

$\left\{ \begin{array}{ll} Y_1^0 = \{(000), (011), (110)\}; Y_1^1 = \{(001), (100), (101)\} \\ Y_2^0 = \{(010), (111)\}; Y_2^1 = \{(001), (101), (100)\} \end{array} \right.$

We will form a set of system minterms (i.e. $\{Y^i, Y^j\}$), with which we will do the splitting procedure by using the matrix $M^3_3$ and the procedure of its covering, for example, for the mask $\{-l\}$:

$Y^*_{1,2} = \{(000)_1, (010)_2, (011)_1, (100)_2, (101)_1\}$

$\Rightarrow \exists \{-l\} = \{(000)_1, (111)_1, (100)_2, (101)_1\}$

After removal of the system minterm $(011)_2$ the set of covering will look like

$Y^*_{1,2} = \{(000)_1, (110)_2, (010)_1, (111)_1, (001)_2, (011)_2, (100)_1, (101)_1\}$

Having distributed the system conjuncters on the functions, we obtain the system

$\left\{ \begin{array}{ll} Y_1^0 : Y_1^1 = \{(-1), (000), (010), (111)\}; (001), (100), (101) \end{array} \right.$

$\left\{ \begin{array}{ll} Y_2^0 : Y_2^1 = \{-1\}; (010), (111) \end{array} \right.$

We will do the splitting procedure with the minterms of the PSTF of the function $f_1$ by using the matrix $M^3_3$ and the procedure of its covering:

$\left\{ \begin{array}{ll} (000), (010), (111) ; (001), (100), (101) \end{array} \right.$

$\Rightarrow \exists \{(000), (010), (111)\}; (001), (100), (101) \}$

$\Rightarrow \exists \{(000), (010), (011)\}; (001), (100), (101) \}$

$\Rightarrow \exists \{(000), (010), (011)\}; (001), (100), (101) \}$

$\Rightarrow \exists \{(000), (010), (011)\}; (001), (100), (101) \}$

$\Rightarrow \exists \{(000), (010), (011)\}; (001), (100), (101) \}$
Having taken into account the rule (3) \( (0--) \oplus (1--) = (1--) \oplus (0--) \) we will obtain two solutions of the minimal PSTF \( Y^\oplus_1 = \{(-1-), (1--), (-01)\} \oplus \). We will do similar procedures for the minterms of the PSTF of the function \( f_2 \) by applying the matrix \( M^l_2 \) for their splitting:

\[
\{(110);(001), (011), (100)\} \oplus S \Rightarrow \begin{bmatrix}
11- & 00- & 01- & 10- \\
1-0 & 0-1 & 0-1 & 1-0 \\
-10 & -01 & -11 & -00
\end{bmatrix} \Rightarrow \{(1-0), (0-0)\} \oplus.
\]

After the transformation according to the rule (3) \( (1--) \oplus (0-0) \Rightarrow \{(1--), (001)\} \oplus \), we obtain two solutions of the minimal PSTF \( Y^\oplus_2 = \{(-1-), (1--), (-01)\} \oplus \). The given system of functions has two solutions of minimization which reflect the PSTF

\[
\begin{align*}
Y^\oplus_1 &= \{(-1-), (-1-), (0--)\} \oplus; \\
Y^\oplus_2 &= \{(-1-), (-1-), (1--)\} \oplus.
\end{align*}
\]

The analytical expressions correspond to these solutions:

\[
\begin{align*}
1. \quad & f_1(a, b, c) = b \oplus c \oplus \bar{a}; \\
& f_2(a, b, c) = b \oplus c \oplus a; \\
2. \quad & f_1(a, b, c) = b \oplus \bar{c} \oplus a \\
& f_2(a, b, c) = b \oplus \bar{c} \oplus \bar{a}.
\end{align*}
\]

Cost of realization of the system for the both solutions is equal to \( k^h_0/k^l_1 = 4/4 \). If compared with [37] it is a better result, where cost of realization is equal to \( k^h_0/k^l_1 = 4/7 \), namely:

\[
\begin{align*}
& f_1(a, b, c) = b \oplus \bar{c} \oplus ab \\
& f_2(a, b, c) = b \oplus ab\bar{c}.
\end{align*}
\]

6 Conclusions

A new minimization method in the polynomial set-theoretical format of complete and incomplete logic functions with \( n \) variables and their system has been presented. It consists in the splitting procedure of given minterms and iterative simplification of conjuncterms on the based set-theoretical rules. The method’s efficiency has been proved by numerous examples borrowed from well-known publications (see References) related to different minimization methods. The vast majority of functions and their systems minimized by the proposed method showed better results. This is due to the fact that in the process of transformation are involved the conjuncterms with Hamming distance \( d \geq 3 \), the transformed PSTF of which may have elements for which in the given set there will be a pair with smaller \( d \). The search procedure of such elements has a combinative character: after each replacement of a chosen pair of conjuncterms of the given PSTF \( Y^\oplus \) for certain set of the transformed PSTF \( Y^\oplus \) we obtain a new set where it is necessary to determine distance \( d \) between new pairs and, having chosen from them the elements with minimal \( d \), to apply the rules of respective theorem and build again
a new set and so on. As a result, the probability of effective simplification of conjunctive terms set increases through the use of appropriate transformation rules that reduce the implementation cost of realization of minimized function.

References

The Handling of Missing Values in Medical Domains with Respect to Pattern Mining Algorithms

Danilo Schmidt\(^2\), Matthias Niemann\(^1\), and Gabriela Lindemann-von Trzebiatowski\(^3\)

1 Department of Transfusion Medicine, University Hospital Charite
   matthias.niemann@charite.de
2 Department of Nephrology, University Hospital Charite
   danilo.schmidt@charite.de
3 Department of Governing Bodies, Humboldt University of Berlin
   gabriela.lindemann@uv.hu-berlin.de

Abstract. Missing values are a wide spread problem in analyzing large data sets. In the medical domain they are unavoidable and complete analyzing methods fail here. In the paper we give an overview of kinds of missingness and common methods to handle missing values in machine learning algorithms. We introduce the Charité Query Language Toolkit which was developed to find out similar patterns in patient data records with respect to post-kidney-transplant patients. The toolkit uses available case analysis methods combined with a preprocessing of missing values as a compromise of simplicity and functionality.

Key words: data mining, medical data, missing values

1 Introduction and Background

Missing values are a wide spread problem for analyzing methods, such as machine learning, pattern recognition or data-mining algorithms, in many domains. For medical data sets missing values are unfortunately unavoidable. In a complete case analysis for these data sets all patient records with missing data would excluded. Performing clinical studies only with complete patient data sets lead to a significantly smaller sample size with reduced statistical expressiveness.

Depending of the choosen method for the statistical analysis missing values can restrict the cohort so much that the whole study is endangered.

In the last decades the amount of electronically collected patient data has grown rapidly and the demand of researchers and physicians for the development of analyzing methods and tools for data-sets with missing values is obvious.

In our work, we will describe the different kinds of missing values and follow here in principle the systematic of Pigott \cite{pigott2003} and de Goeij et.al. \cite{de2005}. We give an impression how to deal with missing values by example of pattern mining algorithms and introduce some useful preprocessing methods for medical data. At last we present a short example for including these methods in our frequent pattern mining toolkit.
2 Kinds of Missingness

Missing values are a common issue when analyzing data in a wide range of research fields. In the medical domain it seems unavoidable, especially in long-term treatments. De Goeij et al. [1] define a missing value as "hiding the value of an attribute". While analyzing a dataset a missing value occurs when the specific value is not available. This does not necessarily mean that the value does not exist, but it is unknown. The missing value may be one of the attribute values (e.g. a categorical value) or a unique value (e.g. a numerical value). To quantify missingness, a ratio of missing values and all values can formed over all attributes by a simple formula:

\[
\text{missingness}(B) = \frac{|B_{\text{missing}}|}{|B|}
\]

There are several reasons for the missingness of values in medical data-sets. Depending on the ratio missingness(B) and the applied analyzing method, missing values may distort the final result and the underlying data missing mechanism may cause a biased statistical analysis. Therefore it is appropriate to spend some considerations into the kind of missingness of the values of a special data-set before choosing an adequate analyzing algorithm. With respect to the reasons of missingness there are distinguished three categories MCAR, MAR and NMAR.

MCAR - missing completely at random - is the strongest assumption for missing values of a dataset. The missing of a value neither depends on the observed parameters nor on the unknown value itself. MCAR will not bias the analysis of data, because the missing data has the same distribution as the available data. In medical domain it occurs e.g. if it was forgotten to induce an examination or there were problems in the transmission of laboratory data.

A less strict mechanism is MAR - missing at random. The missing of a value is allowed to be dependent on the observed parameters but not on the missing value itself. In long-term treatment of a patient it happens e.g. if the patient was not motivated to come to a medical control round because he had no health problems.

When adjusting a set of variables, MAR can be avoided by selecting highly correlated variables to be observed.

But clearly, this requires a specialist with domain-specific knowledge. To the third category of missingness belong data where values are not missing at random - NMAR. Here the missing of a value depends on the value itself. E.g. a creatinine value of a kidney transplanted patient was not measured because of rejection and loss of the transplant.

Without evaluating the dataset, no type of missing value can be ruled out. Even worse, each type of missingness can occur in a single dataset where different mechanisms overlay as our instances above show. Furthermore, it is not trivial to find out what kind of missingness applies.
3 Preprocessing Missing Values for Application in Pattern Mining Algorithms

There are several ways to handle missing values in pattern mining and data-analysis algorithms. Especially in the medical domain there are several studies on different approaches of dealing with missing values. Van der Heijden et al. [3] and Marlin [4] developed methods for handling missing values for several machine learning techniques. The most common methods are:

1. Complete Case Analysis (only for remaining complete rows),
2. Available Case Analysis (complete rows for current pattern),
3. Single Unconditional Mean Imputation (impute column’s mean),
4. Single Conditional Mean Imputation (impute mean based on conditional columns),
5. Multiple Imputation (regression model generates complete sets),
6. Maximum Likelihood (estimate underlying distributions),
7. Pattern-Mixture Model (user defined patterns of missingness).

All methods are suitable for MCAR data-sets. Furthermore, Single Conditional Mean Imputation, Multiple Imputation, Maximum Likelihood and Pattern-Mixture Model are additional usable for MAR data-sets, but only the last one is practicable for NMAR data-sets. In the following we will introduce in short the handling of missing values for the mentioned methods.

3.1 Complete Case Analysis

The easiest way to handle missing values is to delete all cases (rows) that contain missing values. The remaining data will be complete and all methods requiring complete data sets can be applied without further issues. When applying Complete Case analysis (CC) to table 1, the patients 2, 3 and 4 would be discarded. In subsequent operations, only patient 1 would be considered. If the missing values were not missing completely at random, Complete Case analysis will be biased. Unfortunately MCAR applies rarely (see table 2).

Furthermore, the deletion of many cases is not applicable if there are missing values in almost every case. The loss of information would increase while the significance decreases [3].

<table>
<thead>
<tr>
<th>Patient</th>
<th>Test A</th>
<th>Test B</th>
<th>Test C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>positive</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>3.0</td>
<td>negative</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>negative</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td>?</td>
<td>negative</td>
</tr>
</tbody>
</table>

Table 1. Data set with missing values
3.2 Available Case Analysis

Available Case analysis (AC), which is sometimes referred to as pair wise deletion, is less strictly than Complete Case analysis. When analyzing a subset of the observed variables, all complete cases for that subset are viewed. That means only missing values are ignored [3]. Considering tests B and C of table 1, patients 2 and 4 are discarded because for the given set of tests only patients 1 and 3 are complete. This method is rather easily applicable but has some drawbacks as mentioned by [3]. Because of the varying number of observations, errors in estimated covariance matrices might occur. Furthermore, only if missing values are MCAR, the estimates are consistent. It has been shown that Available Case analysis is superior to complete case analysis for weakly correlated variables. For strong correlations, AC is inferior to CC.

Table 2. As A2 is only done if A1 is positive, the missing of A2 is MAR

<table>
<thead>
<tr>
<th>Patient</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>negative</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>negative</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>positive</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>negative</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>positive</td>
<td>positive</td>
</tr>
<tr>
<td>6</td>
<td>positive</td>
<td>positive</td>
</tr>
</tbody>
</table>

3.3 Single Unconditional Mean Imputation

A contrary approach to CC and AC is imputation. Generally spoken, missing values will be filled up (imputed) with calculated values. After that procedure the data can be handled like a complete set. There are different methods of imputing values, which will be introduced briefly in the following.

The Single Unconditional Mean Imputation (sometimes referred to as single value imputation) replaces all missing of an observed variable by the mean of the available values of that variable [3]. In table 1, the mean of test A is 2.0. Hence the missing values of patients 3 and 4 would be replaced with 2.0.

There are several drawbacks of that method: The variance of the imputed variables decreases while the precision is overrated. The results of an unconditional mean imputation will always be biased. This method is rather easily applicable as there is no further information about the dataset required. Adapting this approach to categorical data, the most frequent value of a column is imputed. In order to avoid precision overrating, the unconditional imputation may be extended to analyze columns in order to find the underlying random distribution. The imputation is then based on the columns distribution.
3.4 Single Conditional Mean Imputation

An improvement to unconditional imputation is the conditional imputation method. By linear regression on the conditional (observed) variables with complete data, the missing values are imputed. When considering test B as the condition for test A in table 1, the mean of column A is calculated for all patients where test B is negative (mean = 3.0) and once more for all patients where test B is positive (mean = 1.0). Hence the imputed value for test A of patient 3 would be 3.0. The selection of conditional variables is not trivial. When selecting too many columns, the imputed value may be over fitted.

3.5 Multiple Imputation

In contrast to the previously mentioned single imputation methods, Multiple Imputation (MI) does not calculate a single mean of an observed variable in order to impute a missing value, but creates a set of possible complete data sets. Each imputed parameter is selected by the columns underlying random distribution that was determined by regression. On each complete set the analysis is done. Finally, the results will be brought together. Practical tests show that MI often performs better than CC and AC, especially in the field of nephrology. This holds for MCAR and MAR data as long as the model specification is suitable. An overrated precision is avoided by imputing data several times, while biasing is avoided by applying regression.

3.6 Maximum Likelihood

Maximum Likelihood (ML) methods estimate the parameters of the underlying distributions of the observed variables. To get the most probable parameters, an EM algorithm can be used. If the algorithm converges, the coefficients with the highest likelihood can be used in linear regression models [3]. In contrast to imputation methods, there are no estimated values filled into the gaps of the data set. Instead, ML methods can help to provide significant estimates for regression models. ML is unbiased for data that is MCAR or MAR and outperforms CC, AC and single imputations methods. But a proper statistical model is fundamental.

3.7 Pattern-Mixture Model

For pattern-mixture models the missing data mechanism may remain unknown. Instead, a mixture of different patterns describes the missingness in the data set, whereas each pattern describes a subset of the missing values. These patterns support the statistical model and can therefore improve the analysis. Hence pattern mixture models can produce good estimates for data that is MCAR, MAR and NMAR. Unfortunately, creating patterns requires a lot of domain specific knowledge about the data.

4 The Charité Query Language Toolkit

The Charité Query Language Toolkit was developed to find out similar patterns in patient data records with respect to post-kidney-transplant patients. Physicians should be
enabled to find out similar courses of diseases and treatments to infer from it for actual cases.

For the development of a toolkit, it might be disappointing that there is no simple general purpose method that handles all missing values in each imaginable query, especially if the data missing mechanism is unknown. Calders et.al. [5] summarizes the common methods and proposes to use different approaches in order to estimate the robustness. Applying multiple methods handling missing values might be confusing for future users of the software, so a compromise has to be found. The toolkit focuses on an easy usage. The user is not expected to provide additional information about the data. For that reason the user cannot be asked for selecting variables to impute the data. Model based approaches introduce better estimations at the cost of higher complexity and therefore have to be avoided too. Since there are different data sources, lots of missing values can be expected. In complex queries, complete case analysis can lead to a drop of all transactions. Even in simple queries the missingness may be very high. That is why complete case analysis has to be avoided either.

The toolkit uses available case analysis methods combined with a preprocessing of missing values as a compromise of simplicity and functionality. It does not focus on creating statistically faultless results. Biased correlations caused by violations of the MCAR-property can be expected and are accepted for that purpose.

When searching for new correlations, the user may not be interested in strong and therefore possibly known rules but in weaker or overlooked associations.

In the design phase of the toolkit, two essential settings have to be done. Firstly, the definition of time slices in order to discretize the time axis is required (see figure 1). In the second step, the definition of norms in order to provide a discretization of the parameters is necessary.

Norm values of parameters can be set in the norms tab (see figure 2). A group of norms for the same parameter was named norm family. This classification is necessary in order to recognize missing values. The norm type depends on the detected parameter as some tests generate qualitative (e.g. HCV, CMV, etc.) and some result in quantitative values (e.g. heart rate, creatinine, etc.). Qualitative norms are simply a mapping of a string value to the norm’s name.

This allows to assign several values to the same category (e.g. weak and positive are both mapped to not negative). Quantitative norms are ranges for numeric parameters that are mapped to the name of a norm (e.g. a creatinine value of 1.2 to 6.0 is mapped to bad).

Norms can be created automatically as well. The database contains references for several parameters that may be loaded. Furthermore, a function calculating quartiles and creating norms by these is available.

5 Discussion and Conclusion

In our paper we give a survey of kinds of missing values and common methods to handle them in pattern mining algorithms. We introduce the Charité Query Language Toolkit which was configured to work on post-kidney-transplant patient data. Since the data does not differ from other medical domains, the toolkit may be used in other
Fig. 1. time slices

Fig. 2. norms
departments as well. Either a separate database is provided or the data is loaded into the current database. Depending on the domain, individual preprocessing plug-ins might be necessary in order to provide proper data transformation abilities.

References

Petri Net Models of Simple Rule-Based Systems for Programming Physarum Machines

Extended Abstract

Andrew Schumann$^1$ and Krzysztof Pancerz$^{1,2}$

$^1$ University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszow, Poland
aschumann@wsiz.rzeszow.pl
$^2$ University of Management and Administration
Akademicka Str. 4, 22-400 Zamosc, Poland
kpancerz@wszia.edu.pl

Abstract. In the paper, we show that biological substrate in the form of Physarum polycephalum can be used to simulate simple rule-based systems. To extort a proper behavior from the substrate, appropriate distribution of stimuli (attractants and/or repellents) is required. To model behavior of the substrate and then program Physarum machine (a biological computing device experimentally implemented in the plasmodium of Physarum polycephalum), we propose to use Petri net models that can be treated as a high-level description. Petri net models enable us to reflect propagation of protoplasmic veins of the plasmodium in consecutive time instants (step by step).

Key words: Physarum polycephalum, unconventional computing, Petri nets, rule-based systems

1 Models of Simple Rule-Based Systems

There are various knowledge representation methods (cf. [4]) that have been developed to make real-world knowledge suitable for being processed by computers. One of the most popular knowledge representation systems are the rule-based ones. Rules can be easily interpreted by humans. Formally, rules can be presented in the framework of propositional logics. Propositional logics is concerned with the study of propositions, whether they are true or false. Propositions are formed by other propositions with the use of logical connectives. A production rule in rule-based systems is a rule which describes the relation between two propositions $d_i$ and $d_j$, i.e., a production rule points out to us an antecedent-consequence relationship from proposition $d_i$ to proposition $d_j$, where $d_i \neq d_j$. The general formulation of a production rule has the form IF $d_i$, THEN $d_j$, where $d_i$ and $d_j$ are propositions that can be evaluated as true or false with respect to any circumstance. If the antecedent part or consequence part of a production rule contains AND or OR connectives, then it is called a composite production rule. Four types of the composite production rules can be distinguished [8]:

- Type 1: IF $d_{i_1}$ AND $d_{i_2}$ AND ... AND $d_{i_k}$, THEN $d_j$. 

Type 2: IF $d_i$, THEN $d_{j_1}$ AND $d_{j_2}$ AND \ldots AND $d_{j_k}$.

Type 3: IF $d_{i_1}$ OR $d_{i_1}$ OR \ldots OR $d_{i_k}$, THEN $d_j$.

Type 4: IF $d_i$, THEN $d_{j_1}$ OR $d_{j_2}$ OR \ldots OR $d_{j_k}$.

Further, we will take into consideration types of rules 1 and 3, only.

Unconventional computing becomes an interdisciplinary field of science, where computer scientists, physicists and mathematicians apply principles of information processing in natural systems to design novel computing devices and architectures. In Physarum Chip Project: Growing Computers from Slime Mould [2] supported by FP7, we are going to implement programmable amorphous biological computers in plasmodium of Physarum polycephalum. Physarum polycephalum is a one-cell organism manifesting some primitive intelligence in its propagating and foraging behavior (cf. [9]). A biological computing device implemented in the plasmodium of Physarum polycephalum is said to be a Physarum machine. A comprehensive information on Physarum machines can be found in [1]. The Physarum machine comprises an amorphous yellowish mass with networks of protoplasmic veins, programmed by spatial configurations of attracting and repelling stimuli.

To program Physarum machines, i.e., to set the spatial distribution of stimuli, we are designing a new object-oriented programming language [10], [11], [13], called the Physarum language. Moreover, to support research on programming Physarum machines, we are developing a specialized software tool, called the Physarum software system, shortly PhysarumSoft (see [16]). Our language is based on the prototype-based approach (cf. [6]). According to this approach, there are inbuilt sets of prototypes, implemented in the language, that correspond to both the high-level models used for describing behaviour of Physarum polycephalum (e.g., ladder diagrams, transition systems, timed transition systems, Petri nets) and the low-level model (distribution of stimuli). In [15], we proposed to use Petri nets with inhibitor arcs (cf. [3]) as one of the high-level models to describe behaviour of Physarum polycephalum. The inhibitor arcs test the absence of tokens in a place and they can be used to disable transitions. This fact can model repellents in Physarum machines. A transition can only fire if all its places connected through inhibitor arcs are empty (cf. [20]). Each high-level model (including a Petri net one) is translated into the low-level language, i.e., spatial distribution of stimuli (attractants and/or repellents). Such distribution can be treated as a program for the Physarum machine.

In the literature, one can find a lot of approaches using Petri nets as models of rule-based systems (e.g., [5], [7], [18], [19]). First of all, structures of Petri nets reflect structures of rule-based systems. Various options of structures have been considered, according to respective approaches. Moreover, the proposed approaches differ in the dynamics that models reasoning processes. In our research, we propose another approach in order to reflect dynamics of Physarum machines, i.e., propagation of protoplasmic veins of the plasmodium according to activation/deactivation of stimuli. In the proposed Petri net models of Physarum machines, we can distinguish several kinds of places:

- Places representing Physarum polycephalum.
- Places representing control stimuli (attractants or repellents) corresponding to propositions in antecedent parts of rules.
Places representing auxiliary stimuli (attractants) corresponding to partial results of evaluation of logical expressions in antecedent parts of composite production rules.

Places representing output stimuli (attractants) corresponding to propositions in consequence parts of rules.

For each kind of places, we adopt different meaning (interpretation) of tokens (see, for example, Tables 1, 2 and 3 for places representing control stimuli and places representing output stimuli, respectively). Each token corresponds to proper evaluation of the proposition according to the role played by a given stimulus. In our models of simple rule-based systems, we have implemented an idea of flowing power used in ladder diagrams to model digital circuits. The same idea was used by us to construct logic gates through the proper geometrical distribution of stimuli in Physarum machines (see [17]). Flowing power is replaced with propagation of plasmodium of

<table>
<thead>
<tr>
<th>Token</th>
<th>Meaning</th>
<th>Evaluation of proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>Stimulus activated true</td>
<td></td>
</tr>
<tr>
<td>Absent</td>
<td>Stimulus deactivated false</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Token</th>
<th>Meaning</th>
<th>Partial evaluation of expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>Stimulus occupied by plasmodium</td>
<td>true</td>
</tr>
<tr>
<td>Absent</td>
<td>Stimulus not occupied by plasmodium</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Token</th>
<th>Meaning</th>
<th>Evaluation of proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>Stimulus occupied by plasmodium</td>
<td>true</td>
</tr>
<tr>
<td>Absent</td>
<td>Stimulus not occupied by plasmodium</td>
<td>false</td>
</tr>
</tbody>
</table>
Physarum polycephalum. Therefore, in each Petri net model of a rule, a place representing Physarum polycephalum is present. Petri net models are a useful tool to reflect dynamics of Physarum machines, i.e., propagation of protoplasmic veins of the plasmodium in consecutive time instants. Tokens present in places representing output stimuli show which attractants of Physarum machines are occupied by the plasmodium at given time instants.

In general, we can distinguish two techniques to control behavior of Physarum polycephalum: repellent-based and attractant-based [1]. Attractants are sources of nutrients or pheromones, on which the plasmodium feeds. In case of repellents, the fact that plasmodium of Physarum avoids light and some thermo- and salt-based conditions is used. These possibilities are reflected in the created Petri net models. Technically, the second approach is easier to implement. In case of repellent-based control approach, Petri net models of production rules of type 1 and 3 have the form as in Figures 1 and 2, respectively. In these models, the places $R_{d1}$, $R_{d2}$, ..., $R_{d3}$ correspond to propositions in the antecedent parts of the rules. The relationship between meaning of tokens and evaluation of propositions is shown in Table 1. These places are translated into repellents in the low-level model (distribution of stimuli). The places $A_1$, $A_2$, ..., $A_{k-1}$ correspond to auxiliary stimuli. The relationship between the meaning of tokens and the evaluation of propositions is shown in Table 2. The place $A_{dj}$ correspond to the output stimulus.

---

**Fig. 1.** A Petri net model of a rule of type 1: the repellent-based control approach

**Fig. 2.** A Petri net model of a rule of type 3: the repellent-based control approach
The relationship between meaning of tokens and evaluation of propositions is shown in Table 3. It is easy to see that, in case of type 1 of production rules, the token is present in $A_{dj}$ (the proposition in the consequence part is true), if all places $R_{di1}$, $R_{di2}$, ..., $R_{dil}$ do not hold tokens (the propositions in the antecedent part are true). In case of type 3 of production rules, the token is present in $A_{dj}$ (the proposition in the consequence part is true), if at least one of the places $R_{di1}$, $R_{di2}$, ..., $R_{dil}$ does not hold a token (at least one of the propositions in the antecedent part is true). The structures of Physarum machines for production rules of type 1 and 3 are shown in Figure 3 (a) and (b), respectively. Distributions of stimuli can be treated as programs for these machines. In the further research, we will consider more complex rule-based systems. However, we are aware of the topological constraints if the Physarum machine is implemented in the two-dimensional space (e.g., on the Petri dish). In this case, propagation of protoplasmic veins forming a planar graph is admissible only.

Another challenging problem is to use Physarum machines in the process of optimization of rule-based systems. Physarum polycephalum is originally famous as a computing biological substrate due to its alleged ability to approximate shortest path from its inoculation site to a source of nutrients (cf. [1]).

Acknowledgments

This research is being fulfilled by the support of FP7-ICT-2011-8.

References

Rough Sets Inspired Extension of Forward Inference Algorithm

Roman Siminski and Alicja Wakulicz-Deja

Institute of Computer Science, University of Silesia, Poland
{roman.siminski,alicja.wakulicz-deja}@us.edu.pl

Abstract. The main goal of this work is to introduce theoretical background of the extended forward inference algorithm. Proposed algorithm allow to continue inference after its failure. Inference failure means that the inference engine is unable to obtain the solutions — the new facts or goals confirmation. Two-phase extension of classical inference algorithm is considered. In the first phase, classical forward inference is executed. If inference fails, second phase is activated and targeted search for additional facts is executed in the interactive mode. Inference extension proposed in this work is inspired by the rough sets theory which provides the conception of lower and upper approximations of particular sets.

Key words: knowledge base, forward inference, rules groups

1 Introduction

Rule-based systems are a well known solvers for specialized domains of competence, in which effective problem solving normally requires human expertise. This approach to solving ill-structured and non-algorithmic problems has been known for many years and it seemed that in this field everything had been said. But the rules are still the most popular, important and useful tool for constructing knowledge bases. During the last decade we could observe the growth of interest on rule knowledge bases applications.

In the presented work, the extension of the forward inference algorithm will be considered. The modifications and extensions of the inference algorithms for rule-based systems were described in the previous works [5, 3]. The main goal of this work is to introduce another extension of forward inference algorithm which allow inference to continue after its failure. Inference failure means that the inference engine is unable to obtain the solutions: the new facts or goals confirmation. Two-phases extension of classical inference algorithm is considered. In the first phase, classical forward inference is executed. If inference fails, second phase is activated and targeted search for additional facts is executed in the interactive mode. Proposed approach is limited to systems that have the ability to acquire new facts from the environment.

2 The Problem, Related Works, Proposed Approach

The real-world rules bases often contain only rules in the Horn clauses form. Inference with such rules can be done through the forward and backward chaining algorithms.
Forward chaining inference represents a data-oriented approach that searches for the solution space from an initial state to a final goal state. In a forward chaining system, the initial facts are processed using the rules to draw new facts as the conclusions of the applied rules. Contrary to the backward inference, forward chaining algorithm operates in the batch mode and it does not require an interaction with the system’s environment. The environment nature may vary depending on the system application. Typically the user interacts with the system, but in the context of embedded systems, information for inference can be provided by the technical equipment (for example: sensors, detectors). In the literature we can find forward inference optimization algorithms [4]. Regardless of the applied particular methods of forward inference, unsuccessful inference is an important problem. This leads to a negative assessment of knowledge base system, inference failure is interpreted as inability to obtain any solution by the system. The literature studies allow us to identify three main approaches to the unsuccessful inference.

First group of approaches finishes any consideration after inference failure. This is typical for currently used main expert system shells and frameworks for expert systems. Second group attempts to continue inference using question-asking strategies. If the initial information is insufficient, the system will ask the user for additional data. There are few articles which discuss the question-asking problem analytically. Some of the early applications of experts systems consider the issue of question asking strategies — the system EXPERT [2], PROSPECT [1]. In [10] the authors define an unconfirmed observable set of assertions \( P_i \), as a set of unconfirmed assertions that, if all were confirmed true, \( P_i \) would be proved true, but if even one was false, then \( P_i \) could not be concluded from the others. A smallest unconfirmed observable set over all the unconfirmed observable sets of top level assertions is called a global minimum inquiry set of the Horn clause system. The authors proved [10, 8] that the problem of finding a global minimum inquiry set of a Horn clause system is NP-hard, therefore they introduced and discussed the Minimum Usage Strategy — an efficient strategy approximating sub-effectiveness, in which question selection is a natural extension of the deduction process. Labelling Algorithm selects a next question for this strategy by using dynamic programming. In the [8] different strategies of question-asking strategy are considered. Other work [9] describes question-asking strategies for a Horn system in which the response costs of the questions and tile probabilistic estimates of the answers are given, the authors introduce a question sequencing rule and enhance an efficient question-asking strategy. Methods mentioned above try to deal with inference failure using variations of logic and probabilistic methods.

Third group of approaches treats inference failure as a symptom of incompleteness. The most popular approach to process incomplete data is based on the uncertain reasoning. Reasoning under uncertainty has been studied in the fields of statistic, probability theory and decision theory. The oldies methods applied in uncertain reasoning are probability theory, including Bayesian networks, certainty factors, relatively newer are Dempster-Shafer theory, fuzzy logic and fuzzy set, rough sets, several non-monotonic logic have been also proposed. Methods mentioned above differ significantly, they try to quantitative estimate the degree of truth information, which can not be true from the logical point of view.
In this work we want to introduce different proposals of dealing with inference failure. We propose an extension of classical inference algorithm, which includes two steps. In the first stage, classical forward inference is executed. If inference is successful, its result is presented in the typical way and the second stage is unnecessary and not activated. If inference fails, we propose simple idea: resumption of the inference and the acquisition of the missing facts. It could be done if a system is able to work in the interactive mode and the source of information about missing facts exist. Typically, the end-user may provide such information, in general, such information can be provided by any system environment, which is able to operate interactively (for example technical equipment with sensors, detectors). The acquisition of the missing facts seems to be a naive idea, however, we propose targeted search for additional facts, based on the results of failed inference from first stage. Detailed description of proposed approach is the topic of the next section. Introduced approach is similar to the described above question-asking method, the main difference is the utilisation of rough set inspired method of evaluating the acceptable fact extension set. In contrast to described question-asking method, proposed approach appends to resume classical forward inference and acceptable fact extensions a trigger for new inference.

3 Methods

3.1 Preliminary Issues

In presented work the knowledge base is defined as a pair \( KB = (R, F) \) where \( R \) is a non-empty finite set of rules and \( F \) is a finite set of facts. \( R = \{r_1, r_2, \ldots, r_n\} \), each rule \( r \in R \) will have a form of Horn’s clause \( r : p_1 \land p_2 \land \cdots \land p_m \rightarrow c \), where \( m \) — the number of literals in the conditional part of rule \( r \), and \( m \geq 0 \), \( p_i \) — \( i \)-th literal in the conditional part of rule \( r \), \( i = 1 \ldots m \), \( c \) — literal of the decisional part of rule \( r \). For each rule \( r \in R \) the following functions are defined:

- \( \text{concl}(r) \) — the value of this function is the conclusion literal of rule \( r \): \( \text{concl}(r) = c \);
- \( \text{cond}(r) \) — the value of this function is the set of conditional literals of rule \( r \): \( \text{cond}(r) = \{p_1, p_2, \ldots, p_m\} \);
- \( \text{literals}(r) \) — the value of this function is the set of all literals of rule \( r \): \( \text{literals}(r) = \text{cond}(r) \cup \{\text{concl}(r)\} \);
- \( \text{csizeof}(r) \) — conditional size of rule \( r \), equal to the number of conditional literals of rule \( r \): \( \text{csizeof}(r) = |\text{cond}(r)| = m \);
- \( \text{sizeof}(r) \) — whole size of rule \( r \), equal to the number of conditional literals of rule \( r \) increased by the 1 for single conclusion literal, for rules in the form of Horn’s clause: \( \text{sizeof}(r) = \text{csizeof}(r) + 1 \).

We will also consider the \textit{facts} as clauses without any conditional literals. The set of all such clauses \( f \) will be called \textit{set of facts} and will be denoted by \( F : F = \{f : \forall f \in F \text{ cond}(f) = \{\} \land f = \text{concl}(f)\} \).

In this work, rule’s literals will be denoted as the pairs of attributes and their values. Attributes are defined in a manner that is quite similar to that of a rough set theory. Let
A be a nonempty finite set of conditional and decision attributes\(^1\). For every attribute \(a \in A\) the set \(V_a\) will be denoted as the set of values of attribute \(a\). Attribute \(a \in A\) may be simultaneously conditional and decision attribute. Also a conclusion of a particular rule \(r_i\) can be a condition in other rule \(r_j\). It means that rule \(r_i\) and \(r_j\) are connected and it is possible that inference chains to occur. The literals of the rules from \(R\) are considered as attribute-value pair \((a, v)\), where \(a \in A\) and \(v \in V_a\). Furthermore the notation \((a, v)\) and \(a = v\) is equivalent.

### 3.2 Unsuccessful Inference

Typically, inference failure is the result of incompleteness — it is possible to consider incompleteness of facts actually describing problem to solve, and/or the incompleteness or rules base. The condition \(p \in \text{cond}(r)\) of rule \(r\) will be true, if it is a fact: \(p \in F\). For each rule \(r\) and non-empty set of facts \(F \neq \emptyset\) it is possible to show following cases of matching the conditional part of each rule \(r \in R\): \(\text{cond}(r)\) and set of facts \(F\):

1. \(\text{cond}(r) \subseteq F\) — full matching, all rule’s conditions are facts: \(\forall_{p_i \in \text{cond}(r)} p_i \in F\), rule \(r\) is fireable, and \(r\) is able to draw new fact \(\text{concl}(r)\).
2. \((\text{cond}(r) \not\subseteq F) \land (\text{cond}(r) \cap F = \emptyset)\) — the lack of matching, all rule’s conditions are not facts: \(\forall_{p_i \in \text{cond}(r)} p_i \not\in F\), rule \(r\) is definitively not fireable, and is not able to draw new fact.
3. \((\text{cond}(r) \not\subseteq F) \land (\text{cond}(r) \cap F \neq \emptyset)\) — partial matching, not all rule’s conditions are facts: \(\exists_{p_i \in \text{cond}(r)} p_i \not\in F\), rule \(r\) is not fireable, and is not able to draw new fact.

The degree of matching rule \(r\) to the facts \(F\) we will describe using rule to facts matching factor \(MF(r)\) defined in the following way:

\[
MF(r) = \frac{|\text{cond}(r) \cap F|}{|\text{cond}(r)|}
\]

For cases described above: \(MF(r) = 1\) for case 1, \(MF(r) = 0\) for case 2, \(0 < MF(r) < 1\) for case 3. From the inferential point of view, both cases 2 and 3 cause the impossibility of obtaining new facts. However, in the third case the rule \(r\) is promising — it will be fireable when we will succeed to asset as true conditions, that were considered to be false until now. The higher the \(MF(r)\) is, the greater is rule \(r\) usefulness. For the clarity of the further presentation, let \(R^F\) be the set of rules fully matched to the facts, \(R^P\) be the set of rules partially matched to the facts and \(R^N\) denotes the set of rules that do not match to the facts at all:

\[
R^F = \{r \in R : \text{cond}(r) \subseteq F\}
\]
\[
R^P = \{r \in R : (\text{cond}(r) \not\subseteq F) \land (\text{cond}(r) \cap F \neq \emptyset)\}
\]
\[
R^N = \{r \in R : \text{cond}(r) \cap F = \emptyset\}
\]

\(^1\) Decision attributes are attributes that are at least once included in conclusion of any rule from \(\mathcal{R}\).
3.3 Rough Set Inspiration

Inference extension proposed in this work is inspired by the rough sets theory [7]. Rough sets theory provides the conception of lower and upper approximations of particular sets [6]. In general, the proposed approach is inspired by the set approximation, but relation with the rough set theory is not strict. The description presented below informally refers to the lower and upper approximations.

Based on the knowledge provided by the rule \( r \in R \) and nonempty set of facts \( F \), it is possible to identify the set of rule \( r \) conditions, which can be with certainty classified as the facts:

\[
F(r) = \{ c \in \text{cond}(r) : \text{cond}(r) \subseteq F \}.
\]

Informally, the set \( F(r) \) is conceptually similar to the lower approximation of the fact set \( F \) under the knowledge described by the rule \( r \).

It is also possible to identify set of rule \( r \) conditions, which can be only classified as possible facts:

\[
F_{\text{BN}}(r) = \{ c \in \text{cond}(r) : \text{cond}(r) \cap F \neq \emptyset \}.
\]

The items of \( F_{\text{BN}}(r) \) are facts or they appear in the rule \( r \) premises partially matched to the facts set. Informally, the set \( F_{\text{BN}}(r) \) is conceptually similar to the upper approximation of the fact set \( F \) under the knowledge described by the rule \( r \).

The set \( F_{\text{BN}}(r) \) will be called basic fact extension set for rule \( r \).

It is possible to consider approximation-like total sets described above, for any nonempty set of rules \( R \subseteq R \):

\[
F_T(R) = \{ F(r) : r \in R \},
\]

\[
F_{\text{T}}(R) = \{ F(r) : r \in R \},
\]

\[
F_{\text{TBN}}(R) = F(R) - F(R)
\]

The first naive extension of forward inference proposed in this work, will use the set \( F_{\text{TBN}}(R) \) it will be called fact extension set for rules \( R \). The pseudo-code 1 presents the implementation of classical forward inference algorithm (CFI). The input data are — rule base: \( R = \{ r_1, r_2, ..., r_m \} \), facts set: \( F = \{ f_1, f_2, ..., f_k \} \). The output data are — new facts in \( F = \{ f_1, f_2, ..., f_k, f_{k+1}, ..., f_{nk} \} \), function result: \( \text{true} \) if new facts inferred, \( \text{false} \) otherwise.

The first proposed extended forward inference algorithm (EFI) is presented by the pseudo-code 2. At the beginning the EFI algorithm calls classical forward inference CFI. If inference is successful, function CFI returns true and the first stage of the EFI algorithm is the last one — the EFI works like CFI. If inference is unsuccessful, function CFI returns false and the second stage of algorithm is activated. On this stage, the promising rules subset \( R \) is selected and the \( F_{\text{BN}}(R) \) is determined. The \( F_{\text{BN}}(R) \) set contains sets of conditions representing possible facts. This set can be ordered by the selected strategy — for simplicity the minimal subset will be considered first.

Next, the algorithm looks for first acceptable extension of the facts set \( e \in F_{\text{BN}}(R) \). Decision about the truth of the fact is taken by the function accepted, which return true if proposed extension \( e \) is acceptable, false otherwise. In object-oriented implementation of EFI algorithm, function accepted is defined as the abstract function or method. It
**Algorithm 1: CFI — Classical Forward Inference algorithm**

```
function CFI( R, F ) : boolean
    var A ← ∅
    var NF ← ∅
    begin
        select rules subset R^F from R according to F
        while R^F ≠ ∅ do
            r ← select rule from R^F according to current selection strategy
            NF ← NF ∪ {concl(r)}
            F ← F ∪ NF
            A ← A ∪ {r}
            select rules subset R^F from R − A according to F
        end while
        return NF ≠ ∅
    end function
```
3.4 Criticism of the Proposed Solution

Proposed method of inference after failure looks for first acceptable facts sets extension $F_{TBN}(r)$. Method of calculation the possible facts extension set described above, ensures successful classical inference after acceptance of any set’s element. It can trigger only one step of classical inference, or it may cause many iteration of the algorithm. Unfortunately, the results of inference may be unpredictable and distant from expectation. Additionally, iterative acquisition of acceptable fact extension may be uncomfortable for the user — the number of possible query for large rule bases can be high, thus unacceptable in the real-world applications.

Proposed algorithm is also sensitive to the order of selection of the $F_{TBN}(R)$ set’s elements. Different order of the elements selected from $F_{TBN}(R)$ can lead to different inference results. It is not clear whether the obtained inference results are useful and satisfactory from the point of view of the problem being solved. Although the proposed solution may be capable of providing potentially useful results, it is possible to formulate more appropriate methods of determination of $F_{TBN}(R)$ set and selection of promising facts extension. As presented below, extraction of the internal rules dependencies will be a source of information for improving selection of possible facts.

3.5 Rules Groups as Simple Decision Model

For each rule base $R$ with $n$ rules, it is possible to consider a partition $PR$ of a set $R$. $PR$ is grouping of set’s $R$ rules into non-empty subsets, in such way that every rule is included in one and only one of the subsets: $\emptyset \notin PR, \bigcup_{R \in PR} = R$ and if $R_i, R_j \in PR$ and $R_i \neq R_j$ then $R_i \cap R_j = \emptyset$. The sets in $PR$ will be called the rules groups of the partition. In this work only simple partitioning strategies will be considered. The membership criterion function decides about the membership of rule $r$ in a particular group $R \subseteq PR$ according to the membership function $mc$. Simple strategy divides the rules by using the algorithm with time complexity not higher than $O(n \cdot k)$, where $n = |R|$ and $k = |PR|$. Simple strategy creates final partition $PR$ by a single search of the rules set $R$ and allocation of each rule $r$ to the proper cell $R$, according to the value of the function $mc(r, R)$ described bellow.

Proposed approach assumes that the membership criteria is defined by the $mc$ function, which is defined individually for every simple partition strategy. The function: $mc : R \times PR \rightarrow [0..1]$, return the value 1 if the rule $r \in R$ with no doubt belongs to the group $R \subseteq PR$, 0 in the opposite case. The value of the function from the range $0 < mc < 1$ means the partial membership of the rule $r$ to the group $R$. The method of determining its value and its interpretation depends on the specification of a given partition method. It is possible to achieve many different partitions of rule base using single $mc$ function.

The simple strategy partitioning algorithm is presented [3], it is simple and for this reason will be omitted. The input parameters are: the knowledge base $R$, the function $mc$ that defines the membership criteria, and the value of the threshold $T$. Output data is the partition $PR$. In connection with the main problem, two partitioning strategies will be discussed. The first is basic decision oriented partition $PS_B$ which creates groups of
the rules from $\mathcal{R}$ by grouping rules with the same conclusions. The membership criteria for rule $r$ and group $R$ is given by the function $mc$ defined as follows:

$$mc(r, R) = \begin{cases} 1 & \text{if } \forall r_i \in R \text{ concl}(r_i) = \text{concl}(r) \\ 0 & \text{otherwise} \end{cases}$$

(1)

By using the simple partition algorithm [3] with the $mc$ function defined in this way, we obtain the following groups: $R = \{ r \in \mathcal{R} : \forall r_i \in R \text{ concl}(r_i) = \text{concl}(r) \}$. The number of groups in the partition depends on the number of different decisions included in conclusions of such rules. When we distinguish different decisions by the different conclusions appearing in the rules, we get one group for each conclusion. All included in conclusions of such rules. When we distinguish different decisions by the different attribute from conclusions appearing in the rules — we obtain one group for each decision attribute. Thus any rule set from an elementary partition is similar to the decision tables $DT = (U, A \cup \{d\})$ — considered in the rough set theory [7, 6] — values of decision attributes $d$ defines partition of objects from $U$ into the decision class $CLASS_A(d) = \{ X_A^1, X_A^2, \ldots X_A^{|d|} \}$, where $r(d)$ is the cardinality of value set for attribute $a$ and $X_A^i$ is the $i$-th decision class of $A$. Each group of rules from an elementary partition is similar to $i$-th decision class $X_A^i$.

The second partitioning strategy considered in this work is decision oriented partition denoted $PR_D$. It also uses the simple partition algorithm with the $mc$ function defined in the following way:

$$mc(r, R) = \begin{cases} 1 & \text{if } \forall r_i \in R \text{ attrib(\text{concl}(r_i))} = \text{attrib(\text{concl}(r))}, \\ 0 & \text{otherwise.} \end{cases}$$

(2)

Each generated group of the rules have the following form: $R = \{ r \in \mathcal{R} : \forall r_i \in R \text{ attrib(\text{concl}(r_i))} = \text{attrib(\text{concl}(r))} \}$. When we distinguish different decisions by the different attribute from conclusions appearing in the rules — we obtain one group for each decision attribute. Thus any rule set $R \in PR_B$ can be described as $R = \{ \bigcup R' \in PR_B : \forall r_i, r_j \in R' \text{ attrib(\text{concl}(r_i))} = \text{attrib(\text{concl}(r_j))} \}$. It means that the decision produced by the ordinal decision partition can be constructed as the composition of the basic decision partitions. By analogy, decision oriented partition is similar to the decision tables $DT = (U, A \cup \{d\})$, where $d$ is single decision attribute.

The partitioning strategy $PR_B$ and $PR_D$ describes global dependencies appearing in the rule base. To express and utilize dependencies between rules in any partition $PR$, it is possible to define partitions connection graph $G_{PR} = (PR, C)$. $PR$ represents nodes of $G_{PR}$, $C \subseteq PR \times PR$ represents relation which defines edges of $G_{PR}$. In the context of connection graph, we assume that for any rules group $R$ we consider two sets — $In(R)$ and $Out(R)$. $In(R)$ and $Out(R)$ denote respectively the set of input and output data of the rules set. Items of those sets are literals appearing in the rules from $R$. The structure of $In(R)$ and $Out(R)$ depends on the currently considered partition strategy. The $C$ relation can be defined in the following way: $C = \{(x, y) \in PR \times PR : \text{Out}(x) \cap \text{In}(y) \neq \emptyset \}$. In the introduced modification of inference, the decision partition strategy is considered and following mappings are proposed: $Out(R) = Concl(R)$, $In(R) = Cond(R)$. Thus, the $C$ relation connects the group of rules with the common attributes in conclusion and conditions. The subsets can be defined: connected inputs sub-set $In_C(R) \subseteq In(R)$ can be considered:
\[ In_C(R) = \{(a, v) \in In(R) : \exists r \in R \ (a, v) = concl(r)\}, \text{ and connected output sub-set} \]

\[ Out_C(R) \subseteq Out(R) : Out_C(R) = \{(a, v) \in Out(R) : \exists r \in R \ (a, v) \in cond(r)\}. \]

### 3.6 Rules Groups in After Failure Inference Algorithm

Simple decision models provided by the two previously described partitioning strategies allow to direct the searching of the fact extension set. For relatively small rule sets, or „flat” rules sets (\( \forall_{R \subseteq PR} In_C(R) = \emptyset \land Out_C(R) = \emptyset \)), basic decision partition can be used. The partitions connection graph \( G_{PR} \) provides enough information to identify such situations. For rules bases containing big number of rules, basic decision partition may produce large number of rules group and ordinal decision partition strategy may be used. Apart of used partitioning method, decision model allows to consider different strategies of selection the promising rules set for facts extension searching.

First, it is possible to reject rules sets from \( PR \) with empty intersection with facts set. Let’s assume that \( PR_F \subseteq PR \) is subset of partition \( PR \) with non empty intersection with the facts set: \( PR_F = \{R : \exists r \in R \ cond(r) \cap \mathcal{F} \neq \emptyset\} \). For each group \( R \in PR \) it is possible to determine fact extension set \( \mathcal{F}_{TBN}(R) \). Thus, the set of fact extension sets can be considered: \( \text{FES} = \{\mathcal{F}_{TBN}(R) : R \in PR_F\} \).

Selection of the first acceptable fact set extension will consist of two steps. In first step, the most promising rules group from \( PR_F \) will be selected. In the second step, the most adequate facts extension set for selected rules set will be promoted for acceptance. The combination of two above steps provides the different strategies of selecting facts extension set. On the level of the most promising rules group \( R \) selection, it is possible to indicate the following possibilities:

- The connected output sub-set: \( Out_C(R) = \emptyset \) — inference will produce the one or multiple new facts from \( Out(R) \) and the inference will stop. The new facts are unable to trigger any other rule. This rule selection strategy offers shortest inference path and predictable effects — only facts from \( Out(R) \) are expected.

- The connected output sub-set: \( Out_C(R) \neq \emptyset \) — inference will produce the one or multiple new facts from \( Out(R) \) and inference will likely continue. The new facts are able to trigger other rules from rules group connected with \( R \). This rule selection strategy offers the ability to obtain new facts not only from the \( Out(R) \).

- It is possible to select the rule group \( R \) as a start point of longest path in the \( G_{PR} \). This strategy offers the possibility of obtaining a variety new facts.

- The rules group with maximal facts coverage can be selected — \( MF(R) = \frac{|\text{Cond}(R) \cap \mathcal{F}|}{|\text{Cond}(R)|} \).

The modified after failure inference algorithm differs in two lines and therefore will not be presented as separate pseudo-code. The line number (1) in the algorithm 2 should be replaced by the following two lines:

1. \( PR = \text{createDecisionPartition}( \mathcal{R} ) \)
2. select most promising rules subset \( R \subseteq PR \)
4 Implementation Issues and First Experiments

The first version of EFI algorithm (2) has been implemented as a part of kbExplorator desktop system. kbExplorator is the system which integrates two components — desktop and web application. Web application allows to create, edit and manage rules bases which are stored on the kbExplorator server. The knowledge bases are assigned to the users registered in the web application. Desktop application allows to perform different operations on the knowledge base stored on the server, operations jointly referred to as exploration. Web application is mainly dedicated to the knowledge engineers as a tool for knowledge base building and improving. Desktop application is the tool for more sophisticated knowledge exploration. The kernel of the desktop application consist of object-oriented packages implemented in Java. Packages offer, among other things, different types of reasoning, both classic and modified versions and also new algorithms, including EFI proposed in this work. Applications now have the status of a prototype, they will be available as the free software online in the coming months. Till now, kb-Explorator is one of the few expert system building tools which allows to continue inference after its failure.

The first version of EFI algorithm selects a minimal acceptable facts extensions. The main advantage of this algorithm is its simplicity. Unfortunately, experiments confirmed earlier expectations, the results of inference may be unpredictable and iterative acquisition of acceptable fact extension have proven to be uncomfortable for the user. For large rule bases, iterative acquisition of missing fact was not acceptable for the real-world applications.

Proposed algorithm is also sensitive to the order of selection of the fact extension set elements. Different order of the elements selections can lead to different inference results. It is not clear whether the obtained inference results are useful and satisfactory from the point of view of the problem being solved. Although the proposed solution may be capable of providing potentially useful results, modification proposed in the previous section will be considered in the future work. Modified EFI algorithm utilizes the internal dependencies between rules divided into the decision oriented group of rules. Detailed experiments will be made in the next stage of research and the experimental results will be presented in the next publications. One of the reasons is the need of implementation of similar methods for a comparative study. Unfortunately, there is a lack of detailed source information which presents enough information about the implementation details on the proper level of detail. Additional literature studies are needed and future research could be focused on a comparative study of algorithms proposed in this work with the question-asking oriented methods.

Main tool of proposed optimisation are decision oriented partitions of rule base. The complexity of decision partition is $O(n \cdot k)$, where $n = |R|$, $k = |PR|$, where the number of groups in the partition $k : 1 \leq k \leq n$ typically is significantly smaller than the number of rules $n$. We have to store additional information for created rules partitions, however additional memory or disk space occupation for data structures seems acceptable. For $n$ rules and $k$ rules group we need approximately $is \cdot n + ps \cdot m$ bytes of additional memory for data structures ($is$ - size of integer, $ps$ - size of a pointer or reference).
5 Conclusions

In the presented work, the conception of the extended forward inference algorithm was presented. This algorithm allows to continue the inference after its failure. Inference failure means that the inference engine is unable to obtain the solutions: the new facts or goals confirmation. It often happens that the initial facts are not sufficient for a knowledge-based system to reach any conclusion, and more information is needed. In this work two-phase extension of classical inference algorithm was considered. In the first phase classical forward inference is executed. If inference fails, second phase is activated and targeted search for additional facts is executed in the interactive mode. Proposed solution is similar to a question-asking method described in [10, 8]. Question-asking is to figure out what additional information should be known if no useful results can be proved with the available data. The QA problem is computationally hard in a propositional knowledge-based system, even if the system is composed only of Horn clauses. The existing approaches are based on logic. Described solutions try to examine an unconfirmed observable assertion set of a top level conclusion. Inference extension proposed in this work is inspired by the rough sets theory, which provides the conception of lower and upper approximations of particular sets.

In general, the proposed approach is inspired by the set approximation, but relation with the rough set theory is not strict. The description presented in this article informally refers to the lower and upper approximations. Based on the knowledge provided by the rules and set of facts, it is possible to identify the set of rules conditions which can be with certainty classified as the facts. It is conceptually similar to the lower approximation of the fact set. It is also possible to identify set of rules conditions which can be only classified as possible facts, because they appear in the rules premises partially matched to the facts set. It is conceptually similar to the upper approximation of the fact set. The boundary set contains conditions which represent missing facts. If these conditions are facts, some rules will be fireable. It is informally similar to the boundary region of facts, and thus consists of those conditions that cannot be classify as the facts on the basis of knowledge described in the rules base, but are interesting in the context of inference after failure. This work introduced theoretical background of the algorithm, detailed experiments will be made in the next stage of research and the experimental results will be presented in the next publications.

Acknowledgements

This work is a part of the project „Exploration of rule knowledge bases” founded by the Polish National Science Centre (NCN: 2011/03/D/ST6/03027).

References


Hybrid Planning by Combining SMT and Simulated Annealing

Extended Abstract

Jaroslaw Skaruz\(^1\), Artur Niewiadomski\(^1\), and Wojciech Penczek\(^2\)

\(^1\) Institute of Computer Science, Siedlce University, Poland
{jaroslaw.skaruz}, {artur.niewiadomski}@uph.edu.pl
\(^2\) Institute of Computer Science, PAN Warsaw and Siedlce University, Poland
wpenczek@gmail.com

1 Introduction

We present a new approach to the concrete planning (CP) shown to be a NP-hard problem [7]. This is the third stage of Web service composition (WSC) in the Planics framework [1]. The first two phases, namely an abstract planning and offer collecting, basing on an ontology, a user query, and a service registry provide data necessary for CP. A new hybrid algorithm (HSA), which combines Simulated Annealing (SA) [6] with Satisfiability Modulo Theories (SMT) [3], has been designed and implemented. The main idea of our hybrid solution relies upon generating an initial individual by an SMT-based procedure. Then, in the subsequent iterations of SA, the individual is improved. The experimental results show that HSA is superior to the other methods we have applied to the CP problem, including these based on Genetic Algorithm (GA) [4], SMT used separately [10], and SMT combined into the hybrid algorithms RH and SRH [9], as well as the IPH algorithm [8].

Our direct motivation to develop hybrid algorithms is based on the observation that every method applied separately to WSC yields fair results, but suffers from some disadvantages. While the SMT-based algorithm is able to find always the optimal solution, its main problem is a long execution time and large memory consumption. On the other hand the evolutionary methods are quick and demand less resources, but at the price of the quality and a lower probability of finding solutions. We are aiming at combining the algorithms in order to get a trade-off between speed and quality.

Recently, we have developed three planning methods based on joining GA and SMT: RH (Random Hybrid), SRH (Semi-Random Hybrid), and IPH (Initial Population Hybrid). The first two algorithms run alternately several iterations of GA and the SMT-based procedure which is aimed at improving the best individuals of a GA population. The IPH algorithm makes use of an SMT-based procedure in order to generate (a part of) the initial population meeting the given constraints, and then the individuals are improved by GA. The experiments have shown that the latter method is superior in most cases, thus we have chosen this scheme to be used in further investigations.

*This work has been supported by the National Science Centre under the grant No. 2011/01/B/ST6/01477.
After replacing GA by SA (and introducing several necessary technical modifications) we have obtained the HSA algorithm which seems to be the best one so far.

The problems similar to CP have been intensively studied in the last decade. In [2] the authors use two algorithms to deal with WSC, namely GA and SA. Experiments show better quality of the solutions found by SA comparing to GA, but at the price of a higher computation time and lower scalability. On the other hand, hybrid algorithms for WSC are also known in the literature, see e.g., [5], where a modified version of the Particle Swarm Optimization algorithm is used.

Next, we present our solution followed by experimental results and conclusions.

2 Solution

First, we briefly recall the formulation of CP as a constrained optimization problem. We focus here on the intuitions only, but all the formal definitions can be found in [10].

The input for the concrete planning consists of \( n \) offer sets and a set of constraints. Each offer from a particular set is a tuple of values corresponding to attributes of objects being processed by services. Overall, a solution of CP consists in selecting one offer from each offer set, such that all the constraints are satisfied and the value of the quality function \( Q \) (given as a part of a user query) is maximized.

Algorithm. SA is an optimisation technique based on an observation of the physical process of annealing in metallurgy, i.e., a technique of heating and slow cooling of a material in order to improve its properties. SA implements the cooling process as a slow decrease in the probability of accepting worse solutions while the algorithm explores the search space. In SA, our “processed material” is a potential solution of a problem, called an individual. The space exploration is realised by applying a neighbourhood operator to the individual, which results in obtaining a new potential solution. In our case, the individual is a sequence of natural values representing offer indices, while the neighbourhood operator changes one value randomly.

It is quite hard to force SA to search for better solutions with the constraints satisfied, because employing standard mechanisms like penalty functions known from GA is problematic. Thus, our algorithm applies an SMT-based procedure (similar to the one described in [9]) to generate an initial individual which satisfies all constraints, and the next steps of our hybrid algorithm are similar to those of a standard SA. The main difference is that new individuals may be accepted provided that all constraints are still satisfied. Since an initial individual is already a solution (usually of a low quality), the HSA algorithm always returns a result, and the objective is to increase the value of the quality function.

Experiments. We have evaluated the efficiency of HSA using the same six benchmarks as in the papers [10, 9]. All the instances represent plans of length 15. Each offer set of Instance 1, 3, and 5 contains 256 offers, which makes the number of the potential solutions equal to \( 256^{15} = 2^{120} \). In the case of Instance 2, 4, and 6, each offer set consists of 512 offers, which results in the search space size as large as \( 512^{15} = 2^{135} \).

In Table 1 we compare the performance of HSA with our previous results. The first column contains the instance number, while the next columns present computation
times and the quality function values of the solutions found with our different methods: HSA, IPH (in two variants - with 1 and 500 individuals generated by SMT), and non-hybrid ones: SMT and GA. The HSA algorithm is the fastest one and finds solutions of reasonable quality. It has to be mentioned that all the presented methods return solutions with 100% probability, except for GA, where the probability is much lower: from 7% to 12%.

Table 1. Experimental results of the IPH and HSA algorithms.

<table>
<thead>
<tr>
<th>Instance</th>
<th>HSA t[s] Q</th>
<th>IPH1 t[s] Q</th>
<th>IPH500 t[s] Q</th>
<th>SMT t[s] Q</th>
<th>GA t[s] Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.0 1420</td>
<td>5.6 1229.5</td>
<td>13.9 1317.9</td>
<td>266.0 1443.0</td>
<td>5.0 1218.5</td>
</tr>
<tr>
<td>2</td>
<td>5.1 1402</td>
<td>6.4 1248.8</td>
<td>20.1 1382.4</td>
<td>388.0 1467.0</td>
<td>5.6 1319.9</td>
</tr>
<tr>
<td>3</td>
<td>5.4 2153</td>
<td>6.4 1706.8</td>
<td>14.6 2090.6</td>
<td>500.0 2266.0</td>
<td>6.0 2085.4</td>
</tr>
<tr>
<td>4</td>
<td>6.5 2194</td>
<td>7.5 1788.0</td>
<td>20.9 2280.3</td>
<td>500.0 2409.0</td>
<td>6.6 2001.9</td>
</tr>
<tr>
<td>5</td>
<td>4.7 354</td>
<td>6.4 329.1</td>
<td>27.8 634.5</td>
<td>500.0 781.0</td>
<td>5.1 436.0</td>
</tr>
<tr>
<td>6</td>
<td>6.3 541</td>
<td>8.8 458.2</td>
<td>55.2 524.7</td>
<td>500.0 755.0</td>
<td>5.9 537.8</td>
</tr>
</tbody>
</table>

3 Conclusions

We have presented a new approach to concrete planning. An SMT-based procedure is employed for generating an initial individual of the SA algorithm. Such an individual is already a solution because it satisfies all the constraints. During the next steps of our algorithm the individual is improved in order to find a concrete plan having a greater value of the quality function. The experiments confirm that HSA is efficient, since the results are better than those obtained by all our former algorithms, including the hybrid ones.

References

Optimization of Backward Fuzzy Reasoning Based on Rule Knowledge

Zbigniew Suraj¹, Piotr Grochowalski¹, and Sibasis Bandyopadhyay²

¹ Chair of Computer Science, University of Rzeszow, Poland
{zbigniew.suraj,piotrg}@ur.edu.pl
² Department of Mathematics, Visva-Bharati, Santiniketan, India
sibasisbanerjee@rediffmail.com

Abstract. In [14], we have presented a fuzzy forward reasoning methodology for rule-based systems using the functional representation of rules (fuzzy implications). In this paper, we extend methodology for selecting relevant fuzzy implications from [14] in backward reasoning. The proposed methodology takes full advantage of the functional representation of fuzzy implications and the algebraic properties of the family of all fuzzy implications. It allows to compare two fuzzy implications. If the truth value of the conclusion and the truth value of the implication are given, we can easily optimize the truth value of the implication premise. This methodology can be useful for the design of an inference engine based on the rule knowledge for a given rule-based system.

Key words: fuzzy implication, knowledge representation, backward reasoning, rule-based system

1 Introduction

Recently we can observe further growth of an interest in the design and exploitation of rule-based systems built on the basis of uncertain knowledge. Various methods of knowledge representation and reasoning have already been proposed. One of the most popular approaches to knowledge representation are the fuzzy production rules. However, reasoning is mainly classified into two types: forward reasoning and backward reasoning. The inference mechanism of forward reasoning is based on a data-derived way, and has a powerful prediction ability. It is capable of warning against latent hazards, forthcoming accidents, and faults. By contrast, backward reasoning is based on a goal-derived manner, it has explicit objectives, which are generally used to search for the most possible causes related to an existing fact. Backward reasoning plays an essential role in fault diagnosis, accident analysis, and defect detection.

In this paper, we mainly focus on backward reasoning based on the fuzzy rules. They can be presented in the form of IF-THEN and interpreted as fuzzy implications [1]. There exist uncountably many implication functions in the field of fuzzy logic, and the nature of the fuzzy inference changes variously depending on the implication function to be used. The variety of implication functions existing in the fuzzy set framework has always been seen as a rich potential for modeling different shades of expert attitude.
in the inference process (e.g. [7]), although no precise, practical interpretation was provided for the different implication functions [10]. Moreover, it is very difficult to select a suitable implication function for actual applications.

From over eight decades a number of different fuzzy implications have been proposed [2],[4]-[6],[8]-[9],[11]-[12],[17]-[18]. In the family of basic fuzzy implications the partial order induced from [0,1] interval exists. Pairs of incomparable fuzzy implications can generate new fuzzy implications by using min(inf) and max(sup) operations. As a result the structure of lattice is created ([1], page 186). This leads to the following question: how to choose the relevant functions among basic fuzzy implications and other generated as described above.

In [14], we have presented a fuzzy forward reasoning methodology for rule-based systems using the functional representation of rules (fuzzy implications). In this paper, we extend a methodology for selecting relevant fuzzy implications from [14] in backward reasoning. The proposed methodology takes full advantage of the functional representation of fuzzy implications and the algebraic properties of the family of all fuzzy implications. It allows to compare two fuzzy implications. If the truth value of the conclusion and the truth value of the implication are given, we can easily optimize the truth value of the implication premise. This general methodology is considered in details in [13]. It can be useful for the design of an inference engine based on the rule knowledge for a given rule-based system. Using the proposed approach, we can reduce the efforts related to a selection of a suitable implication function.

The rest of this paper is organized as follows. In Sect. 2, we briefly recall some definitions related to partially ordered sets, the fuzzy production rules, fuzzy implications and basic algebraic properties of fuzzy implications. The research problem considered in the paper is formulated in Sect. 3. Sect. 4 presents the main theorem together with its proof concerning a selection of suitable implication function. Sect. 5 presents two algorithms solving the given research problem. The first algorithm allows to select the suitable implication function based on information concerning a given set of fuzzy implications, their truth-values, and the truth value of conclusion. The second algorithm allows to select the "optimal" fuzzy implication using the same information as for the first one. In Sect. 6, we present an example illustrating these algorithms in the use. Sect. 7 includes the summary of our research and some remarks.

2 Basic Notions and Definitions

2.1 Partially Ordered Sets

Let \( R \) be a binary relation on a set \( A \). A relation \( R \) on \( A \) is said to be a partial ordering on \( A \) if it is reflexive, transitive and antisymmetric. A partial ordering \( R \) on \( A \) is said to be a linear ordering on \( A \) if at least one of the following conditions: \((x, y) \in R, (y, x) \in R \) or \( x = y \) holds for any \( x, y \in A \). If \( R \) is a partial ordering on \( A \), then the pair \( U = (A, R) \) is said to be a partially ordered set (abbreviated poset). If \( R \) is a linear ordering on \( A \), then the pair \( U = (A, R) \) is said to be a linearly ordered set.

Let \( U = (A, R) \) be a poset, and \( X \subseteq A \). The element \( a_0 \in A \) is said to be the upper (lower) bound in \( U \) of a subset \( X \subseteq A \) if \((x, a_0) \in R ((a_0, x) \in R) \) for all \( x \in X \).
The upper (lower) bound in \( U \) of \( A \) is the greatest (least) element in \( U \). An element \( a \in A \) is said to be maximal (minimal) in \( U \) if \( (a, x) \in U \) (respectively \( (x, a) \in R \)) implies \( x = a \). It is clear that the greatest (least) element is maximal (minimal), and if \( R \) is a linear ordering, then the element maximal (minimal) in \( U \) is also the greatest (least) in \( U \). It is obvious that if the greatest (least) element in \( U \) exists, then all the maximal (minimal) elements are equal. If \( B \) is a set of upper bounds in \( U = (A, R) \) of a set \( A_1 \subseteq A \), then the least element in \( (B, R \cap B^2) \) is said to be the least upper bound in \( U \) of the set \( A_1 \) and is denoting \( \sup(A_1, U) \). Replacing in the preceding definition "upper" and "least" respectively by "lower" and "greatest" we obtain the definition of the greatest lower bound of \( A_1 \) in \( U \) which will be denoted by \( \inf(A_1, U) \). It is clear that \( \sup(A_1, U) \) and \( \inf(A_1, U) \) are uniquely determined by \( A_1 \) and \( U \) if they exist. A poset \( U \) is said to be a lattice if for any \( a, b \in A \) in \( U \) there are \( \sup\{a, b\}, U \) and \( \inf\{a, b\}, U \). If \( R \cap X^2 \) is a linear ordering on \( X \), then \( X \) is said to be a chain in \( U \).

For more detailed information about partially ordered sets the reader is referred to [3].

### 2.2 Fuzzy Production Rules and Fuzzy Implications

Let \( R \) be a set of fuzzy production rules, \( R = \{r_1, r_2, ..., r_n\} \). The general formulation of the \( i \)-th fuzzy production rule is as follows:

\[
r_i : \text{IF} \; d_j \; \text{THEN} \; d_k \; (\text{CF}=z_i)
\]

where: (1) \( d_j \) and \( d_k \) are statements; the truth degree of each statement is a real value between zero and one. (2) \( z_i \) is the value of the certainty factor (CF), \( z_i \in [0, 1] \). The larger the value of \( z_i \), the more the rule is believed in.

We can use a fuzzy implication model [1] to represent the fuzzy production rules of a rule-based system.

Fuzzy implications are one of the main operations in fuzzy logic [1]. Now we recall a definition of a fuzzy implication and some of its properties that will be used in the paper.

A function \( I : [0, 1]^2 \rightarrow [0, 1] \) is said to be a fuzzy implication if it satisfies, for all \( x, x_1, x_2, y, y_1, y_2 \in [0, 1] \), the following conditions:

1. \( I(\cdot, y) \) is decreasing (i.e., if \( x_1 \leq x_2 \), then \( I(x_1, y) \geq I(x_2, y) \)).
2. \( I(x, \cdot) \) is increasing (i.e., if \( y_1 \leq y_2 \), then \( I(x, y_1) \leq I(x, y_2) \)).
3. \( I(0, 0) = 1, I(1, 1) = 1 \), and \( I(1, 0) = 0 \).

The family of all fuzzy implications will be denoted by \( FI \).

**Remark 1.** Let us observe that each fuzzy implication \( I \) is constant for \( x = 0 \) and for \( y = 1 \) (i.e., \( I \) fulfils the following conditions, respectively: (1) \( I(0, y) = 1 \) for \( y \in [0, 1] \), (2) \( I(x, 1) = 1 \) for \( x \in [0, 1] \)).

If, for two fuzzy implications \( I_1 \) and \( I_2 \), the inequality \( I_1(x, y) \leq I_2(x, y) \) holds for all \( (x, y) \in [0, 1]^2 \), then we say that \( I_1 \) is less than or equal to \( I_2 \) and we write \( I_1 \leq I_2 \). We shall write \( I_1 < I_2 \) whenever \( I_1 \leq I_2 \) and \( I_1 \neq I_2 \), i.e., if \( I_1 \leq I_2 \) and for some \( (x_0, y_0) \in [0, 1]^2 \) we have \( I_1(x_0, y_0) < I_2(x_0, y_0) \). In this case we also say that \( I_1 \) is
comparable with $I_2$. Moreover, if, for two fuzzy implications $I_1$ and $I_2$, the inequality $I_1(x, y) < I_2(x, y)$ holds for all $(x, y) \in D \subset [0, 1]^2$, then we say that $I_1$ is less than $I_2$ and we write $I_1 \prec I_2$.

**Example 1.** Since there exist uncountably many fuzzy implications, we list below only a few of basic fuzzy implications known from the subject literature. Figures 1 and 2 illustrate the plots of $I_{LK}$, $I_{RC}$, $I_{KD}$ and $I_{YG}$ implications, respectively.

**Fig. 1.** Plots of $I_{LK}$ and $I_{RC}$ fuzzy implications

**Fig. 2.** Plots of $I_{KD}$ and $I_{YG}$ fuzzy implications

1. $I_{LK}(x, y) = \min(1, 1 - x + y)$ (the Łukasiewicz implication) [9];
2. $I_{GD}(x, y) = 1$, if $x \leq y$, and $I_{GD}(x, y) = y$ otherwise (the Gödel implication) [5];
3. $I_{RC}(x, y) = 1 - x + xy$ (the Reichenbach implication) [11];
4. $I_{KD}(x, y) = \max(1 - x, y)$ (the Kleene-Dienes implication) [2],[8];
5. $I_{GG}(x, y) = 1$, if $x \leq y$, and $I_{GG}(x, y) = \frac{y}{x}$ otherwise (the Goguen implication) [6];
6. \( I_{RS}(x,y) = 1 \), if \( x \leq y \), and \( I_{RS}(x,y) = 0 \) otherwise (the Rescher implication) [12];
7. \( I_{WB}(x,y) = 1 \), if \( x < 1 \), and \( I_{WB}(x,y) = y \), if \( x = 1 \) (the Weber implication) [17];
8. \( I_{FD}(x,y) = 1 \), if \( x \leq y \), and \( I_{FD}(x,y) = \max(1 - x, y) \) otherwise (the Fodor implication) [4];
9. \( I_{YG}(x,y) = 1 \), if \( x = 0 \) and \( y = 0 \), and \( I_{YG}(x,y) = y^x \), if \( x > 0 \) or \( y > 0 \) (the Yager implication) [18].

**Example 2.** Let \( A \) be the basic fuzzy implications from Example 1, and \( R \) be the relation \(<\). It is easy to check that the pair \( U = (A,R) \) is a poset. A graphical representation of five chains: \( C_1 = \{I_{KD}, I_{RC}, I_{LK}, I_{WB}\} \), \( C_2 = \{I_{RS}, I_{GD}, I_{GG}, I_{LK}, I_{WB}\} \), \( C_3 = \{I_{YG}, I_{RC}, I_{LK}, I_{WB}\} \), \( C_4 = \{I_{KD}, I_{FD}, I_{LK}, I_{WB}\} \), \( C_5 = \{I_{RS}, I_{GD}, I_{FD}, I_{LK}, I_{WB}\} \) in \( U \) is shown in Figure 3.

![Fig. 3. A graphical representation of the chains from Example 2](image)

**Remark 3.** It is also worth to point out that incomparable pairs of fuzzy implications generate new fuzzy implications by using the standard \( \min \) and \( \max \) operations. In particular, incomparable pairs of basic implications from Example 1 generate new implications in the lattice of fuzzy implications. Elements obtained in such way can be combined with other implications, which leads to the new fuzzy implications forming the lattice of fuzzy implications. This issue will not be dealt with here, and we will refer the reader to ([1], page 186).

We can use fuzzy implications to represent the fuzzy production rules of a rule-based system. For example, the following fuzzy production rule \( r_i : \text{IF } d_j \text{ THEN } d_k \) (CF=\( z_i \)) can be interpreted as a fuzzy implication \( z = I(x,y) \), where values for \( z, x, y \) correspond to CF, the truth degree of a statement \( d_j \) (premise), and the truth degree of a statement \( d_k \) (conclusion), respectively. The value of \( z_i \) is given by a domain expert.
However, the value for \( x \) (or \( y \)) is given by the user of a rule-based system dependently on a selected reasoning method (forward or backward, respectively).

3 Problem Statement

Let us consider a lattice \((FI, <)\) \((1), page 183\), where \( FI \) is the family of all fuzzy implications and \( < \) is the inequality relation between fuzzy implications from \( FI \) induced in the standard way from the unit interval \([0,1]\) (see Sect. 2). Let \( U \) be a finite subset of \( FI \).

Our goal is to elaborate on two algorithms which using information on a value of an argument \( y \) of a given fuzzy implication \( J \) from \( U \) and a truth-value of the implication \( J \) find in the set \( U \) form:

1. a "worse" fuzzy implication \( I \) than \( J \) (if there exists) such that:
   \[ I(x_1, y) = J(x_2, y) \]
   for the given argument \( y \) and some arguments \( x_1, x_2 \in [0,1] \), and \( x_1 < x_2 \), i.e., a fuzzy implication \( I \) with the strictly less value of the argument \( x_1 \) than it is possible to compute using the implication \( J \);
2. an "optimal" (minimal) fuzzy implication \( I_{opt} \) (if there exists), i.e., a fuzzy implication that fulfils the following two requirements:
   - \( I_{opt}(x_1, y) = J(x, y) \),
   - \( x_1 \) is the least value among all values \( x' \) possible to obtain using any fuzzy implication \( K \) comparable with \( J \) belonging to the set \( U \) and satisfying the condition: \( K(x', y) = J(x, y) \).

4 Theorem

Now we are ready to formulate and prove a theorem which suggests how to select from a given finite set of fuzzy implications \( U \) the suitable implication function for a given fuzzy implication \( J \) in order to obtain a less truth-value of its premise \( x \) in reasoning taking into account information on the truth-value \( J(x, y) \) of this implication and the truth-value of its conclusion \( y \).

Theorem. Let \( I \) and \( J \) be fuzzy implications such that \( I \prec J \) on a set \( D \subset [0,1]^2 \), and \( x_1, x_2, y \in [0,1] \) such that \( I(x_1, y) = J(x_2, y) \). Then \( x_1 < x_2 \).

Proof: Proof by contradiction. Suppose \( x_1 \geq x_2 \). Then from the definition of fuzzy implication (see item 1) it follows that \( I(x_1, y) \leq I(x_2, y) \). From that and from the equality \( I(x_1, y) = J(x_2, y) \) it follows that \( I(x_2, y) \geq J(x_2, y) \). Since \( I \prec J \), \( I(x_2, y) < J(x_2, y) \). Thus, we have reached a contradiction. Therefore, we conclude that the theorem is correct.

Remark 4. The analogous theorem, but for forward fuzzy reasoning has been presented in [14]. Moreover, the detailed considerations related to a set \( D \) (the domain) for particular basic fuzzy implications used in forward/backward fuzzy reasoning are presented in [15] and [13], respectively.

As a simple consequence of the above theorem is the following fact.
Conclusion. The above theorem is false for \( y = 1 \).

Proof: From the property of a fuzzy implication presented in Remark 1 (item 1) we have 
\[ I(x, 1) = 1 \text{ for any } x \in [0, 1]. \] 
It means that for any two fuzzy implications \( I \) and \( J \) the following double dependency 
\[ I(x_1, 1) = J(x_2, 1) = 1 \text{ is true for any } x_1, x_2 \in [0, 1]. \]
Hence, we get that this equality is true not only for \( x_1 < x_2 \).

5 Algorithms

In this section, we present two algorithms formulated on the basis of the theorem from Sect. 4. The first algorithm allows to select the suitable (worse) implication function (see the condition 1, Sect. 3) based on information concerning a given set of fuzzy implications, their truth-values, and the truth value of conclusion. The second one allows to select the optimal fuzzy implication (see the condition 2, Sect. 3) using the same information as for the first algorithm.

Let \((FI, <)\) be a lattice of all fuzzy implications, a finite set \( U \subset FI \), and \( J \in U \).

Algorithm 1 finding a worse implication \( I \in U \) (in the sense of the condition 1, Sect. 3).

Input: \( U \) - a given finite set of fuzzy implications, \( J \in U \), \( y \in [0, 1] \), and \( k \in [0, 1] \) - a truth-value of \( J \).

Output: A worse implication \( I \in U \) than \( J \).

1. Order the set \( U \) with respect to the relation \(<\).
2. Identify the implication \( J \in U \).
3. if there exists an implication \( I \in U \) such that \( I < J \) 
   then
   Compute a value \( x_1 \) from the dependency \( I(x_1, y) = k \).
   Return \( x_1 \).
   else Stop.

Remark 5. The correctness of the Algorithm 1 follows immediately from the theorem presented in Sect. 4.

Example 3. Consider a set of fuzzy implications \( U = \{I_{LK}, I_{RC}, I_{KD}, I_{WB}\} \), the Łukasiewicz implication \( I_{LK} \), a given argument \( y = a \) (\( a < 1 \)), and the truth-value of \( I_{LK} = b \) (\( b > a \)). After executing the first step of the Algorithm 1 we obtain only one maximal chain \( c: I_{KD} < I_{RC} < I_{LK} < I_{WB} \) (see Example 2, item 1). Let us observe that the Łukasiewicz implication \( I_{LK} \) belongs to the chain \( c \). Moreover, it is easy to verify that there are two other implications less than \( I_{LK} \) with respect to the relation \(<\) in this chain, i.e., the Reichenbach implication \( I_{RC} \) and the Kleene-Dienes implication \( I_{KD} \). If, for example, we select the implication \( I_{RC} \), then from the dependency \( I_{RC}(x_1, a) = b \) we can compute a value \( x_1 = \frac{b - 1}{a - 1} \). Whereas a value \( x \) computed for the dependency \( I_{LK}(x, a) = b \) equals \( a - b + 1 \). It is easy to see that \( x_1 < x \).

Algorithm 2 finding an optimal implication in \( U \) (in the sense of the condition 2, Sect. 3).
Input: $U$ - a given finite set of fuzzy implications, $J \in U$, $y \in [0, 1)$, and $k \in [0, 1]$ - a truth-value of $J$.

Output: An optimal implication $I_{\text{opt}} \in U$ and a value $x_{\text{opt}}$.

1. Order the set $U$ with respect to the relation $\prec$.
2. Compute a set $C$ of all maximal chains in $U$ such that $J$ belongs to each of them.
3. for each chain $c \in C$ do find (if there exists) the least implication $I_c \prec J$.
   for each implication $I_c$ do compute a value $x_c$ (if there exists) from the dependency $I_c(x_c, y) = k$.
4. Compute a value $x_{\text{opt}} = \min\{x_c : c \in C\}$.
5. Return $(I_{\text{opt}}, x_{\text{opt}})$.

Remark 6. The correctness of the Algorithm 2 follows from the theorem (see Sect. 4) and the finiteness of set $U$.

Example 4. Now consider a set of fuzzy implications $U' = \{I_{LK}, I_{RC}, I_{KD}, I_{WB}, I_{YG}\}$, the Łukasiewicz implication $I_{LK}$, a given argument $y = a$ ($a < 1$), and the truth-value of $I_{LK} = b$ ($b > a$). After executing the steps 1 and 2 of the Algorithm 2 we obtain two maximal chains as follows: $c_1 = I_{KD} < I_{RC} < I_{LK} < I_{WB}$ and $c_2 = I_{YG} < I_{RC} < I_{LK} < I_{WB}$ (see Example 2, items 1 and 3). We can identify the Łukasiewicz implication in these two chains. Moreover, it is easy to check that $I_{KD}$ is the least implication in the chain $c_1$ with respect to the relation $\prec$, while $I_{YG}$ is the least implication in the chain $c_2$. Next, solving the equations $I_{KD}(x_{c_1}, a) = b$ and $I_{YG}(x_{c_2}, a) = b$, we obtain $x_{c_1} = 1 - b$ for $b > a$, and $x_{c_2} = \log_a b$ for $0 < a < b < 1$. Hence, we have $I_{\text{opt}} = I_{YG}$ and $x_{\text{opt}} = x_{c_2}$.

6 Illustrating Example

In order to illustrate our methodology, let us describe a simple example coming from the domain of train traffic control. We consider the following situation: a train $B$ waits at a certain station for a train $A$ to arrive in order to allow some passengers to change train $A$ to train $B$. Now, a conflict arises when the train $A$ is late. In this situation, the following alternatives can be taken into account:

- train $B$ departs in time, and an additional train is employed for the train $A$ passengers;
- train $B$ departs in time. In this case, passengers disembarking train $A$ have to wait for a later train;
- train $B$ waits for train $A$ to arrive. In this case, train $B$ will depart with delay.

In order to describe the traffic conflict, we propose to consider the following four IF-THEN fuzzy rules:

- $r_1$: IF $s_2$ THEN $s_6$ (CF = 0.6)
- $r_2$: IF $s_3$ THEN $s_6$ (CF = 0.6)
- $r_3$: IF $s_1$ AND $s_4$ AND $s_6$ THEN $s_7$ (CF = 0.5)
- $r_4$: IF $s_4$ AND $s_5$ THEN $s_8$ (CF = 0.8)
where:

- $s_1$: 'Train $B$ is the last train in this direction today',
- $s_2$: 'The delay of train $A$ is huge',
- $s_3$: 'There is an urgent need for the track of train $B$',
- $s_4$: 'Many passengers would like to change for train $B$',
- $s_5$: 'The delay of train $A$ is short',
- $s_6$: '(Let) train $B$ depart according to schedule',
- $s_7$: 'Employ an additional train $C$ (in the same direction as train $B$)',
- $s_8$: 'Let train $B$ wait for train $A$'.

In the further considerations we accept the following assumptions:

- the logical operator AND we interpret as min fuzzy operator;
- to the statements $s_7$ and $s_8$ we assign the fuzzy values 0.6 and 0.4, respectively;
- each of rules $r_1$, $r_2$, $r_3$, and $r_4$ we interpret firstly as the Łukasiewicz implication;
- the truth degrees of rules $r_1$, $r_2$, $r_3$, and $r_4$ are equal to 0.6, 0.6, 0.5, 0.8, respectively.

Assume that the user wants, for example, to know for which the truth degree of statements $s_4$ and $s_5$ the truth degree of the statement $s_8$ (i.e., the conclusion of the rule $r_4$) is equal to 0.6. Observe that in this situation the rule $r_4$ can be considered. Taking into account the dependency $I_{LK}(x, a) = b$ from Example 3 with $a = 0.4$ (the truth degree of the statement $s_8$) and $b = 0.8$ (the truth degree of the rule $r_4$) we get the truth degree of statements $s_4$ and $s_5$ equal to $x = a - b + 1 = 0.6$. However, if we interpret these four rules as the Reichenbach implications ($I_{RC}(x_1, a) = b$), and if we choose the same rule as above we obtain the truth degree of the statements $s_4$ and $s_5$ equal to $x_1 = \frac{b - 1}{a} \approx 0.33$. At last, if we execute the similar simulation of backward fuzzy reasoning for the rule $r_4$ considered above and, if we interpret these rules as the Kleene-Dienes implications we obtain the truth degree of the statements $s_4$ and $s_5$ equal to 0.2. Hence, we have $I_{opt} = I_{KD}$ for considered three interpretations of the rule $r_4$, and $x_{opt} = 0.2$. In analogous way one can analyze the situation in which the user wants to know the truth degree of the statements $s_1$, $s_2$, $s_3$, $s_6$ knowing the truth degree of the statement $s_7$.

This example shows clearly that different interpretations for the rules may lead to quite different truth degree of starting statements (corresponding to premises of given production rules). Choosing a suitable interpretation for fuzzy implications we may apply the theorem and the two algorithms presented in Sects. 4 and 5, respectively. The rest in this case certainly depends on the experience of the decision support system designer to a significant degree.

7 Concluding Remarks

In the paper, we have presented a methodology for selecting relevant fuzzy implication in backward reasoning, which has for example the least truth value of the premise when the truth value of the conclusion and the truth value of the implication are given. This
methodology takes full advantage of the functional representation of fuzzy implications and the algebraic properties of the family of all fuzzy implications.

We know that there are a lot of implication functions in the field of fuzzy logic, and the nature of the inference changes variously depending on the implication function to be used. However, it is very difficult to select a suitable implication function for actual applications. But taking into account the methodology proposed in this paper we can reduce the efforts related to a selection of a suitable implication function.

Acknowledgments. This work was partially supported by the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszów. The authors are also very grateful to the anonymous reviewer for giving him precious and helpful comments.

References

Inverted Fuzzy Implications in Backward Reasoning Without Yager Implication

Zbigniew Suraj\textsuperscript{1} and Agnieszka Lasek\textsuperscript{2}

\textsuperscript{1} Chair of Computer Science, 
Faculty of Mathematics and Natural Sciences, University of Rzeszow, 
Prof. S. Pigonia Str. 1, 35-310 Rzeszow, Poland 
zbigniew.suraj@ur.edu.pl

\textsuperscript{2} Department of Electrical Engineering and Computer Science 
Lassonde School of Engineering, York University, 
4700 Keele Street, Toronto, Canada 
alasek@cse.yorku.ca

\textbf{Abstract.} One of the most popular methods of knowledge representation are the fuzzy rules. One of the ways of representation of fuzzy rules is the functional representation. From over eight decades a number of different fuzzy implications have been described, e.g. [5]-[9]. This leads to the following question: how to choose the proper function among basic fuzzy implications. This paper is a continuation of study [15], where we proposed a new method for choosing implications in backward reasoning. Here we presented a way of simplify the analysis by skipping Yager fuzzy implication.

\textbf{Key words:} fuzzy logic, fuzzy implications, inverted fuzzy implications, backward reasoning

1 Introduction

One of the most popular methods of knowledge representation are the fuzzy rules. From imprecise inputs and fuzzy rules imprecise conclusions are obtained. Reasoning is mainly classified into two types: forward reasoning and backward reasoning. The inference mechanism of forward reasoning has a strong forecasting capability, whereas the aim of backward reasoning generally is to find the most possible causes associated with the existing reality. Backward reasoning plays an essential role in fault diagnosis, accident analysis, and defect detection. This kind of reasoning uses fuzzy logic [3] to reason about data in the inference mechanism instead of many other logics, including Boolean logic, (non-fuzzy) many-valued logics, non-monotonic logics, etc.

Paper [4] discusses different representations of rules in a non-fuzzy setting and extends these representations to rules with a fuzzy conclusion part. It introduces the different types of fuzzy rules and put them in the framework of fuzzy sets and possibility theory.

Fuzzy rules are often presented in the form of implications. In [3] a typology of fuzzy rules and the problem of multiple-valued implications are discussed. The paper
reviews the problem of representing fuzzy knowledge, and ranges from linguistic variables to conditional if-then rules and qualified statements.

One of the ways of representation of fuzzy rules is the functional representation (e.g., [11], [12], [17]). The definition of fuzzy implications and their mathematical properties can be found e.g. in [1] and [16]. One of basic problems in building an inference system is choosing the relevant fuzzy implication. In [10] authors proposed a method allowing to choose the most suitable fuzzy implication in an inference system application. They introduced an algorithm that calculates the distance between two fuzzy implications and which is based on generalized modus ponens.

In paper [13] we have presented a fuzzy forward reasoning methodology for rule-based systems using the functional representation of fuzzy rules. In [15] we extended this methodology for selecting relevant fuzzy implications for backward reasoning. The proposed methodology takes full advantage of the functional representation of fuzzy implications and the algebraic properties of the family of all fuzzy implications. It allows to compare two fuzzy implications. If the truth value of the conclusion and the truth value of the implication are given, we can easily optimize the truth value of the implication premise. In particular, in [15] we introduced an algorithm of finding the fuzzy implication which has the highest truth value of the antecedent when the truth value of the consequent and the truth value of the implication are given. This methodology can be useful for the design of inference engine based on the rule knowledge for a given rule-based system.

In the solution in [15] we divided the domain of fuzzy implications into areas, in which it was possible to select appropriate fuzzy implication, and to do that we had to use the Lambert W function. Lambert W function is a special function used when solving equations containing unknown to both the base and the exponent power. It is marked \( W(z) \) and defined as the inverse of \( f(z) = ze^z \), where \( z \) belongs to the set of complex numbers. Thus, for each complex number \( z \) holds: \( z = W(z)e^{W(z)} \). The Lambert W function cannot be expressed in terms of elementary functions.

In this paper we present the way of avoiding this complexity of solution presented in [15].

The rest of this paper is organized as follows. Sect. 2 contains basic information on fuzzy implications. In Sect. 3 the research problem is formulated. Sect. 4 presents the solution of the given research problem. Sect. 5 is devoted to the pseudo-code of an algorithm for determining a basic fuzzy implication which has the highest truth value of the antecedent when the truth value of the consequent and the truth value of the implication are given. Sect. 6 includes summarizing of our research and some remarks.

## 2 Preliminaries

In this section we recall a definition of a fuzzy implication and we list a few of basic fuzzy implications known from the subject literature [1].

A function \( I: [0,1]^2 \rightarrow [0,1] \) is called a fuzzy implication if it satisfies, for all \( x, x_1, x_2, y, y_1, y_2 \in [0,1] \), the following conditions:

- if \( x_1 \leq x_2 \), then \( I(x_1, y) \geq I(x_2, y) \), i.e., \( I(., y) \) is decreasing;
Table 1. Examples of basic fuzzy implications

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Formula of basic fuzzy implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Łukasiewicz</td>
<td>1923, [9]</td>
<td>$I_{LK}(x,y) = \min(1, 1 - x + y)$</td>
</tr>
<tr>
<td>Gödel</td>
<td>1932, [4]</td>
<td>$I_{GD}(x,y) = \begin{cases} 1 &amp; \text{if } x \leq y \ y &amp; \text{if } x &gt; y \end{cases}$</td>
</tr>
<tr>
<td>Kleene-Dienes</td>
<td>1938, [8]; 1949, [2]</td>
<td>$I_{KD}(x,y) = \max(1 - x, y)$</td>
</tr>
<tr>
<td>Goguen</td>
<td>1969, [7]</td>
<td>$I_{GG}(x,y) = \begin{cases} 1 &amp; \text{if } x \leq y \ y &amp; \text{if } x &gt; y \end{cases}$</td>
</tr>
<tr>
<td>Rescher</td>
<td>1969, [12]</td>
<td>$I_{RS}(x,y) = \begin{cases} 1 &amp; \text{if } x \leq y \ 0 &amp; \text{if } x &gt; y \end{cases}$</td>
</tr>
<tr>
<td>Yager</td>
<td>1980, [18]</td>
<td>$I_{YG}(x,y) = \begin{cases} 1 &amp; \text{if } x = 0 \text{ and } y = 0 \ x &amp; \text{if } x &gt; 0 \text{ or } y &gt; 0 \end{cases}$</td>
</tr>
<tr>
<td>Weber</td>
<td>1983, [17]</td>
<td>$I_{WB}(x,y) = \begin{cases} 1 &amp; \text{if } x &lt; 1 \ y &amp; \text{if } x = 1 \end{cases}$</td>
</tr>
<tr>
<td>Fodor</td>
<td>1993, [3]</td>
<td>$I_{FD}(x,y) = \begin{cases} 1 &amp; \text{if } x \leq y \ \max(1 - x, y) &amp; \text{if } x &gt; y \end{cases}$</td>
</tr>
</tbody>
</table>

There exist uncountably many fuzzy implications. The following Table 1 contains a few examples of basic fuzzy implications. One of the fuzzy implication in the table is Yager implication. As we noted in Sect. 1 we skip this implication in our analysis in this paper to avoid complexity of solution presented in [15].

Figure 1 gives us some plots of these functions.

3 Problem Statement

Our goal is to design an algorithm to find a method of selecting fuzzy implication in view of the value of the implication antecedent.

Assume that there is given a basic fuzzy implication $z = I(x, y)$, where $x, y$ belong to $[0,1]$. $y$ is the truth value of the consequent and is known. $z$ is the truth value of the implication and is also known. In order to determine the value of the truth of the implication antecedent $x$ it is needed to compute the inverse function $\text{InvI}(y,z)$. In other words, the inverse function $\text{InvI}(y,z)$ has to be determined. Not every of basic implications can be inverted. The function can be inverted only when it is injective.

4 Results

Table 2 lists inverse fuzzy implications and their domains and in Figure 2 there are some plots of them.
Fig. 1. Plots of \( I_{LK} \) and \( I_{KD} \) fuzzy implications

<table>
<thead>
<tr>
<th>Formula of inverted fuzzy implication</th>
<th>Domain of inverted fuzzy implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Inv} I_{LK}(y, z) = 1 - z + y )</td>
<td>( y \leq z &lt; 1, y \in [0, 1) )</td>
</tr>
<tr>
<td>( \text{Inv} I_{RC}(y, z) = \frac{1 - z}{1 - y} )</td>
<td>( y \leq z \leq 1, y \in [0, 1) )</td>
</tr>
<tr>
<td>( \text{Inv} I_{KD}(y, z) = 1 - z )</td>
<td>( y &lt; z \leq 1, y \in [0, 1) )</td>
</tr>
<tr>
<td>( \text{Inv} I_{GG}(y, z) = \frac{y}{z} )</td>
<td>( y \leq z &lt; 1, y \in (0, 1) )</td>
</tr>
<tr>
<td>( \text{Inv} I_{FD}(y, z) = 1 - z )</td>
<td>( y &lt; z &lt; 1 - y, y \in [0, 1) )</td>
</tr>
</tbody>
</table>

Table 2. Inverted fuzzy implications

Fig. 2. Plots of \( \text{Inv} I_{LK} \) and \( \text{Inv} I_{KD} \) fuzzy implications
The domains of every considered inverted fuzzy implications are included in a half of the unit square, where \( y \leq z < 1 \) and \( y \in (0,1) \). Only one inverted fuzzy implication has a domain which is smaller than this area. This is inverted Fodor implication and in the whole its domain ( \( y \leq z < 1 - y, y \in [0,1]\) ) this function is equal to inverted Kleene-Dienes implication.

For \( y \leq z < 1 - y \) there are the following inequalities: \( \text{InvI}_{FD} = \text{InvI}_{KD} < \text{InvI}_{RC} < \text{InvI}_{LK}, \text{InvI}_{RC} < \text{InvI}_{LK}, \text{InvI}_{GG} < \text{InvI}_{LK} \). A graphical representation of the ordering of inverted basic fuzzy implications is given in Figure 3.

\[
\begin{align*}
\text{InvI}_{LK} & \quad \text{InvI}_{RC} \quad \text{InvI}_{GG} \\
\text{InvI}_{KD} & = \quad \text{InvI}_{FD}
\end{align*}
\]

**Fig. 3.** A graphical representation of the ordering of inverted basic fuzzy implications for \( y \leq z < 1 - y \)

For \( 1 - y \leq z < 1 \) and \( y \leq z \) there are the same inequalities, but without inverted Fodor implication, because this function does not exist in this area.

The resulting inverse functions can be compared with each other so that it is possible to order them. However, some of those functions are incomparable in the whole domain. By taking into account six inverted fuzzy implications (including inverted Yager implication) and by dividing their domain into separable areas, we obtained 19 inequalities between inverted fuzzy implications for any \( y \leq z < 1 \) and \( y \in (0,1) \) described in [15].

To simplify that solution and avoid Lambert W function in this paper we skip Yager fuzzy implication in our analysis. With this assumption there is only five different area and inequalities instead of nineteen. The areas are shown in the Figure 4 and the inequalities are given in Table 3.
Fig. 4. The unit square $[0, 1]^2$ divided into five separable areas

Table 3: Table of inequalities

<table>
<thead>
<tr>
<th>No</th>
<th>Area and inequality</th>
<th>Chart of area</th>
<th>Graph of inequalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>For $z &gt; 1 - y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{InvI}<em>{KD} &lt; \text{InvI}</em>{RC} &lt; \text{InvI}<em>{GG} &lt; \text{InvI}</em>{LK}$</td>
<td><img src="chartA.png" alt="Chart of area A" /></td>
<td><img src="graphA.png" alt="Graph of inequalities A" /></td>
</tr>
<tr>
<td>B.</td>
<td>For $z = 1 - y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{InvI}<em>{KD} &lt; \text{InvI}</em>{RC} = \sqrt{\frac{1}{4} - 4y}$</td>
<td><img src="chartB.png" alt="Chart of area B" /></td>
<td><img src="graphB.png" alt="Graph of inequalities B" /></td>
</tr>
</tbody>
</table>

Continued on next page
### Table 3 – Continued from previous page

<table>
<thead>
<tr>
<th>No</th>
<th>Area and inequality</th>
<th>Chart of area</th>
<th>Graph of inequalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.</td>
<td>For ( z &gt; \frac{1+\sqrt{1-4y}}{2} ) or ( z &lt; \frac{1-\sqrt{1-4y}}{2} ) \ or ( z \in (0.25, 0.5) ) \ and ( z &lt; 1 - y )</td>
<td>![Diagram C]</td>
<td>![Graph C]</td>
</tr>
<tr>
<td>( InvI_{KD} &lt; InvI_{GG} ) ( InvI_{RC} &lt; InvI_{LK} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.</td>
<td>For ( z = \frac{1+\sqrt{1-4y}}{2} ) or ( z = \frac{1-\sqrt{1-4y}}{2} ) ( InvI_{GG} = InvI_{KD} ) ( InvI_{RC} &lt; InvI_{LK} )</td>
<td>![Diagram D]</td>
<td>![Graph D]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E.</td>
<td>For ( z &gt; \frac{1+\sqrt{1-4y}}{2} ) and ( z &lt; \frac{1-\sqrt{1-4y}}{2} ) ( InvI_{GG} &lt; InvI_{KD} ) ( InvI_{RC} &lt; InvI_{LK} )</td>
<td>![Diagram E]</td>
<td>![Graph E]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All inequalities given in Table 3 can be proven in a similar way. As examples, we will consider one of inequalities. Let \( y \in (0, 1) \) and \( z \in (y, 1) \). \( y < z \), so obviously \( y^2 < yz \). By adding and subtracting \( 1 - z + y \) to the equation we obtained \( 1 - z < 1 - z + y - y + yz - y^2 \). And therefore, \( \frac{1-z}{1-y} < 1 - z + y \). This completes the proof of the inequality: \( InvI_{RC} < InvI_{LK} \) in domains of these functions.

### 5 Algorithm

Below we present the pseudo-code of the algorithm (DetermineImplicationGTVA) for determining a basic fuzzy implication which has the highest truth value of the an-
tecedent whereas the truth value of the consequent and the truth value of the implication are given.

The algorithm uses the results of our research presented in Table 3. The first step in the algorithm determines to which area \((A) - (E)\) from Table 3 point \((y, z)\) belongs to.

Algorithm \textit{DetermineImplicationGTVA}

\begin{itemize}
  \item \textbf{Input:} \(W\) - a given subset of the basic fuzzy implications;
  \(y\) - the truth value of the consequent;
  \(z\) - the truth value of the implication
  \item \textbf{Output:} \(I \in W\) - fuzzy implication(s) which has (have) the highest truth value of the antecedent
\end{itemize}

\begin{enumerate}
  \item \(a \leftarrow \text{area}(y, z)\) //determines the area from \((A) - (E)\) to which a point \((y, z)\) belongs to;
  \item \textbf{order} the set \(W\) with respect to the graph \(G_a\) of inequalities from the area \(a\);
  \item \(I \leftarrow \text{the maximal element(s)}\) from the ordered set \(W\);
  \item \textbf{return} \(I\);
\end{enumerate}

6 Concluding Remarks

In the paper, we introduced an algorithm for finding the fuzzy implication which has the highest truth value of the antecedent from a given subset of the basic fuzzy implications, when the truth value of the consequent and the truth value of the implication are given. In order to simplify the solution we skipped Yager fuzzy implication in the presented analysis.

We considered a set of basic implications mentioned in Table 1, because they are well known and widely used. But considering only these basic implications implied the solution which does not cover the whole unit square as in the case with the forward reasoning [13], only one of its halves. It raises the question how to find such a set of implications that could give a solution for a backward reasoning in the whole unit square. Our future works will focus on answering the question whether such implications could exist and how they could be defined.

Acknowledgments. This work was partially supported by the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszów. The authors are also very grateful to the anonymous reviewer for giving him precious and helpful comments.

References

SMT-based Searching for $k$-quasi-optimal Runs in Weighted Timed Automata

Extended Abstract

Bozena Wozna-Szczesniak, Agnieszka M. Zbrzezny, and Andrzej Zbrzezny

IMCS, Jan Długosz University Al. Armii Krajowej 13/15, 42-200 Czestochowa, Poland
{b.wozna,agnieszka.zbrzezny,a.zbrzezny}@ajd.czest.pl

Abstract. We investigate an optimal cost reachability problem for weighted timed automata, and we use a translation to SMT to solve the problem. In particular, we show how to find a run of length $k \in \mathbb{N}$ that starts at the initial state and terminates at a state containing the target location, its total cost belongs to the interval $[c, c+1)$, for some natural number $c \in \mathbb{N}$, and the cost of each other run of length $k$, which also leads from the initial state to a state containing the target location, is greater or equal to $c$. This kind of runs we call $k$-quasi-optimal.

We exemplify the use of our solution to the mentioned problem by means of the weighted timed generic pipeline protocol, and we provide some preliminary experimental results.

1 Introduction

Reachability is a core decision problem that appears in several different contexts, for example: concurrent systems [8, 17], time critical systems [12], and probabilistic systems [10]. For computational models of time critical systems like weighted timed automata [2] (see Section 2 for the formal definition), or priced timed automata [5], it is reasonable to make inquiries about the minimum (optimal) cost of reaching a desirable state of the system, i.e., to investigate the optimal reachability problem.

The optimal reachability problem was considered by numerous researchers and several methodologies treating the issue in the setting of timed automata have been described in the literature, but none of them used SMT- or SAT-based methods (see the Related Work section). The acronym SMT means satisfiability modulo theories. SMT-solvers are tools for deciding the satisfiability of formulae in a number of theories [4]. Nevertheless, in the paper [19] we made the first attempt to solve, so called, the $k$-optimal cost reachability problem for weighted timed automata (see Section 3 for the formal definition). The proposed solution used the translation to SAT, and it could only be applied to systems modelled by a single weighted timed automaton.

In this paper we also deal with the $k$-optimal cost reachability problem, but for time critical systems modelled by a network of weighted timed automata. Moreover, we are interested in using SMT-based verification methods to solve the problem instead of the SAT-based verification. The use of a translation to SMT allows us to avoid an intermediate discretised model.

* Partly supported by National Science Centre under the grant No. 2014/15/N/ST6/05079.
The solution which we propose for the $k$-optimal cost reachability problem is the combination of a well-known forward reachability algorithm with the bounded model checking (BMC) method \cite{18, 19} that uses SMT solvers instead of SAT solvers (see Section 3 for the informal and formal description of the solution). The forward reachability algorithm searches the state space using the breadth first mode, whereas the BMC performs a verification on a part of the automata model exploiting SMT solvers.

How our solution to the $k$-optimal cost reachability problem is effective we show by means of the weighted timed generic pipeline protocol (see Section 4).

## 2 Preliminaries

Let us start by introducing the key sets of numbers and variables used in the rest of the paper. The first is the set $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ of natural numbers. The second is the set $\mathbb{R}$ of non-negative real numbers, and the third is the set $\mathbb{R}_+$ of positive real numbers. The next is the set $\mathcal{P}$ of propositional variables, and the final is a finite set $\mathcal{X}$ of real variables, called clocks.

For the set $\mathcal{X}$ of clocks, $x, y \in \mathcal{X}$, $c \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$, the set $\mathcal{C}(\mathcal{X})$ of all the clock constraints over $\mathcal{X}$ is defined by the following grammar:

$$ \mathcal{cc} ::= true \mid x \sim c \mid x - y \sim c \mid \mathcal{cc} \land \mathcal{cc} $$

Furthermore, a clock valuation is a total mapping $\mathbf{cv} : \mathcal{X} \to \mathbb{R}$, and satisfiability of a clock constraint $\mathcal{cc} \in \mathcal{C}(\mathcal{X})$ by a clock valuation $\mathbf{cv}$ ($\mathbf{cv} \models \mathcal{cc}$) is defined inductively as follows:

- $\mathbf{cv} \models true$,
- $\mathbf{cv} \models (x \sim c)$ iff $\mathbf{cv}(x) \models c$,
- $\mathbf{cv} \models (x - y \sim c)$ iff $\mathbf{cv}(x) - \mathbf{cv}(y) \models c$,
- $\mathbf{cv} \models \mathcal{cc}_1 \land \mathcal{cc}_2$ iff $\mathbf{cv} \models \mathcal{cc}_1$ and $\mathbf{cv} \models \mathcal{cc}_2$.

Given a clock valuation $\mathbf{cv}$ and $\delta \in \mathbb{R}_+$, by $\mathbf{cv} + \delta$ we denote a clock valuation $\mathbf{cv}'$ such that $\mathbf{cv}'(x) = \mathbf{cv}(x) + \delta$, for all $x \in \mathcal{X}$. Moreover, for a subset of clocks $X \subseteq \mathcal{X}$, $\mathbf{cv}[X := 0]$ denotes the valuation $\mathbf{cv}'$ such that for all $x \in X$, $\mathbf{cv}'(x) = 0$ and for all $x \in \mathcal{X} \setminus X$, $\mathbf{cv}'(x) = \mathbf{cv}(x)$. Finally, by $\mathbf{cv}^0$ we denote the initial clock valuation, i.e., the valuation such that $\mathbf{cv}^0(x) = 0$ for all $x \in \mathcal{X}$.

**Definition 1.** A weighted timed automaton is a tuple $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$, where $\Sigma$ is a finite set of actions, $L$ is a finite set of locations, $l^0 \in L$ is an initial location, $\mathcal{X}$ is a finite set of clocks, $E \subseteq L \times (\Sigma \times \mathcal{C}(\mathcal{X}) \times \mathbb{R} \times \mathcal{X}) \times L$ is a transition relation, $\mathcal{I} : L \mapsto \mathcal{C}(\mathcal{X})$ is an invariant function, $J_s : \Sigma \mapsto \mathbb{N}$ is a switch cost function, $J_d : L \mapsto \mathbb{N}$ is a duration cost function, $z$ is a real variable, and $\mathcal{V} : L \mapsto 2^{\mathcal{P}}$ is a valuation function assigning to each location a set of propositional variables true in that location.

The switch cost function assigns to each action a cost expressing the price of taking the action. The duration cost function assigns to each location a cost expressing the price of staying in this location for one time unit. The invariant function assigns to each location a clock constraint expressing the condition under which $\mathcal{A}$ can stay in this location.
Each element \( t = (l, \sigma, cc, X, l') \in E \) represents a transition from the location \( l \) to the location \( l' \), where \( \sigma \) is the action of the transition \( t \), \( cc \) defines the enabling conditions for \( t \), and \( X \) is a set of clocks to be reset.

Note that the syntax of weighted timed automata is borrowed from [2, 19], but it is extended to a special real variable \( z \). Moreover, we have changed the domain of the switch cost function. Namely, we assume that the domain of the switch cost function is the set of actions instead of the set of transitions. Such a change is necessary, since we considered systems that are modelled through a network of weighted timed automata instead of a single weighted timed automaton.

In general, weighted timed automata are often composed into a network of weighted timed automata consisting of \( n \) weighted timed automata \( A_i = (\Sigma_i, L_i, l_0^i, X_i, E_i, I_i, J_i, J_d, I, V, z) \), for \( i = 1, \ldots, n \), that run in parallel and communicate via synchronised actions. A formal definition of a parallel composition of the weighted timed automata is the following.

**Definition 2.** Let \( \Sigma = \bigcup_{i=1}^n \Sigma_i \), \( \sigma \in \Sigma \), and \( \Sigma(\sigma) = \{ 1 \leq i \leq n \mid \sigma \in \Sigma_i \} \) be the set of indexes of automata that synchronise at \( \sigma \), and \( z_1 = \ldots = z_n \) (i.e., all the automata have to share the variable). For \( i = 1, \ldots, n \), a parallel composition of weighted timed automata \( A_i \) is the weighted timed automaton \( A = (\Sigma, L, l^0, X, E, I, J, J_d, z, V) \), where \( \Sigma = \bigcup_{i=1}^n \Sigma_i \), \( L = \prod_{i=1}^n L_i \), \( l^0 = (l_1^0, \ldots, l_n^0) \), \( X = \bigcup_{i=1}^n X_i \), a transition \((l_1, \ldots, l_n), \sigma, l' \in \Sigma(\sigma) \in E \) iff \((\forall j \in \Sigma(\sigma)) \in E_j\), and \((\forall i \in \{1, \ldots, n\} \setminus \Sigma(\sigma)) \in E_i\) \( l'_i = l_i \), \( I(l_1, \ldots, l_n) = \bigwedge_{i=1}^n I_i(l_i) \), \( J_i(\sigma) = \sum_{j \in \Sigma(\sigma)} J^i_j(\sigma) \), \( J_d(l_1, \ldots, l_n) = \sum_{i=1}^n J^i_d(l_i) \), \( z = z_1 \), and \( V(l_1, \ldots, l_n) = \bigcup_{i=1}^n V_i(l_i) \).

The semantics of weighted timed automata is defined by associating to them dense models as defined below.

**Definition 3.** Let \( \mathcal{A} = (\Sigma, L, l^0, X, E, I, J, J_d, z, V) \) be a weighted timed automaton, \( z : \{ z \} \to \mathbb{R} \) a valuation for \( z \), and \( z^0 \) the initial valuation for \( z \) (i.e., \( z^0(z) = 0 \)). A dense model for \( \mathcal{A} \) is the tuple \( M = (\Sigma \cup \mathbb{R}_+, S, s^0, \to, V') \), where \( \Sigma \cup \mathbb{R}_+ \) is a set of labels, \( S = \{(l, cv, z) \mid l \in L, \ cv \in \mathbb{R}^{|X|}, \ cv \models I(l), z \in \mathbb{R} \} \) is a set of states, \( s^0 = (l^0, cv^0, z^0) \) is the initial state, \( V' : S \to 2^{PV} \) is a valuation function such that \( V'(l, cv, z) = V(l) \), and \( \to \subseteq S \times \Sigma \cup \mathbb{R} \times S \) is the smallest transition relation defined by the following two rules:

- **action transition:** for \( \sigma \in \Sigma \), \( (l, cv, z) \xrightarrow{\sigma} (l', cv', z') \) iff there exists a transition \( i = (l, \sigma, cc, X, l') \in E \) such that \( cv \models cc \), \( cv \models I(l) \), \( cv[X := 0] \models I(l') \), and \( z' = z + J_i(\sigma) \).

- **time transition:** for \( \delta \in \mathbb{R}_+ \), \( (l, cv, z) \xrightarrow{\delta} (l, cv + \delta, z') \) iff \( cv \models I(l) \), \( cv + \delta \models I(l) \), and \( z' = z + J_d(l) \cdot \delta \).

Intuitively, an action transition corresponds to an action performed by the automaton under consideration. The action can be performed only if the underlying enabling condition is satisfied. Moreover, all the clocks that are associated with the action are set to zero, its locations change accordingly, and the value of the variable \( z \) is increased by the switch cost. A time transition causes an equal increase in the value of all the clocks,
and does not involve a location change. Obviously, the new clock valuations have to still satisfy all the location invariants, and the value of the variable $z$ is increased by the duration cost.

Let us denote by $s \xrightarrow{\delta, \sigma} s'$ the sequence of the following time and action transitions:

$s \xrightarrow{\delta} s''$ and $s'' \xrightarrow{\sigma} s'$, where $\sigma \in \Sigma$, $\delta \in \mathbb{R}_+$, and $s, s', s'' \in S$. 

**Definition 4.** Let $k \in \mathbb{N}$, $i \in \{0, \ldots, k\}$, $s_i \in S$, $j \in \{1, \ldots, k\}$, $\sigma_j \in \Sigma$, and $\delta_j \in \mathbb{R}_+$. A $k$-run $\rho$ of a weighted timed automaton $\mathcal{A}$ is a finite sequence of transitions:

$s_0 \xrightarrow{\delta_1, \sigma_1} s_1 \xrightarrow{\delta_2, \sigma_2} \ldots xrightarrow{\delta_{k-1}, \sigma_{k-1}} s_{k-1} \xrightarrow{\delta_k, \sigma_k} s_k$. 

In other words, a $k$-run is a finite path of $\mathcal{A}$, where action transitions are taken finitely often and time transitions are aggregated. Moreover, the $k$-run does not permit two consecutive actions to be performed one after the other; such a run is called **strongly monotonic**.

Given a $k$-run $\rho$ of $\mathcal{A}$ and cost functions $J_s$ and $J_d$, we associate cost to $\rho$ as follows:

$J_s(\rho) = \sum_{i=1}^{k} J_s(\sigma_i)$, and $J_d(\rho) = \sum_{i=1}^{k} \delta_i \cdot J_d(l_{i-1})$. Furthermore, the **total cost** associated to a $k$-run $\rho$ is defined as $J(\rho) = J_d(\rho) + J_s(\rho)$. Finally, we recall the definitions of both the **$k$-optimal cost** and **$k$-quasi-optimal** that have been introduced in [19].

**Definition 5.** The **$k$-optimal cost** for $k$-runs that start at a state containing location $l$ and end at a state containing location $l'$ is defined as: $J_k^*(l, l') = \inf \{J(\rho) \mid \rho$ is a $k$-run from a state containing location $l$ to a state containing location $l'\}$.

If $[J(\rho)] = [J_k^*(l, l')]$, then a $k$-run $\rho$ from a state containing location $l$ to a state containing location $l'$ is $k$-quasi-optimal.

In this paper, for given two locations $l$ and $l'$ we are interested in finding the greatest integer lower bound (g.i.l.b. for short) of the $k$-optimal cost for $k$-runs starting at a state $s$ containing location $l$ and terminating at a state $t$ containing location $l'$, where $k$ is the length of a shortest run from $s$ to $t$. Moreover, we are interested in finding $k$-**quasi-optimal** runs. Therefore, in Section 3 we define $k$-optimal cost reachability problem, and we show how to solve it using SMT-methods.

### 3 **$k$-Optimal Cost Reachability Problem**

In this section we formally define the $k$-optimal cost reachability problem for weighted timed automata, and we present a solution to the problem which uses SMT-solvers. We begin with defining the problem, and then we describe our solution informally, which will help to understand the formal algorithm presented later on in this section.

**Definition 6** (**$k$-optimal cost reachability**). Given a weighted timed automaton $\mathcal{A} = (\Sigma, I, I^0, X, E, I, J_s, J_d, z, V)$, and a desirable location $l^p \in L$ satisfying a property $p$, the $k$-optimal cost reachability problem consists in finding out a $k$-quasi-optimal run $\rho$ starting at $s^0 \in M$ and terminating at a state in $M$ containing location $l^p$.

Note that if $\rho$ is a $k$-quasi-optimal run, then there exists $c \in \mathbb{N}$ such that: $c \leq J(\rho) < c + 1$, and for all the $k$-runs $\rho'$ that starts at $s^0$ and terminates at a state in $M$ containing location $l^p$, $J(\rho') \geq c$ holds.
An informal explanation. To solve the $k$-optimal cost reachability problem we proceed as follows. We first encode by quantifier-free first-order formulae with individual variables ranging over the real numbers both the property $p$, and the unfolding of the transition relation of $M$ up to depth $k$ (for $k \in \mathbb{N}$). Let $\varphi_k$ be the conjunction of the two above formulae. We test $\varphi_k$ for satisfiability using an SMT-solver. If the test for $\varphi_k$ is positive, we calculate the cost $r_0 \in \mathbb{R}$ of the resulting witness $\rho_0$, and we know that $J(\rho_0) < [r_0]$. Next, we set $c_0 = [r_0] - 1$, and we run the satisfiability test once again, but for the formula $\phi_k(c_0) = \varphi_k \land (z < c_0)$. If the test for $\phi_k(c_0)$ is positive, we calculate the cost $r_1 \in \mathbb{R}$ of the resulting witness $\rho_1$, and we know that $r_1 < c_0$. Next, we set $c_1 = [r_1] - 1$, and we run the satisfiability test once again, but for the formula $\phi_k(c_1) = \varphi_k \land (z < c_1)$, and so on. We stop testing, if the test for $\phi_k(c_i)$ is negative or $r_i = 0$.

Notice that, if the test for $\phi_k(c_i)$ is negative, we can perform one more test for the formula $\psi_k(c_i) = \varphi_k \land (z = c_i)$. If the test for $\psi_k(c_i)$ is positive, we can conclude that $k$-optimal cost is equal to $c_i$. Otherwise, we can only conclude that the g.i.l.b. of the $k$-optimal cost is equal to $c_i$.

A formal algorithm for finding the g.i.l.b. of $k$-optimal cost. Algorithm 1 uses the procedure $\text{checkSMT}(\gamma)$ that for any given quantifier-free first-order formula $\gamma$ returns a pair $(W, X)$, where $W$ denotes the valuation returned by a SMT solver, and $X$ can be one of the following three values: $\text{TRUE}$, $\text{FALSE}$, and $\text{UNKNOWN}$. The meanings of the values $\text{TRUE}$ and $\text{FALSE}$ are self-evident. The value $\text{UNKNOWN}$ is returned either if the procedure $\text{checkSMT}$ is not able to decide satisfiability of its argument within some preset timeout period, or has to terminate itself due to exhaustion of available memory. Algorithm 1 also uses the procedure $\text{getCOST}(W)$ that for the valuation $W$, which represents a $k$-run $\rho$, returns a natural number $c$ such that the cost of $\rho$ is less than $c$. Finally, for a given quantifier-free first-order formula $\varphi_k$, the symbol $\phi_k(c)$ denotes the formula $\varphi_k \land (z < c)$, and the symbol $\psi_k(c)$ denotes the formula $\varphi_k \land (z = c)$.

Translation to quantifier-free first-order formulae. Let $A = (\Sigma, L, l^0, X, E, I, J_s, J_d, z, \mathcal{V})$ be a weighted timed automaton that is a parallel composition of $n$ weighted timed automata $A_i$, $M = (\Sigma \cup \mathbb{R}, S, s^0, \rightarrow, \mathcal{V}')$ a dense model for $A$, and $k \in \mathbb{N}$. Each state $s \in S$ of $M$ can be represented by a valuation of a symbolic state $w = ((1, \ldots, 1_n), (\xi_1, \ldots, \xi_{|X'|}), (d_1, \ldots, d_n), z)$ that consists of symbolic local states, symbolic clock valuations, symbolic duration costs, and symbolic valuation of variable $z$. Each symbolic local state $1_i$ ($1 \leq i \leq n$) is an individual variable ranging over the natural numbers. Each symbolic clock valuation $\xi_i$ ($1 \leq i \leq |X'|$) is an individual variable ranging over the real numbers. Each symbolic duration cost $d_i$ ($1 \leq i \leq n$) is an individual variable ranging over the natural numbers, and symbolic valuation of variable $z$ is an individual variable ranging over the real numbers. Similarly, each action $\sigma \in \Sigma$ can be represented by a valuation of a symbolic action $a$ that is an individual variable ranging over the natural numbers, each real number $r \in \mathbb{R}$ can be represented by a valuation of a symbolic number $x$ that is an individual variable ranging over the real numbers, and finally each switch cost $s$ can be represented by a valuation of a symbolic switch cost $s$ that is an individual variable ranging over the natural numbers.

A finite sequence $(w_0, \ldots, w_k)$ of symbolic states is called a symbolic $k$-path.
Algorithm 1: An algorithm for finding g.i.l.b. of $k$-optimal cost

1: $k \leftarrow 0$
2: repeat
3:  $(W, result) \leftarrow \text{checkSMT}(\varphi_k)$
4:  if $result = \text{FALSE}$ then
5:    $k \leftarrow k + 2$
6:  else if $result = \text{UNKNOWN}$ then
7:    return $\text{UNKNOWN}$
8:  end if
9: until $result = \text{TRUE}$
   \{there exists a witness of the length $k$ for a desirable property\}
10: $c \leftarrow \text{getCOST}(W)$
11: repeat
12:  if $c = 0$ then
13:    return $k$-optimal cost is equal to 0
14:  end if
15:  $(W, result) \leftarrow \text{checkSMT}(\phi_k(c - 1))$
16:  if $result = \text{TRUE}$ then
17:    $c \leftarrow \text{getCOST}(W)$
18:  else if $result = \text{UNKNOWN}$ then
19:    return $\text{UNKNOWN}$
20:  end if
21: until $result = \text{FALSE}$
   \{optimal cost of any $k$-run is greater or equal to $c$\}
22: $(W, result) \leftarrow \text{checkSMT}(\psi_k(c))$
23: if $result = \text{TRUE}$ then
24:    return $k$-optimal cost is equal to $c$
25: else
26:    return g.i.l.b. of $k$-optimal cost is equal to $c$
27: end if

For two symbolic states $w, w'$, we define the following propositional formulae:

- $I_s(w)$ is a formula that encodes the state $s$ of $M$
- $p(w)$ is a formula that encodes the set of states of $M$ in which $p \in \mathcal{PV}$ holds.
- $T_a(w, (a, s), w')$ is a formula that encodes the action transition relation of $M$.
- $T_t(w, r, w')$ is a formula that encodes the time transition relation of $M$.

We can now define the quantifier-free first-order formula $\varphi_k$, introduced in the informal description section. The formula $\varphi_k$ is a conjunction of two formulae. The first formula $p(w)$ is a translation of a propositional variable $p$ that represents a location in question. The second formula $[M^{s^0}]_k$ encodes the unfolding of the transition relation of $M$ up to depth $k \in \mathbb{N}$.

The formula $[M^{s^0}]_k$ is defined over symbolic states $w_i$, symbolic actions $a_i$, symbolic switch costs $s_i$, and symbolic numbers $r_i$, for $0 \leq i \leq k$, and it constrains the
symbolic $k$-path to be valid $k$-run of $M$. Namely, let $T(w_i, w_{i+1}) = T_s(w_i, r_i, w_{i+1})$ if $i$ is even, and $T(w_i, w_{i+1}) = T_a(w_i, (a_i, s_i), w_{i+1})$ if $i$ is odd. Then,

$$[M^s]^k_0 := I_{s_0}(w_0) \land \bigwedge_{i=0}^{k-1} T(w_i, w_{i+1})$$

### 4 Experimental Results

In this section we experimentally evaluate the performance of our SMT-based solution to the $k$-optimal cost reachability problem by means of the weighted timed generic pipeline protocol (WTGPP).

**WTGPP.** The WTGPP (adapted from [16]) consists of $n + 2$ automata: Producer $P$ that is able to produce data within certain time interval $[a, b]$ or being inactive, Consumer $C$ that is able to receive data within certain time interval $[c, d]$, to consume data within certain time interval $[e, f]$ or being inactive, and a chain of $n$ intermediate Nodes $N_i$ which can be ready for receiving data within certain time interval $[c, d]$, processing data within certain time interval $[e, f]$, sending data, or being inactive. The local locations, the possible local actions, the local clocks, the clock constraints, invariants and the local transitions for each automaton are shown in Fig. 1. Moreover, we assume the following two switch cost functions for each automaton:

- $J^P_a(Produce) = 4, J^C_a(Consume) = 4, J^P_a(send_1) = 2, J^C_a(send_{n+1}) = 2, J^N_a(send_i) = J^N_a(send_{i+1}) = J^N_a(Proc) = 2$.
- $J^P_a(Produce) = 4000000, J^P_a(send_1) = 2000000, J^C_a(Consume) = 4000000, J^C_a(send_{n+1}) = 2000000, J^N_a(send_i) = J^N_a(send_{i+1}) = J^N_a(Proc) = 2000000$.

Finally, we assume the following duration cost function for each automaton:

- $J^P_a(ProdReady) = 4, J^C_a(ConsFree) = 4$,
- $J^P_a(j) = 1$ for $j \in \{ProdSend, ConsStart, ConsReady\} \cup \bigcup_{m=1}^n L_{N_m}$, and $i \in \{P, C, N_1, \ldots, N_n\}$.

We can define the set of global states $S$ for the protocol as the product $((L_P \times \prod_{i=1}^n L_{N_i} \times L_C) \times ([n+2] \times \mathbb{R}) \times \mathbb{R})$, and we consider the following initial state $s^0 = ((ProdReady, Node_1Start, \ldots, Node_nStart, ConsStart), (0, \ldots, 0), 0)$.

The example can be scaled by adding Nodes, or by changing the length of intervals (i.e., the parameters $a, b, c, d, e, f, g, h$) that are used to adjust the time properties of Producer $P$, Consumer $C$, and Nodes $N_i$ $(i = 1, \ldots, n)$, or by changing the switch cost functions or by changing the duration cost functions.

It should be straightforward to infer the model that is induced by the above description of WTGPP. Next, in the dense model of the protocol we assume the following set of proposition variables: $P \mathcal{V} = \{ConsFree\}$, and the following definition of valuation functions for Consumer: $V_C(ConsFree) = ConsFree$.

The $k$-optimal reachability property we consider is the following. Given a weighted timed automata model of the WTGPP system, decide whether there is a $k$-optimal run of the weighted timed automaton from the initial state of the system to the given global state of the system that contains the Consumer location $ConsFree$. 


**Fig. 1.** A WTGPP protocol.

### Performance evaluation.
We have computed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Arch Linux with the kernel 3.15.3. Moreover, we used the state of the art SMT-solver Z3 [14].

In Tables 1 and 2 we present experimental results for the WTGPP system modelled by the network of automata on Figure 1 and for $k$-optimal reachability property. As can be seen from these tables, our method allows us to locate $k$-optimal path for 5 Nodes within a reasonable time. This result does not change if we drastically increase the values of the considered switch cost function, which may suggest that the effectiveness of our method does not depend on values of the switch cost function. In addition, a noticeable enormous increase in time for 5 Nodes suggests that a significant impact on the effectiveness of our method has a number of arithmetic operations performed, especially multiplication.

<table>
<thead>
<tr>
<th>No. of Nodes</th>
<th>BMC</th>
<th>Z3</th>
<th>BMC + Z3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>0.0</td>
<td>1.8</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>0.0</td>
<td>7.3</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>0.2</td>
<td>211.6</td>
</tr>
<tr>
<td>5</td>
<td>44</td>
<td>0.3</td>
<td>7594.7</td>
</tr>
</tbody>
</table>

**Table 1.** Time and memory used for $k$-optimal path and a number of Nodes $n$. Assumed the first switch cost function.

### 5 Conclusions and Related Work

In this paper we defined, solved, and implemented the $k$-optimal cost reachability problem for a network of weighted timed automata. The proposed solution is based on the reduction to the satisfiability problem of the quantifier-free first-order formulae, and it
Table 2. Time and memory used for $k$-optimal path and a number of Nodes $n$. Assumed the second switch cost function.

<table>
<thead>
<tr>
<th>No. of Nodes</th>
<th>BMC k sec. MB</th>
<th>Z3 sec. MB</th>
<th>BMC + Z3 sec. MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12 0.0</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>18 0.0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>26 0.0</td>
<td>6.4</td>
<td>6.4</td>
</tr>
<tr>
<td>4</td>
<td>34 0.1</td>
<td>270.5</td>
<td>270.7</td>
</tr>
<tr>
<td>5</td>
<td>44 0.3</td>
<td>17715.3</td>
<td>17715.6</td>
</tr>
</tbody>
</table>

uses an external tool that is the state of the art SMT-solver Z3 [14]. Experimental results, which we carried out, show that the proposed algorithm can be extremely helpful in finding g.i.l.b. of $k$-optimal cost.

Clearly, our method takes into account discovering just lower and upper bounds on the cost to which the $k$-quasi-optimal run belongs (an unit interval $[c, c+1)$, for $c \in \mathbb{N}$), but in many real-time settings such a cost optimal approximation is sufficient.

Related Work. The issue of processing lower and upper bounds on time delays in timed automata was addressed in [9]. A duration-bounded reachability issue for timed automata expanded to incorporate the duration cost function is considered in [1]. This issue inquires as to whether there is a run of the timed automaton from the initial state to the given last state such that the duration of the run fulfils an arithmetic requirement (an optimal cost). The duration-bounded reachability issue has been also analysed in [11]. This is because the problem can be reduced to checking whether a duration formula, which defines an optimal cost, is fulfilled by a integer computation of an integration graph (a sort of a timed automaton). The solution is based on constructing a set of equations that characterises the length of time a computation spends in each location of the given automaton.

The paper [3] also handles the optimal (minimum-time) reachability problem for timed automata. Specifically, here, the issue is formulated in terms of a timed game automaton (TGA), and solved by constructing an optimal strategy utilizing a backward fixed-point calculation on the state-space of the TGA. The minimum-time reachability problem for timed automata is likewise illuminated in [15]. However here, the solution is based on the forward fixed-point algorithm that generates on-the-fly a forward reachability graph for a given timed automaton.

The paper [5] defines priced timed automata as an extension of timed automata with costs on both transitions and locations, and demonstrates how to solve the minimum cost reachability problem; this kind of automata we used in our paper. In [2] such reachability problem is called as the single-source optimal reachability problem, and it is solved by a reduction of the problem to a parametric shortest-path problem. The methods exhibited in both papers [5] and [2] are taking into account clock region graphs; in [2] the authors refer to priced timed automata as weighted timed automata.

Furthermore, the paper [6] solves the optimal reachability problem for weighted timed automata with cost functions allowing for both positive and negative costs on edges and locations, and apply the proposed method to timed games. Next, the paper [13] deals with the decidability of the optimal (minimum and maximum cost) reachability problems for multi-priced timed automata (an extension of timed automata with
multiple cost variables evolving according to given rates for each location). Finally, the paper [7] handles cost-optimal infinite schedules in terms of minimal (or maximal) cost per time ratio in the limit.

References


Reduct Calculation and Discretization of Numeric Attributes in Sparse Decision Systems

Wojciech Swieboda and Hung Son Nguyen
Institute of Mathematics, The University of Warsaw,
Banacha 2, 02-097, Warsaw Poland

Abstract. In this paper we discuss three problems in Data Mining Sparse Decision Systems: the problem of short reduct calculation, discretization of numerical attributes and rule induction. We present algorithms that provide approximate solutions to these problems and analyze the complexity of these algorithms.

1 Introduction

In the paper we discuss algorithms for Data Mining [3] Sparse Decision Tables. We first review basic notions of Information Systems, Decision Systems and Rough Set Theory [9]. We introduce a convenient representation for sparse decision tables and finally discuss algorithms for short reduct calculation, discretization and rule induction.

2 Rough Set Preliminaries

An information system is a pair $I = (U, A)$ where $U$ denotes the universe of objects and $A$ is the set of attributes. An attribute $a \in A$ is a mapping $a : U \to V_a$. The co-domain $V_a$ of attribute $a$ is often also called the value set of attribute $a$.

A decision system is a pair $D = (U, A \cup \{dec\})$ which is an information system with a distinguished attribute $dec : U \to \{1, \ldots, d\}$ called a decision attribute. Attributes in $A$ are called conditions or conditional attributes and may be either nominal or numeric (i.e. with $V_a \subseteq \mathbb{R}$).

Throughout this paper $n$ will denote the number of objects in a decision system and $k$ will denote the number of conditional attributes.

3 Sparse Data Sets and Decision Systems

In many situations a convenient way to represent the data set is in terms of Entity-Attribute-Value (EAV) Model [11], which encodes observations in terms of triples. For an information system $I = (U, A)$, the set of triples is $\{(u, a, v) : a(u) = v\}$. This representation is especially handy for information systems with numerous attributes, missing or default values. Instances with missing and default values are not included in EAV representation, which results in compression of the data set. In this paper we are only dealing with default values. Their interpretation/semantics is the same as of any other attribute. In practice we store triples corresponding to numeric attributes and to
Table 1. A typical decision system with symbolic attributes represented as a table. Attributes Diploma, Experience, French and Reference are conditions, whereas Decision is the decision attribute. All conditional attributes in this decision system are nominal.

<table>
<thead>
<tr>
<th>Diploma</th>
<th>Experience</th>
<th>French</th>
<th>Reference</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>MBA</td>
<td>Medium</td>
<td>Yes</td>
<td>Excellent</td>
</tr>
<tr>
<td>x₂</td>
<td>MBA</td>
<td>Low</td>
<td>Yes</td>
<td>Neutral</td>
</tr>
<tr>
<td>x₃</td>
<td>MCE</td>
<td>Low</td>
<td>Yes</td>
<td>Good</td>
</tr>
<tr>
<td>x₄</td>
<td>MSc</td>
<td>High</td>
<td>Yes</td>
<td>Neutral</td>
</tr>
<tr>
<td>x₅</td>
<td>MSc</td>
<td>Medium</td>
<td>Yes</td>
<td>Neutral</td>
</tr>
<tr>
<td>x₆</td>
<td>MSc</td>
<td>High</td>
<td>Yes</td>
<td>Excellent</td>
</tr>
<tr>
<td>x₇</td>
<td>MBA</td>
<td>High</td>
<td>No</td>
<td>Good</td>
</tr>
<tr>
<td>x₈</td>
<td>MCE</td>
<td>Low</td>
<td>No</td>
<td>Excellent</td>
</tr>
</tbody>
</table>

Table 2. A decision system in which all conditional attributes are numeric.

<table>
<thead>
<tr>
<th>a₁, a₂, a₃</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>0 1.3 0  F</td>
</tr>
<tr>
<td>x₂</td>
<td>3.3 0.9 0  F</td>
</tr>
<tr>
<td>x₃</td>
<td>0 1.5 0  F</td>
</tr>
<tr>
<td>x₄</td>
<td>0 1.2 2.5 F</td>
</tr>
<tr>
<td>x₅</td>
<td>0 1.3 3.6 F</td>
</tr>
<tr>
<td>x₆</td>
<td>3.7 2.7 2.4 T</td>
</tr>
<tr>
<td>x₇</td>
<td>4.1 1.0 2.8 T</td>
</tr>
</tbody>
</table>

symbolic attributes in two separate tables, and store decisions (which we assume are never missing) of objects in a separate vector.

Another related representation, more general then EAV model, is Subject-Predicate-Object (SPO), and is used e.g. in Resource Description Framework (RDF) Model and implemented in several Triplestore databases.

4 Problems for Sparse Decision Systems

In our paper we address the following problems for Sparse Decision Systems:

1. Finding a short reduct or a superreduct [1].
   A reduct is a subset of attributes $R \subseteq A$ which guarantees discernibility of objects belonging to different decision classes.

2. Discretization of numerical attributes [6].
   Discretization of a decision system is determining a set of cuts on numerical attributes so that the induced partitions (i.e. intervals between cutpoints) guarantee discernibility of objects belonging to different decision classes.

3. Generating set of rules or dynamic rules [1].
Table 3. EAV representation of decision system in table 1. The default values (omitted in this representation) for consecutive attributes are 'MBA', 'Low', 'Yes' and 'Excellent'.

<table>
<thead>
<tr>
<th>Entity Attribute Value</th>
<th>Entity Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$ a_2 Medium</td>
<td>$x_1$ Accept</td>
</tr>
<tr>
<td>$x_2$ a_4 Neutral</td>
<td>$x_2$ Reject</td>
</tr>
<tr>
<td>$x_3$ a_1 MCE</td>
<td>$x_3$ Reject</td>
</tr>
<tr>
<td>$x_4$ a_2 High</td>
<td>$x_4$ Accept</td>
</tr>
<tr>
<td>$x_5$ a_1 MSc</td>
<td>$x_5$ Reject</td>
</tr>
<tr>
<td>$x_7$ a_2 High</td>
<td>$x_7$ Accept</td>
</tr>
<tr>
<td>$x_8$ a_1 MCE</td>
<td>$x_8$ Reject</td>
</tr>
<tr>
<td>$x_7$ a_3 No</td>
<td>$x_7$ Accept</td>
</tr>
<tr>
<td>$x_7$ a_4 Good</td>
<td>$x_8$ Accept</td>
</tr>
<tr>
<td>$x_8$ a_3 No</td>
<td>$x_8$ Reject</td>
</tr>
</tbody>
</table>

Table 4. EAV representation of decision system in table 2. The default value (omitted in this representation) for each attribute is 0.

<table>
<thead>
<tr>
<th>Entity Attribute Value</th>
<th>Entity Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$ a_2 1.3</td>
<td>$x_1$ T</td>
</tr>
<tr>
<td>$x_2$ a_1 3.3</td>
<td>$x_2$ T</td>
</tr>
<tr>
<td>$x_2$ a_2 0.9</td>
<td>$x_3$ T</td>
</tr>
<tr>
<td>$x_3$ a_2 1.5</td>
<td>$x_4$ T</td>
</tr>
<tr>
<td>$x_4$ a_2 1.2</td>
<td>$x_5$ T</td>
</tr>
<tr>
<td>$x_4$ a_3 2.5</td>
<td>$x_6$ T</td>
</tr>
<tr>
<td>$x_5$ a_2 1.3</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_5$ a_3 3.6</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_6$ a_1 3.7</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_6$ a_2 2.7</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_6$ a_3 2.4</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_7$ a_1 4.1</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_7$ a_2 1.0</td>
<td>$x_7$ T</td>
</tr>
<tr>
<td>$x_7$ a_3 2.8</td>
<td>$x_7$ T</td>
</tr>
</tbody>
</table>

References

Table 5. A discretized version of the decision system presented in table 2.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-\infty, \infty)$</td>
<td>$(1.25, +\infty)$</td>
<td>$(-\infty, 1.2]$</td>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, \infty)$</td>
<td>$(-\infty, 1.1]$</td>
<td>$(-\infty, 1.2]$</td>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, \infty)$</td>
<td>$(1.25, +\infty)$</td>
<td>$(-\infty, 1.2]$</td>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, +\infty)$</td>
<td>$(1.1, 1.25]$</td>
<td>$(1.2, +\infty)$</td>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, +\infty)$</td>
<td>$(1.25, +\infty)$</td>
<td>$(1.2, +\infty)$</td>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, +\infty)$</td>
<td>$(1.25, +\infty)$</td>
<td>$(1.2, +\infty)$</td>
<td>$T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(-\infty, +\infty)$</td>
<td>$(-\infty, 1.1]$</td>
<td>$(1.2, +\infty)$</td>
<td>$T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exploiters-Based Knowledge Extraction in Object-Oriented Knowledge Representation

Dmytro Terletskyi

Taras Shevchenko National University of Kyiv, Kyiv, 03680, Ukraine
dmytro.terletskyi@gmail.com,
http://cyb.univ.kiev.ua/en/departments.is.terletskyi.html

Abstract. This paper contains the consideration of knowledge extraction mechanisms of such object-oriented knowledge representation models as frames, object-oriented programming and object-oriented dynamic networks. In addition, conception of universal exploiters within object-oriented dynamic networks is also discussed. The main result of the paper is introduction of new exploiters-based knowledge extraction approach, which provides generation of a finite set of new classes of objects, based on the basic set of classes. The methods for calculation of quantity of new classes, which can be obtained using proposed approach, and of quantity of types, which each of them describes, are proposed. Proof that basic set of classes, extended according to proposed approach, together with union exploiter create upper semilattice is given. The approach always allows generating of finitely defined set of new classes of objects for any object-oriented dynamic network. A quantity of these classes can be precisely calculated before the generation. It allows saving of only basic set of classes in the knowledge base.

Key words: knowledge extracting, object-oriented dynamic networks, inhomogeneous class, universal exploiters, upper semilattice

1 Introduction

Nowadays methods of knowledge extracting and reasoning about knowledge are significant constituent part of majority of knowledge-based systems. It gives an opportunity to extract or to obtain new knowledge, based on such called, basic knowledge. Such abilities make knowledge-based systems intelligent and applicable ones at least in such areas of artificial intelligence as information search, problem solving, planning, patterns recognition, decision making, etc.

Currently there is variety of knowledge representation models (KRM), which implement different approaches to knowledge representation. One of them is an object-oriented knowledge representation, the main idea of which is representation of knowledge in terms of objects, classes of objects and relationships among them. Nowadays the most famous KRM within this approach are frames and object-oriented programming (OOP). Both of them have their own knowledge extraction methods, which give some abilities for new knowledge obtaining. Let us consider these mechanisms and their main features in more detail.
2 Knowledge Extraction in Frames and OOP

Frames as a KRM provide representation of knowledge in terms of hierarchies of frames (system of frames), where particular frame is a class-frame or instance-frame [3], [6]. Each frame is connected with others via relations of generalization (is-a, a-kind-of, an-instance-of, etc.), aggregation (a-part-of, part-whole, etc.) and association (owns, plays, creates, etc.). Relation of generalization provides implementation of inheritance mechanism that allows more specific frames, that situated lower in the hierarchy, inherit all slots from more general frames. Such structure of the system allows efficient knowledge representation, because it is based on the idea of representation of new knowledge via previously represented ones.

In addition, each frame can have definite procedural attachments, which allow execution of actions on it. Some procedures execute only when they are in need (when-procedures), other ones execute in particular situations. Thus, extracting of new knowledge in the frame system can be done by dint of the reasoning in inheritance hierarchy or by means of procedural attachments executing [10]. However, frames support two kinds of inheritance – single and multiple ones [6]. Inheritance can cause such problems as problem of exceptions, problem of redundancy and problem of ambiguity [1], [9]. Frames also allow overriding of values of slots in the instance-frames [6], that leads to the situation when the subclass or instance goes beyond its superclass.

In contrast to frames, OOP is divided on two styles – class-based and prototype based ones [5]. Similarly to frames, first approach provides knowledge representation in terms of hierarchies of classes, using inheritance. Second one gives an opportunity for knowledge representation in terms of prototypes. Despite that both styles are object-oriented ones, they have significant differences.

Class-based approach provides ability to work only with instantiated objects, to change values of their properties, to execute their methods and in such a way to obtain new knowledge. We cannot change the description of a class, type of object or hierarchy of classes during program execution. It means that we can obtain only objects of the same type with changed values of the properties. In addition, inheritance in OOP causes the same problems as in frames [5].

Prototype-based style gives an opportunity to operate with prototypes. Each new prototype is a modified clone of another one. It means that such approach is more flexible for description of new concepts, because it allows creating of new prototypes during program execution and implements the idea of partial inheritance. However, it requires much more computer memory and leads to redundancy in representation of particular prototypes.

3 Object-Oriented Dynamic Networks

Besides mentioned object-oriented KRMs, there is one more KRM, such as object-oriented dynamic networks (OODNs), that was proposed in [8]. This KRM has similarity with all mentioned KRMs, however it also has some specific features, which give new opportunities in knowledge representation within object-oriented approach. Let us consider structure of this model.
Definition 1  Object-Oriented Dynamic Network is a 5-tuple

\[ \text{OODN} = (O, C, R, E, M), \]

where:

- \( O \) – a set of objects;
- \( C \) – a set of classes of objects, which describe objects from the set \( O \);
- \( R \) – a set of relations, which are defined on the set \( O \) and \( C \);
- \( E \) – a set of exploiters, which are defined on the set \( O \) and \( C \);
- \( M \) – a set of modifiers, which are defined on the set \( O \) and \( C \).

Definitions of all elements from the tuple \( \text{OODN} = (O, C, R, E, M) \) were introduced and considered in detail in [8]. Each object from the set \( O \) has some properties, which define it as an essence. There are two kinds of object’s properties – quantitative and qualitative ones, which definitions were introduced in [7]. However, we need to consider the properties of a class of objects. Let us define them.

Definition 2  Quantitative property of class of objects \( T \) is a tuple

\[ p(T) = (v(p(T)), u(p(T))), \]

where \( v(p(T)) \) is an quantitative value of \( p(T) \) and \( u(p(T)) \) are units of its measure.

Definition 3  Qualitative property of class of objects \( T \) is a verification function \( p(T) = v f(T) \), which is defined as a mapping \( v f(T) : p(T) \to [0, 1] \) and reflects the degree (measure) of truth (presence) of a property \( p(T) \) for the class \( T \).

Let us define the conception of equivalence of these kinds of properties.

Definition 4  Two quantitative properties \( p(T_1) \) and \( p(T_2) \) of arbitrary classes of objects \( T_1 \) and \( T_2 \) are equivalent, i.e. \( \text{Eq}(p(T_1), p(T_2)) = 1 \), if and only if \( u(p(T_1)) = u(p(T_2)) \land (v(p(T_1)) = v(p(T_2))). \)

Definition 5  Two qualitative properties \( p_1(T_1) \) and \( p_2(T_2) \) of arbitrary classes of objects \( T_1 \) and \( T_2 \) are equivalent, i.e. \( \text{Eq}(p_1(T_1), p_2(T_2)) = 1 \), if and only if \( v f_1(T_1) = v f_2(T_1) \land (v f_1(T_2) = v f_2(T_2)). \)

For every object of a class we can define methods, which can be applied to them and allow definition of their behaviour and manipulating on them.

Definition 6  Method of class of objects \( T \) is a function \( f(T) \), which can be applied to the class \( T \), considering the features of its specification (vector of properties).

From the previous definition, we can see that method is a function, which is defined under the properties. To define the equivalence of methods we should define the equivalence of two arbitrary functions, but in general case such problem is unsolvable one. So, we are going to introduce the equivalence of methods via following definition.

Definition 7  Two methods \( f_1(T_1) \) and \( f_2(T_2) \) of arbitrary classes of objects \( T_1 \) and \( T_2 \) are equivalent, i.e. \( \text{Eq}(f_1(T_1), f_2(T_2)) = 1 \), if and only if \( f_1(T_1) = f_1(T_2) \land (f_2(T_1) = f_2(T_2)). \)
It introduces the equivalence of two methods on the same argument. It means that two methods \( f_1(T_1) \) and \( f_2(T_2) \) can be different as functions, however they can return the same results on the same objects.

Concepts of objects, classes and relations among them have different implementations in various KRMs. One of the main differences is the definition of the class. Within frames and OOP, concept of class is defined as abstract description of some quantity of objects, which have the same nature [3], [5]. That is why, it is possible to conclude, that such class is a homogeneous one, because it contains only objects of the same type. Nevertheless, there are classes, which are inhomogeneous ones [7]. Within OODNs, there are two definitions for both types of classes. Let us consider them in more details.

**Definition 8** Homogeneous class of objects \( T \) is a tuple \( T = (P(T), F(T)) \), where \( P(T) \) is specification (a vector of properties) of some quantity of objects, and \( F(T) \) is their signature (a vector of methods).

According to this definition, all objects of such class have the same type, i.e. they have the same properties and methods as their class. Let us consider the definition of inhomogeneous class of objects.

**Definition 9** Inhomogeneous (heterogeneous) class of objects \( T \) is a tuple

\[
T = (\text{Core}(T), \text{pr}_1(A_1), \ldots, \text{pr}_n(A_n))
\]

where \( \text{Core}(T) = (P(T), F(T)) \) is the core of class of objects \( T \), which includes only properties and methods similar to corresponding properties of specifications \( P(A_1), \ldots, P(A_n) \) and corresponding methods of signatures \( F(A_1), \ldots, F(A_n) \) respectively, and where \( \text{pr}_i(A_i) = (P(A_i), F(A_i)) \), \( i = 1, \ldots, n \) are projections of objects \( A_1, \ldots, A_n \), which consist of properties and methods typical only for these objects.

According to this definition, it is possible to represent certain amount of any types by dint of one class within object-oriented approach. While representation of each type of objects in OOP always requires definition of new class.

Analyzing definitions 8 and 9, we can conclude that a homogeneous class of objects defines a type of objects. In this case the type and the class of objects mean the same. However, an inhomogeneous class of objects defines at least two different types of objects within one class of objects that is why in this case the type and the class are not equivalent. In other words inhomogeneous class of objects includes a few types of objects. Let us define a type of inhomogeneous class of objects.

**Definition 10** Type of arbitrary inhomogeneous class of objects

\[
T = (\text{Core}(T), \text{pr}_1(T), \ldots, \text{pr}_n(T))
\]

is a homogeneous class of objects \( T_i = (\text{Core}(T), \text{pr}_i(T)) \), where \( i = 1, \ldots, n \).

Now, let us define the following tree kinds of subclass relations for classes of objects: homogeneous \( \subseteq \) homogeneous, inhomogeneous \( \subseteq \) inhomogeneous and homogeneous \( \subseteq \) inhomogeneous.
Definition 11 Homogeneous class of objects $T_1 = (P(T_1), F(T_1))$ is a subclass of homogeneous class of objects $T_2 = (P(T_2), F(T_2))$, i.e. $T_1 \subseteq T_2$ if and only if

$$(\forall p_i \in P_1 \exists p_j \in P_2 | E(q(p_i, p_j) = 1) \wedge (\forall f_k \in F_1 \exists f_w \in F_2 | E(q(f_k, f_w) = 1),$$

where $P_1, P_2, F_1, F_2$ are sets, which contain elements of vectors $P(T_1), P(T_2), F(T_1), F(T_2)$ respectively and $i = 1, |P_1|, j = 1, |P_2|, k = 1, |F_1|, w = 1, |F_2|$.

Definition 12 Inhomogeneous class of objects

$T_1 = (Core(T_1), pr_1(T_1), \ldots, pr_n(T_1))$

is a subclass of inhomogeneous class of objects

$T_2 = (Core(T_2), pr_1(T_2), \ldots, pr_m(T_2))$,

i.e. $T_1 \subseteq T_2$ if and only if

$$(\forall a_i \in C_1 \exists a_j \in C_2 | E(q(a_i, a_j) = 1) \wedge (\forall b_h \in pr_h \exists b_{v_w} \in pr_v | E(q(b_h, b_{v_w}) = 1),$$

where $C_1, C_2, pr_h, pr_v$ are sets, which contain elements of vectors from the sets $Core(T_1), Core(T_2), pr_h(T_1), pr_v(T_2)$ respectively and $i = 1, |C_1|, j = 1, |C_2|, k = 1, |pr_h|, w = 1, |pr_v|, h = 1, m, v = 1, n$.

Definition 13 Homogeneous class of objects $T_1 = (P(T_1), F(T_1))$ is a subclass of inhomogeneous class of objects $T_2 = (Core(T_2), pr_1(T_2), \ldots, pr_n(T_2))$, i.e. $T_1 \subseteq T_2$ if and only if

$$(\forall p_i \in P_1 \exists p_j \in C_2 \lor pr_v | E(q(p_i, p_j) = 1) \wedge (\forall f_k \in F_1 \exists f_w \in C_2 \lor pr_v | E(q(f_k, f_w) = 1),$$

where $P_1, F_1$ are sets, which contain elements of vectors $P(T_1), F(T_1)$ and $C_2, pr_v$ are sets, which contain elements of vectors from the sets $Core(T_2)$ and $pr_v(T_2)$ respectively, $i = 1, |P_1|, j = 1, |C_2| + |pr_v|, k = 1, |F_1|, w = 1, |C_2| + |pr_v|, v = 1, n$.

According to the definitions of class of objects, it is possible to define the vector of methods for each class of objects, concerning its specification. Such kind of methods are internal ones, because they are defined under particular properties of the class. Besides them, there are methods, which are called external ones and are defined under whole specification of the class. Depending on the character of actions, all methods can be divided on exploiters and modifiers. Exploiters do not change objects or classes, they just use them as parameters for new knowledge obtaining. While, modifiers change the basic knowledge and allow modelling of their changes or evolution over the time. That is why $F(T)$ contains internal, $E$ and $M$ contain external methods of the class of objects.

Summarizing, OODN can be considered as two conceptual parts. First of them is declarative, which includes sets $O, C, R$, and allows representation of knowledge about particular domain. Second part is procedural one. It includes sets $E, M$ and provides the tools for obtaining new knowledge from basic ones. All following considerations are connected with applications of procedural part of OODN, in particular exploiters-based knowledge extraction.
4 Exploiters-Based Knowledge Extraction

As it was mentioned above, exploiters form significant constituent of procedural part of OODN. Generally, we can define variety of exploiters for each class of objects, however majority of them are locally closed under their classes. That is why, such universal exploiters as union, intersection, difference and symmetrical difference were introduced in [7]. Their applications allow building of new classes of objects. This fact has significant value not only in knowledge extraction, but also in programming, because it is a step toward the implementation of runtime class generation.

Let us define union exploiter for classes of objects, using definition 10.

**Definition 14** Union $\cup$ of two arbitrary nonequivalent classes of objects $T_1$ and $T_2$ is an inhomogeneous class of objects $T = (\text{Core}(T), \text{pr}_1(T), \ldots, \text{pr}_n(T))$, where $\text{Core}(T) = (P(T), F(T))$ is its core and includes only properties and methods, which are similar for types $T_1, T_1, \ldots, T_m, T_2, \ldots, T_k$, and where $\text{pr}_j(T) = (P(T), F(T))$ is projection of type $T_{ij}, i = 1, 2, j = 1, n, n = m + k$ which consist of properties and methods typical only for this type.

Application of union exploiter to classes of objects has some important features besides generation of new classes of objects. Let us formulate and prove a few theorems, which illustrate these features.

**Theorem 1** For any OODN $= (O, C = \{T_1, \ldots, T_n\}, R, E = \{\cup\}, M)$, where classes $T_1, \ldots, T_n$ are homogeneous ones and do not have any common properties and methods, all possible applications of union exploiter $\cup$, including all possible its superpositions, to classes of objects from the set $C$ always generate finite quantity of new classes of objects, which can be calculated by the following formula:

$$q(C_E) = 2^n - n - 1,$$

where $n = |C|$.

**Proof.** It is known that the number of all possible unique $k$-elements combinations from the $n$-elements set can be calculated as $C_n^k$. Similarly, the number of all possible unique classes of objects created from the basic set of classes $C = \{T_1, \ldots, T_n\}$ using union exploiter can be represented as a combination of $k = 2, n$ different classes from the set $C$. It is known that

$$\sum_{n=0}^{k} C_n^k = 2^n.$$

However, we cannot create classes of objects, which describe 1 and 0 different types, applying union exploiter to the classes of objects from the set $C$, i.e. we do not count $C_0^n$ and $C_1^n$, we can conclude that

$$q(C_E) = \sum_{k=0}^{n} C_n^k - C_n^0 - C_n^1 = \sum_{k=2}^{n} C_n^k = 2^n - n - 1.$$

$\square$
Using Theorem 1, we can formulate one more important theorem.

**Theorem 2** Set of classes of objects 

\[ C = \{T_1, \ldots, T_n, T_{n+1}, \ldots, T_{2n-1}\} \]

of any OODN, extended according to Theorem 1, with exploiter \( \cup \) create the upper semilattice, where class \( T_1, \ldots, n = T_1 \cup \cdots \cup T_n \) is its greatest upper bound.

**Proof.** According to the definition of upper semilattice, it should be a system \( SL = (A, \Omega) \), where \( A \) is a poset, \( \Omega = \{\lor\} \) and \( \lor \) is binary, idempotent, commutative and associative operation [2], [4].

In our case, carrier of upper semilattice is the set of classes \( C \), where we define exploiter \( \cup \), thus \( SL = (C, \Omega) \), where \( \Omega = \{\cup\} \). From the definition 14 it follows, that mentioned properties of \( \lor \) are true for \( \cup \), i.e.

1. \( T_1 \cup T_1 = T_1 \);
2. \( T_1 \cup T_2 = T_2 \cup T_1 \);
3. \( T_1 \cup (T_2 \cup T_3) = (T_1 \cup T_2) \cup T_3 \).

Now, let us show that \( C \) is a poset. For this, we should define \( \forall T_1, T_2 \in C \mid T_1 \subseteq T_2 \Leftrightarrow T_1 \cup T_2 = T_2 \) and show that \( \subseteq \) is a relation of partial order under the set \( C \). It means, we should prove that relation \( \subseteq \) is reflexive, antisymmetric, and transitive one.

1. \( T_1 \subseteq T_1 \Leftrightarrow T_1 \cup T_1 = T_1 \) follows from idempotency of \( \cup \);
2. \( T_1 \subseteq T_2 \Leftrightarrow T_1 \cup T_2 = T_2, T_2 \subseteq T_1 \Leftrightarrow T_2 \cup T_1 = T_1 \) and from commutativity of \( \cup \), we can conclude that \( T_1 = T_2 \);
3. \( T_1 \subseteq T_2 \Leftrightarrow T_1 \cup T_2 = T_2, T_2 \subseteq T_3 \Leftrightarrow T_2 \cup T_3 = T_3 \Rightarrow (T_1 \cup T_2) \cup T_3 = T_1 \cup (T_2 \cup T_3) = T_1 \cup T_3 = T_3 \Rightarrow T_1 \cup T_3 = T_3 \Leftrightarrow T_1 \subseteq T_3 \).

Thus, \( SL = (C = \{T_1, \ldots, T_n, T_{n+1}, \ldots, T_{2n-1}\}, \Omega = \{\cup\}) \) is an upper semilattice, where \( T_1, \ldots, n = T_1 \cup \cdots \cup T_n \) is its greatest upper bound. \( \square \)

Using results of Theorem 2, we can formulate the following corollary.

**Corollary 21** Set of classes of objects \( C \) of any OODN, extended according to Theorem 1, and union exploiter \( \cup \), which is defined under it, create a finitely-generated universal algebra

\[ G = (C = \{T_1, \ldots, T_n, T_{n+1}, \ldots, T_{2n-1}\}, \Omega = \{\cup\}) \],

where \( C_b = \{T_1, \ldots, T_n\} \) is generative set for the set \( C \).

Now let us consider an example, which illustrates specific of exploiters-based knowledge extraction within OODN. Let us define the OODN

\[ \text{Salad} = (O, C, R, E, M) \],

which describes some ingredients of a salad, for example cucumber, tomato, onion, cabbage, salt and sunflower oil. For this purpose, we define following sets of objects \( O \), classes of objects \( C \) and set of relations \( R \)

\[ O = \{\text{cuc}, \text{tom}, \text{on}, \text{cab}, \text{sal}, \text{soil}\} \],

\[ C = \{T_1, \ldots, T_n, T_{n+1}, \ldots, T_{2n-1}\} \],
\[ C = \{Cuc, Tom, Cab, On, Spi, Oil\}, \]

\[ R = \{\text{cuc} \text{an-inst-of} Cuc, \text{tom} \text{an-inst-of} Tom, \text{cab} \text{an-inst-of} Cab, \]
\[ \text{on} \text{an-inst-of} On, \text{sal} \text{an-inst-of} Spi, \text{soil} \text{an-inst-of} Oil.\} \]

Suppose the set of exploiters is defined as \( E = \{\cup\}. \) We do not define the set of modifiers \( M, \) because it is not necessary within consideration of exploiters-based knowledge extraction.

Let us define the specifications of classes from set \( C \) in the following way

\[ P(Cuc) = (p_1(Cuc), \ldots, p_4(Cuc)), \]
\[ P(Tom) = (p_1(Tom), \ldots, p_4(Tom)), \]
\[ P(Cab) = (p_1(Cab), \ldots, p_4(Cab)), \]
\[ P(On) = (p_1(On), \ldots, p_4(On)), \]
\[ P(Spi) = (p_1(Spi), \ldots, p_4(Spi)), \]
\[ P(Oil) = (p_1(Oil), \ldots, p_4(Oil)), \]

where \( p_1(Cuc), p_1(Tom), p_1(Cab), p_1(On) \) – masses of vegetables, \( p_1(Spi) \) – type of spices, \( p_1(Oil) \) – type of oil, \( p_2(Cuc), p_2(Tom), p_2(Cab), p_2(On) \) – colors of vegetables, \( p_2(Spi) \) – mass of spices, \( p_2(Oil) \) – volume of oil, \( p_3(Cuc), p_3(Tom), p_3(Cab), p_3(On) \) – freshness of vegetables, \( p_3(Spi) \) – taste of spices, \( p_3(Oil) \) – color of oil, \( p_4(Cuc), p_4(Tom), p_4(Cab), p_4(On), p_4(Spi), p_4(Oil) \) – prices. Values of all properties of these classes are defined in Table 1.

Let us define the specifications of objects from the set \( O, \) using specifications of their classes (see Table 2).

<table>
<thead>
<tr>
<th>( p_i )</th>
<th>Cuc</th>
<th>Tom</th>
<th>Cab</th>
<th>On</th>
<th>Spi</th>
<th>Oil</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 )</td>
<td>0.07, 0.18</td>
<td>kg</td>
<td>0.08, 0.2</td>
<td>kg</td>
<td>0.4, 1.3</td>
<td>kg</td>
</tr>
<tr>
<td>( p_2 )</td>
<td>green</td>
<td>red</td>
<td>green</td>
<td>green-white</td>
<td>kg</td>
<td>0.1, 1.0</td>
</tr>
<tr>
<td>( p_3 )</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
<td>kg</td>
<td>0.1, 1.0</td>
</tr>
<tr>
<td>( p_4 )</td>
<td>3 USD/kg</td>
<td>3.5 USD/kg</td>
<td>4 USD/kg</td>
<td>2 USD/kg</td>
<td>12 USD/kg</td>
<td>9 USD/l</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( p_i )</th>
<th>cuc1</th>
<th>cuc2</th>
<th>tom1</th>
<th>tom2</th>
<th>cab</th>
<th>on</th>
<th>sal</th>
<th>soil</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 )</td>
<td>0.09 kg</td>
<td>0.08 kg</td>
<td>0.12 kg</td>
<td>0.1 kg</td>
<td>0.5 kg</td>
<td>0.1 kg</td>
<td>salt</td>
<td>sunflower</td>
</tr>
<tr>
<td>( p_2 )</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>red</td>
<td>green-white</td>
<td>0.5 kg</td>
<td>0.5 l</td>
<td></td>
</tr>
<tr>
<td>( p_3 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>salty</td>
<td>yellow</td>
</tr>
<tr>
<td>( p_4 )</td>
<td>0.27 USD</td>
<td>0.24 USD</td>
<td>0.42 USD</td>
<td>0.35 USD</td>
<td>2 USD</td>
<td>0.2 USD</td>
<td>6 USD</td>
<td>4.5 USD</td>
</tr>
</tbody>
</table>

We have described the OODN for the salad. Clearly that all elements of sets \( O, C \) and \( R \) are basic knowledge. Let us obtain all possible new knowledge from them using
exploiter $\cup$. According to Theorems 1-2 we obtain such 15 classes, that each of them describes 2 different types of objects

\[
\text{CucTom, CucCab, CucOn, CucSpi, CucOil, TomCab, TomOn, TomSpi, TomOil, CabOn, CabSpi, CabOil, OnSpi, OnOil, SpiOil;}
\]

such 20 classes, that each of them describes 3 different types of objects

\[
\text{CucTomCab, CucTomOn, CucTomSpi, CucTomOil, CucCabOn, CucCabSpi, CucCabOil, CucOnSpi, CucOnOil, CucSpiOil, TomCabOn, TomCabSpi, TomCabOil, TomOnSpi, TomOnOil, TomSpiOil, CabOnSpi, CabOnOil, CabSpiOil, OnSpiOil;}
\]

such 15 classes, that each of them describes 4 different types of objects

\[
\text{CucTomCabOn, CucTomCabSpi, CucTomCabOil, CucTomOnSpi, CucTomOnOil, CucTomOnOil, CucTomOil, CucCabOnSpi, CucCabOnOil, CucCabOil, CucOnSpiOil, CucOnOil, CucOnOil, TomCabOnSpi, TomCabOnOil, TomCabOnOil, TomOnSpiOil, TomOnOil, TomOnOil, CabOnSpiOil, CabOnOil, CabSpiOil, OnSpiOil;}
\]

such 6 classes, that each of them describes 5 different types of objects

\[
\text{CucTomCabOnSpi, CucTomCabOnOil, CucTomCabOnOil, CucTomOnSpi, CucTomOnOil, CucTomOnOil, CucOnOil, TomCabOnSpi, TomCabOnOil, TomCabOnOil, TomOnSpiOil, TomOnOil, TomOnOil;}
\]

and 1 class, which describes 6 different types of objects

\[
\text{CucTomCabOnSpiOil.}
\]

As we can see, we obtain 57 new classes of objects, or in other words, 57 different combinations of salad’s ingredients from the 6 basic ones. In such a way we extended the set of classes $C$ by adding new knowledge, extracted from basic ones. The most general obtained class $\text{CucTomCabOnSpiOil}$ is an inhomogeneous one and has following structure

\[
\text{CucTomCabOnSpiOil} = (pr_1(\text{CucTomCabOnSpiOil}), \ldots, pr_6(\text{CucTomCabOnSpiOil})),
\]

where

\[
pr_1(\text{CucTomCabOnSpiOil}) = P(\text{Cuc}), pr_2(\text{CucTomCabOnSpiOil}) = P(\text{Tom}),
\]

\[
pr_3(\text{CucTomCabOnSpiOil}) = P(\text{Cab}), pr_4(\text{CucTomCabOnSpiOil}) = P(\text{On}),
\]

\[
pr_5(\text{CucTomCabOnSpiOil}) = P(\text{Spi}), pr_6(\text{CucTomCabOnSpiOil}) = P(\text{Oil}).
\]

All other obtained classes have the similar structure.
According to Theorem 2, the extended set of classes \( C \) and exploiter \( \cup \) create upper semilattice. It means, that there is partial order relation \( \subseteq \), which is defined on the set \( C \). Furthermore, according to Corollary 21 they create a finitely-generated universal algebra

\[
G = (C = \{Cuc, Tom, Cab, On, Spi, Oil, \ldots, CucTomCabOnSpiOil\}, E = \{\cup\}),
\]

where \( C_b = \{Cuc, Tom, Cab, On, Spi, Oil\} \subseteq C \) is generative or basic set for the set \( C \).

Summarizing, we obtained all possible unions of the basic classes from the set \( C \). All these classes can be viewed as schemas or recipes for which we can use objects defined in Table 2. In means that in such a way we can create particular salad, moreover we can create different salads using one scheme putting different proportions of ingredients. Using chosen scheme, we can calculate different properties of the cooked salad, for example its prise, mass, etc.

5 Conclusions

This paper contains consideration of main features of knowledge extraction mechanisms of such object-oriented KRMs as frames, OOP and OODNs. Furthermore, conception of universal exploiters within object-oriented dynamic networks is also discussed.

The main achievement of the paper is introduction of new exploiters-based knowledge extraction method for OODNs, which always provides generating of finitely defined set of new classes of objects, based on the basic set of classes. The main features of the proposed method are:

- ability to calculate:
  - quantity of new classes, which can be obtained, using proposed approach,
  - quantity of different types, which each of obtained classes describes;
- the basic set of classes of any OODN, extended according to proposed approach, together with union exploiter, create:
  - upper semilattice,
  - finitely generated universal algebra, for which the basic set of classes of OODN is a generative set.

It allows us to extract new knowledge from the basic ones when we need them and to save only basic set of classes in the knowledge base and database. Moreover, obtained knowledge always have the defined structure, i.e. they form the upper semilattice. It means that we can use the results of upper semilattice theory in such kind of knowledge extraction and representation.

However, despite all advantages, proposed approach requires further research, at least in the following directions:

- study of the case when the basic set of classes of OODN contains classes that has some common properties or methods,
- study of the case when the OODN is a fuzzy one,
- adapting and usage of proposed approach in other known object-oriented KRMs.
References

Space-Time Viewpoints for Concurrent Processes
Represented by Relational Structures

Irina Virbitskaite\textsuperscript{1,2}, Elena Bozhenkova\textsuperscript{1,2}, Evgeny Erofeev\textsuperscript{3}

\textsuperscript{1} A.P. Ershov Institute of Informatics Systems, SB RAS
6, Acad. Lavrentiev avenue, 630090, Novosibirsk, Russia
\textsuperscript{2} Novosibirsk State University
2, Pirogov avenue, 630090, Novosibirsk, Russia
\textsuperscript{3} Carl von Ossietzky University of Oldenburg
D-26111 Oldenburg, Germany

1 Introduction

In contrast to the standard physical theories which model systems by the continuum, Petri proposed a combinatorial representation of a spacetime — concurrent structures (with occurrence nets as a special case thereof) in which notions corresponding to the relativistic concepts of world line and causal cone can be defined by means of the concurrency and causal dependence relations, respectively. As a result, the density notion of the continuum model is replaced by several properties — so called concurrency axioms (including $K$-density, $N$-density, etc.). $K$-density is based on the idea that at any time instant, any sequential subprocess of a concurrent structure must be in some state or changing its state. $N$-density can be viewed as a sort of local density. Furthermore, it has turned out that concurrency axioms allow avoiding inconsistency between syntactic and semantic representations of processes and, thereby, to exclude unreasonable processes represented by the concurrent structures.

Petri’s occurrence nets model system behaviors by occurrences of local states (also called conditions) and of events which are partially ordered. The partial order is interpreted as a kind of causal dependency relation. Also, occurrence nets are endowed with a symmetric, but in general non-transitive, concurrency relation — absence of the causality. Poset models do not discriminate between conditions and events. The power and limitations of concurrency axioms in the context of occurrence nets \cite{4, 5} and posets \cite{6, 9, 20} have been widely studied to allow better understanding the connections of causality and concurrency relations between systems events. In contrast to these treatments, the authors of the paper \cite{16} have dealt with causality and concurrency on cyclic processes represented by net models which do not require an underlying partial order of causality. The paper \cite{21} has studied the interrelations between concurrency axioms in the setting of prime event structures (occurrence nets with forward hereditary (w.r.t. causality) conflicts), where the nondeterministic aspects of concurrent processes are explicitly described. In the more recent papers \cite{1, 2}, algebraic and orthomodular lattices (the elements of the lattices are the closed subsets w.r.t. a closure operator, defined starting from the concurrency relation) have been generated from occurrence nets with and

\textsuperscript{*} This work is supported in part by DFG-RFBR (project CAVER, grants BE 1267/14-1 and 14-01-91334)
without forward conflicts. Also, an alternative characterization of K-density is given on
the basis of a relation between maximal sets of pairwise causally related elements and
closed sets.

It is known that some aspects of concurrent behavior (e.g., the specification of pri-
orities, error recovery, inhibitor nets, treatment of simultaneity, etc.) are to some extent
problematic to be dealt with using only partially ordered causality based models. One
way to cope with the problems is to utilize the model of a so called relational struc-
ture — a set of elements (systems events) with a number of different kind relations
on it. The authors of the papers [11–14] have proposed and carefully studied a sub-
class of the model where general causal concurrent behavior is represented by a pair
of relations instead just one, as in the standard partial order approach. Depending on
the assumptions and goals for the chosen model of concurrency, the pair of the rela-
tions are interpreted in two versions: either as partially ordered causality and irreflexive
weak causality (not in general a partial order) or as a symmetric and irreflexive mutex
relation (non-simultaneity) and irreflexive weak causality (herewith, the relations may
not be completely distinct). The approaches allow modeling and studying concurrency
at different levels of consideration: from abstract behavioral observations — concur-
rent histories (consisting of step sequence executions) to system level models such as
elementary Petri nets and their generalizations with inhibitor arcs and mutex arcs.

In this paper, we intend to get a better understanding of the space-time nature of
concurrency axioms, by establishing their interrelations and revealing their algebraic
lattice views, in the context of a subclass of relational structures with completely dis-
tinct, irreflexive relations on countable sets of elements.

2 Preliminaries

Introduce some notions and notations which will be useful throughout the text.

Sets and relations. Given a set \( X \) and a relation \( R \subseteq X \times X \),

- \( R \) is cyclic iff there exists a sequence of distinct elements \( x_1, \ldots, x_k \in X \) \((k > 1)\)
such that \( x_j R x_{j+1} \) \((1 \leq j \leq k - 1)\) and \( x_k R x_1 \),
- \( R \) is acyclic iff it is not cyclic,
- \( R \) is antisymmetric iff \((xR x') \land (x \neq x') \Rightarrow \neg(x' R x)\), for all \( x, x' \in X \),
- \( R \) is transitive iff \((x R x') \land (x' R x'') \Rightarrow (x R x'')\), for all \( x, x', x'' \in X \),
- \( R \) is irreflexive iff \((x R x)\), for all \( x \in R \),
- \( R^a = R \cup \text{id} \) (the reflexive closure of \( R \)),
- \( R^b = R \cup R^{-1} \) (the symmetric closure of \( R \)),
- \( R^c = R^b \cup \text{id} \) (the reflexive and symmetric closure of \( R \)),
- \( R^d = (R \setminus \text{id}) \setminus (R \setminus \text{id})^2 \) (the irreflexive, intransitive relation), if \( R \) is a transitive
  relation, and \( R^d = R \), otherwise,

Notice that if a relation \( R \) is irreflexive and transitive, then it is acyclic and anti-
symmetric, i.e. a (strict) partial order, and, moreover, \( R^d \) is the immediate predecessor
relation.

Given elements \( x, x_1, x_2 \in X \), subsets \( A \subseteq X' \subseteq X \), and a relation \( R \subseteq X \times X \),
- \([x_1 R x_2] = \{ x \in X \mid x_1 R^a x R^a x_2 \}\),
- \( R x = \{ x' \in X \mid x' R^0 x \} \), \( x R = \{ x' \in X \mid x R^d x' \} \),
\[ RA = \{ x' \in X \mid \exists x \in A : (x R^a x') \}, \quad A^R = \{ x' \in X \mid \exists x \in A : (x R^a x') \}, \]
\[ RA = \{ x' \in X \mid \forall x \in A : (x R x) \}, \quad A_R = \{ x' \in X \mid \forall x \in A : (x R x') \}, \]
\[ A is a (maximal) R-clique of X' if A is a (maximal) set containing only pairwise (R \cup \text{id}_{X'})-related elements of X', \]
\[ A is an R-closed set of X' if A_R = (A_R)_R = A. The family of all the R-closed sets of X' is denoted by RCl(X'). \]

**Partially ordered sets and lattices.** A partially ordered set (poset) is a set together with a partial order \( \leq \), i.e. a binary relation which is reflexive, transitive and antisymmetric. The powerset \( \mathcal{P}(X) \) of a set \( X \) together with inclusion \( \subseteq \) is a poset.

A mapping \( C : \mathcal{P}(X) \to \mathcal{P}(X) \) is a closure operator on a set \( X \) if for all \( A, B \subseteq X \) it holds:
1. \( A \subseteq C(A) \) (\( C \) is extensive),
2. \( A \subseteq B \Rightarrow C(A) \subseteq C(B) \) (\( C \) is monotonic),
3. \( C(C(A)) = C(A) \) (\( C \) is idempotent).

Let \( A \subseteq X \subseteq \mathcal{P}(X) \), and \( \mathcal{P} = (X, \subseteq) \) be a poset. Then,
\[ \mathcal{P} is a lattice if every two elements in \( X \) have both a g.l.b. (denoted \( \wedge \)) and a l.u.b. (denoted \( \vee \)). \]
\[ \mathcal{P} is a complete lattice if every subset of \( X \) has both a l.u.b. and a g.l.b., \]
\[ \mathcal{P} is algebraic if for all \( x \in X \) it holds \( x = \bigvee \{ k \in K(\mathcal{P}) \mid k \subseteq x \} \), where \( K(\mathcal{P}) \subseteq X \) denotes the set of compact elements of \( \mathcal{P} \). An element \( k \in X \) is said to be compact in the lattice \( \mathcal{P} \) if, for every \( S \subseteq X \) such that \( k \subseteq \bigvee S \), it holds that \( k \subseteq \bigvee T \), for some finite \( T \subseteq S \).

### 3 Relational Structures

Fig. 1.

In this section, we define a slight modification of the model of relational structures whose subclasses are put forward and studied in the papers [11–14], as a suitable model of structurally complex concurrent behaviors.

**Definition 1.** A relational structure is a tuple \( S = (E, V_1, \ldots, V_n) \) \( (n \geq 1) \), where
• $E$ is a countable set of elements,
• $V_1, \ldots, V_n \subseteq E \times E$ are irreflexive relations such that
  - $\bigcup_{1 \leq i \leq n} V_i^\beta = (E \times E) \setminus \text{id}$,
  - $V_i^\beta \cap V_j^\beta = \emptyset$, for all $1 \leq i \neq j \leq n$.

Clearly, Winskel’s event structures with forward hereditary conflict [17] and Boudol and Castellani’s event structures with forward and backward non-hereditary conflict [8] can be seen as relational structures with three relations, one of them is transitive (causality), and the two other are symmetric (concurrency and conflict).

**Example 1.** A simple example of a relational structure with four relations is shown in Fig. 1. Assume that $V_1$ is an irreflexive and transitive relation (partial order), $V_2$ is an asymmetric relation, and $V_3$ and $V_4$ are irreflexive and symmetric relations. We can interpret the relation $V_1$ as causality dependence, $V_2$ as asymmetric conflict [19, 7], $V_3$ as synchronous concurrency (simultaneity), and $V_4$ as asynchronous concurrency (independence).

From now on, we shall use $P$, $Q$ and $R$ to denote the relations on $E$ of the form $\bigcup_{V \in \mathcal{V}} V$, where $\mathcal{V} \subseteq \{V_i | 1 \leq i, j \leq n; V_i$ and $V_j$ possess the same relation properties}. We shall call a relational structure $S$ $\mathcal{P}$-finite iff any $P$-clique of $E$ is finite,
- $P$-degree-finite iff $|P_e \cup eP| < \infty$, for all $e \in E$,
- $P$-degree-restricted iff $|P_{e_1} \cap P_{e_2} \neq \emptyset \Rightarrow |P_{e_1}| = |P_{e_2}| = 1$, for all $e_1, e_2 \in E$,
- $P$-discrete iff $|P_{e_1} \cap P_{e_2} \cap E'| < \infty$, for all $e_1, e_2 \in E$ and $P$-cliques $E'$ of $E$,
- $P$-interval-finite iff $|P_{e_1} \cap P_{e_2}| < \infty$, for all $e_1, e_2$,
- $\forall_{PQR}$-free iff in any maximal $(P \cup Q \cup R)$-clique of $E$, there are no distinct elements $e_1, e_2$, and $e_3$ such that $e_1 P e_2 Q e_3 R e_1$.

From now on we shall consider only $P$-discrete relational structures, whenever $P$ is a transitive relation, and call them simply relational structures.

### 4 Concurrency Axioms

The notion of $K$-density and other concurrency axioms introduced by Petri [18] first for non-branching occurrence nets allow one to get better understanding the interaction of causality and concurrency. In [15], an analog of $K$-density under the name $L$-density has been put forward on the so called "sequential" nets with causality and symmetric hereditary conflict relations. Another analog under the name of $R$-density in the context of concurrent and conflict substructures of event structures has been dealt with in the paper [21]. More recently, the authors of [2] have adapted $K$-density to occurrence nets with symmetric hereditary conflicts, and rename it $B$-density. Our aim in this section is to give generalized definitions of a hierarchy of density properties and to study their interrelations, in the setting of relational structures.

Define concurrency axioms as properties of relational structures.
**Definition 2.** Given a relational structure $S$ and a maximal $(P \cup Q)$-clique $\tilde{E}$ of $E$,

- $\tilde{E}$ is $K_{PQ}$-dense iff for any maximal $P$-clique $E'$ of $\tilde{E}$ and for any maximal $Q$-clique $E''$ of $\tilde{E}$, $E' \cap E''$ is a (unique) maximal $(P \cap Q)$-clique of $\tilde{E}$.
- $\tilde{E}$ is $K_{PQ}$-crossing iff for any maximal $P$-clique $E'$ of $\tilde{E}$ and for any maximal $Q$-clique $E''$ of $\tilde{E}$, $E' \cap E'' \neq \emptyset$ and $E' \cap E'' \neq \emptyset$.
- $\tilde{E}$ is $\bowtie_{PQ}$-dense iff whenever $(e_0 \ P \ e_1 \ Q \ e_2)$ and $(e_0 \ Q \ e_3 \ P \ e_2)$, then $(e_0 \ P^\delta \ e_2) \implies (e_3 \ P \ e_1)$, for all distinct elements $e_0, e_1, e_2, e_3 \in \tilde{E}$.
- $\tilde{E}$ is $\bowtie_{PQ, Q}$-dense iff whenever $(e_0 \ P^\beta \ e_1 \ Q^\beta \ e_2)$ and $(e_0 \ Q^\beta \ e_3 \ P^\beta \ e_2)$, then $(e_0 \ P^\beta \ e_2) \implies (e_3 \ P^\beta \ e_1)$, for all distinct elements $e_0, e_1, e_2, e_3 \in \tilde{E}$.
- $S$ is $K_{PQ}$-dense ($K_{PQ}$-crossing, $\bowtie_{PQ}$-dense, $\bowtie_{PQ, Q}$-dense, respectively) iff any maximal w.r.t. $E$ clique $\tilde{E}$ of $(P \cup Q)$ is $K_{PQ}$-dense ($K_{PQ}$-crossing, $\bowtie_{PQ}$-dense, $\bowtie_{PQ, Q}$-dense, respectively).
Fig. 3.

The following results state the relationships between the properties defined so far.

**Proposition 1.** Let $S$ be a relational structure with distinct relations $P$ and $Q$, and let $P$ be a transitive or symmetric relation and $Q$ a symmetric relation. Then,

$$S \text{ is } \sqcap_{PQ}^\beta \text{-dense } \iff S \text{ is } \sqcap_{PQ} \text{-dense}.$$

**Example 2.** Consider the relational structures $S_2$–$S_6$ shown in Fig. 2. It is easy to see that $S_2 = (E_2, V', V'')$, with the transitive or symmetric relation $V'$ and the symmetric relation $V''$, is $\sqcap_{V', V''}^\beta$- and $\sqcap_{V', V''}$-dense. On the other hand, the relational structure $S_3 = (E_3, V', V'', V''')$, with the transitive or symmetric relation $V'$ and the symmetric relation $V'''$, is neither $\sqcap_{V', V'''}$- nor $\sqcap_{V', V'''}^\beta$-dense because in the maximal $(V' \cup V''')$-clique $\{e_2, e_3, e_4, e_5, e_7, e_8, e_9, e_{10}\}$ of $E_3$ there are distinct elements $e_2, e_3, e_4, e_5$ such that $(e_2 V' e_3 V''' e_4)$, $(e_2 V''' e_5 V' e_4)$, and $(e_2 (V')^\beta e_4)$ but $\neg (e_5 V' e_3)$. The relational structure $S_4 = (E_4, V', V'')$ with the non-transitive and non-symmetric relation $V'$ and the symmetric relation $V''$, is $\sqcap_{V', V''}$-dense but not $\sqcap_{V', V''}^\beta$-dense. Indeed, in the maximal $(V' \cup V'')$-clique $\{e_1, \ldots, e_6\}$ of $E_4$ there are elements $e_1, e_2, e_3, e_4$ such that $(e_2 (V')^\beta e_1 (V'')^\beta e_3)$, $(e_2 (V'')^\beta e_4 (V')^\beta e_3)$, and $(e_2 ((V')^\beta)^\beta e_3)$ but $\neg (e_4 (V')^\beta e_1)$. The same holds for the relational structure $S_5 = (E_5, V', V'')$ with the transitive or symmetric relation $V'$ and the asymmetric relation $V''$. Truly, in the maximal $(V' \cup V'')$-clique $\{e_1, \ldots, e_4\}$ of $E_5$ there are distinct elements $e_1, e_2, e_3, e_4$ such that $(e_1 (V')^\beta e_2 (V'')^\beta e_4)$, $(e_1 (V'')^\beta e_3 (V')^\beta e_4)$, and
(e_1 (V')\delta e_2) \text{ but } -(e_3 (V')\beta e_2). \text{ The relational structure } S_0 = (E_0, V', V'') \text{ with the non-transitive and non-symmetric relation } V' \text{ is } \triangleright_{V', V''}^\beta \text{-dense but not } \triangleright_{V', V''}^\alpha \text{-dense. In fact, in the maximal } (V' \cup V'') \text{-clique } \{e_1, \ldots, e_4\} \text{ of } E \text{ there are distinct elements } e_1, e_2, e_3, e_4 \text{ such that } (e_1 V' e_2 V' e_4), (e_1 V' e_3 V' e_4), \text{ and } (e_1 V'\delta e_4) \text{ but } -(e_3 V' e_2).

**Proposition 2.** Let } S \text{ be a } \triangleright_{PQ}^\beta \text{-dense relational structure with distinct relations } P \text{ and } Q, \text{ and let } P \text{ be a transitive relation. Then,}

\[ S \text{ is } K_{PQ} \text{-dense } \iff S \text{ is } K_{PQ} \text{-crossing.} \]

**Example 3.** First, consider the relational structures \( S_2 = (E_2, V', V'') \) and \( S_3 = (E_3, V', V'', V''') \) shown in Fig. 2. We know that \( S_2 \) with the transitive relation \( V' \) and the symmetric relation \( V'' \), is \( \triangleright_{V', V''}^\beta \)-dense (see Example 2). It is easy to check that \( S_2 \) is \( K_{V', V''} \)-dense and \( K_{V', V''} \)-crossing. On the other hand, the relational structure \( S_3 \), with the transitive relation \( V' \) and the symmetric relation \( V''' \), is not \( \triangleright_{V', V''}^\beta \)-dense (see Example 2). It is easy to verify that \( S_3 \) is \( K_{V', V''} \)-dense. Moreover, \( S_3 \) is not \( K_{V', V''} \)-dense because in the maximal \( (V' \cup V'') \)-clique \( E = \{e_2, e_3, e_4, e_5, e_7, e_8, e_9, e_{10}\} \) of \( E \), the intersection of the maximal \( V' \)-clique \( \{e_2, e_4, e_7, e_{10}\} \) of \( E \) with the maximal \( V'' \)-clique \( \{e_3, e_5\} \) of \( E \) is empty. Next, contemplate the relational structures \( S_7 = (E_7, V', V'') \) and \( S_8 = (E_8, V', V'') \) depicted in Fig. 3. The relational structure \( S_7 \) with the transitive relation \( V' \) and the symmetric relation \( V'' \), is \( \triangleright_{PQ}^\beta \)-dense but neither \( K_{V', V''} \)-dense nor \( K_{V', V''} \)-crossing since in the maximal \( (V' \cup V'') \)-clique \( E = \{e_1, e_2, \ldots\} \) of \( E \) the intersection of the maximal \( V' \)-clique \( E' = \{e_{2k+1} \mid k \geq 0\} \) of \( E \) with the maximal \( V'' \)-clique \( E'' = \{e_{2k} \mid k \geq 1\} \) of \( E \) is empty, and, moreover, the intersection of \( E' \) with \( E'' \) is also empty, because \( E'' = E'' \). Further, the relational structure \( S_8 \), with the non-transitive relation \( V' \) and the symmetric relation \( V'' \), is \( \triangleright_{PQ}^\beta \)-dense and \( K_{V', V''} \)-crossing but not \( K_{V', V''} \)-dense because in the maximal \( (V' \cup V'') \)-clique \( E = \{e_1, e_2, \ldots\} \) of \( E \) the intersection of the maximal \( V' \)-clique \( \{e_{2k+1} \mid k \geq 0\} \) of \( E \) with the maximal \( V'' \)-clique \( \{e_{2k} \mid k \geq 1\} \) of \( E \) is empty.

**Theorem 1.** Let \( S \) be a \( P \)- or \( Q \)-finite relational structure with distinct relations \( P \) and \( Q \), and let \( P \) be a transitive relation. Then,

\[ S \text{ is } K_{PQ} \text{-dense } \iff S \text{ is } \triangleright_{PQ}^\beta \text{-dense.} \]

**Example 4.** First, again consider the relational structures \( S_2 = (E_2, V', V'') \) and \( S_3 = (E_3, V', V'', V''') \) shown in Fig. 2. We know that \( S_2 \), with the transitive relation \( V' \) and the symmetric relation \( V'' \), is \( \triangleright_{V', V''}^\beta \)-dense (see Example 2) and \( K_{V', V''} \)-dense (see Example 3). Notice that \( S_2 \) is \( V' \)- and \( V'' \)-finite. On the other hand, the relational structure \( S_3 \), with the transitive relation \( V' \) and the symmetric relation \( V'' \), is neither \( \triangleright_{V', V''}^\beta \)-dense (see Example 2) nor \( K_{V', V''} \)-dense (see Example 3). Moreover, \( S_3 \) is \( V' \)- and \( V'' \)-finite. Next, contemplate the relational structures \( S_8 = (E_8, V', V'') \) and \( S_9 = (E_9, V', V'') \) depicted in Fig. 3. We know from Example 3 that \( S_8 \), with the transitive relation \( V' \) and the symmetric relation \( V'' \), is \( \triangleright_{PQ}^\beta \)-dense but not \( K_{PQ} \)-dense. At the same time, \( S_8 \) is neither \( V' \)- nor \( V'' \)-finite. The relational structure \( S_9 \),
with the non-transitive relation $V'$ and the symmetric relation $V''$, is $K_{V',V''}$-dense but not $\nabla_{V',V''}$-dense because in the maximal $(V' \cup V'')$-clique $\{e_1, \ldots, e_5\}$ of $E_9$, there are distinct elements $e_1, e_2, e_3, e_4$ such that $e_1 V' e_2 V'' e_4, e_1 V'' e_3 V' e_4$, and $e_1 V' e_4$ but $\neg(e_3 V' e_2)$.

**Theorem 2.** Given a $\nabla_{PQR}$-free and $P$- or $Q$- or $R$-finite relational structure $S$ with distinct relations $P$, $Q$, and $R$,

$$S \text{ is } K_{PQ}\text{-dense } \iff S \text{ is } K_{\overline{PQ}}\text{-dense},$$

where $\overline{P} = (P \cup R)$ and $\overline{Q} = (Q \cup R)$.

**Example 5.** Consider the relational structures $S_{10}$-$S_{13}$, with the transitive relation $V'$ and the symmetric relations $V''$ and $V'''$, shown in Fig. 4. It is easy to check that $S_{10} = (E_{10}, V', V'', V''')$ is $K_{V'',V'''}$-dense, $\nabla_{V',V''}V'''$-free, and $K_{V',V''}$-dense, where $\tilde{V}'' = V' \cup V''$ and $\tilde{V}''' = V' \cup V'''$. Clearly, $S_0$ is $V'$- and $V''$- and $V'''$-finite. However, if $V'$- and $V'''$-dense. Hence, $S_{11}$ is neither $\nabla_{V',V''}V'''$-free, because $e_1 V' e_4 V'' e_5 V''' e_1$, nor $K_{V',V''}$-dense, because the intersection of the maximal $V''$-clique $\{e_3, e_4, e_7\}$ of $\tilde{E}$ and the maximal $\tilde{V}'''$-clique $\{e_1, e_3, e_6\}$ of $\tilde{E}$ is not a maximal $V'$-clique of $\tilde{E}$ where $\tilde{V}'' = V' \cup V''$ and $\tilde{V}''' = V' \cup V'''$, and $\tilde{E} = \{e_1, \ldots, e_7\}$ is the only maximal $\tilde{V}''$- and $\tilde{V}'''$-clique of $E_{11}$. The relational structure $S_{12} = (E_{12}, V', V'', V''')$ is $\nabla_{V',V''}V'''$-free but neither $K_{V',V''}$-dense, because the intersection of the maximal $V'$-clique $\{e_1, e_4\}$ of $\tilde{E}'$ with the maximal $V'''$-clique $\{e_2, e_3\}$ of $\tilde{E}'$ is empty, nor $K_{V',V''}$-dense, because the intersection of $\tilde{E'}$ and $\tilde{E}''$ is not $V'$- and $V'''$-dense. Hence, $S_{13}$ is not $K_{V',V''}$-dense.

# 5 Concurrency Axioms and Algebraic Lattices of Closed Sets

In the papers [1, 2], it has been demonstrated that $K$-density of occurrence nets with and without forward conflicts guarantees that the lattices whose elements are the closed subsets of a closure operator, defined starting from the concurrency relation of the models, are algebraic. In this section, we first show that the above results can be extended to the model of relational structures, where a closure operator can be defined from any symmetric relation, and then formulate a necessary condition of the algebraicity of the lattices of the closed sets. Before doing so, we need to introduce the following concepts.

**Definition 3.** Given a relational structure $S$ and a maximal $(P \cup Q)$-clique $\tilde{E}$ of $E$, ...
\$E\$ is $PQ$-encountering iff for any maximal $P$-clique $E'$ of $\tilde{E}$ and for any maximal $Q$-clique $E''$ of $\tilde{E}$, $E' \cap E'' \neq \emptyset$, for some finite $E''' \subseteq E''$. 
• $\tilde{E}$ is weak $K_{PQ}$-dense iff for any maximal $P$-clique $E'$ of $\tilde{E}$ and for any maximal $Q$-clique $E''$ of $\tilde{E}$, $(E' \cap E'')$ is a (unique) $(P \cap Q)$-clique of $\tilde{E}$, or $A \not\subseteq E'$, for any $Q$-closed set $A$ of $\tilde{E}$.
• $\tilde{E}$ is $PQ$-algebraic iff $(\text{QCl}(\tilde{E}), \subseteq)$ is an algebraic lattice.
• $S$ is $PQ$-encountering (weak $K_{PQ}$-dense, $PQ$-algebraic, respectively) iff any maximal $(P \cup Q)$-clique $\tilde{E}$ of $E$ is $PQ$-encountering (weak $K_{PQ}$-dense, $PQ$-algebraic, respectively).

Fig. 5.

The following fact will be helpful to obtain weak $K_{PQ}$-density as a necessary condition of $PQ$-algebraicity.

**Theorem 3.** Given a $\triangleright_{PQ}$-dense relational structure $S$ with a transitive relation $P$ and a symmetric relation $Q$, $S$ is $K_{PQ}$-dense $\iff$ $S$ is $PQ$-encountering.

**Example 6.** Consider the relational structures $S_{14} = (E_{14}, V', V'')$ and $S_{15} = (E_{15}, V', V''')$ with the transitive relation $V'$ and the symmetric relation $V''$, shown in Fig. 5. One can easily check that $S_{14}$ is $V'V''$-encountering, $\triangleright_{V'V''}$- and $K_{V'V''}$-dense. On the other hand, $S_{15}$ is $V'V'''$-encountering and, obviously, not $\triangleright_{V'V''}$-dense. Moreover, as the maximal $V'$-clique $\{e_1, e_4\}$ and the maximal $V'''$-clique $\{e_2, e_3\}$ of $E_{15}$ are disjoint, $S_{14}$ is not $K_{V'V''}$-dense. We know from Example 3 that the relational structure $S_7$ depicted in Fig. 3 is $\triangleright_{V'V''}$-dense but not $K_{V'V'''}$-dense. Furthermore, $S_7$ is not $V'V'''$-encountering, because for the maximal $V'$-clique $E' = \{e_1, e_3, e_5, \ldots\}$ and the maximal $V'''$-clique $E''' = \{e_2, e_3, e_6, \ldots\}$ of $E_7$, we have $E' \cap E''' = \emptyset$ for any finite $E''' \subseteq E''$.

Finally, the following theorem describes interconnections between the properties of $K_{PQ}$-density and weak $K_{PQ}$-density, and $PQ$-algebraicity of a relational structure.

**Theorem 4.** Given a $P$-degree-restricted, $P$-degree-finite and $P$-interval-finite relational structure $S$ with a transitive relation $P$ and a symmetric relation $Q$,

(i) $S$ is $K_{PQ}$-dense $\implies$ $S$ is $PQ$-algebraic,
(ii) $S$ is weak $K_{PQ}$-dense $\iff$ $S$ is $PQ$-algebraic.
Example 7. Contemplate the $V'$-degree-restricted relational structures $S_{16}$–$S_{18}$, with the transitive relation $V'$ and the symmetric relation $V''$, depicted in Fig. 6. The relational structure $S_{16} = (E_{16}, V', V'')$ is, obviously, $V'$-degree-finite, $V'$-interval-finite and $K_{V'V''}$-dense. Since it is finite, it is $V'V''$-algebraic. Next, consider the $V'$-degree-finite relational structure $S_{17} = (E_{17}, V', V'')$ with the maximal $V'$-clique $E' = \{e_{2i+1} \mid i \geq 0\}$ of $E_{17}$ and the maximal $V''$-clique $E'' = \{e_{2j} \mid j \geq 1 \land j \neq 2\}$ of $E_{17}$. Since $E' \cap E'' = \emptyset$, $S_{17}$ is not $K_{V'V''}$-dense. Moreover, $S_{17}$ is not weak $K_{V'V''}$-dense, because $A = \{e_3\}$ is a $V''$-closed set of $E_{17}$ such that $A \not\subseteq E'$. Furthermore, $A$ is not compact in the lattice $(V''\text{Cl}(E_{17}), \subseteq)$. This implies that $\emptyset$ is the only compact element less than $A$. As $\bigvee \emptyset = \emptyset \neq A$, $S_{17}$ is not $V'V''$-algebraic. Finally, consider the non-$V'$-degree-finite relational structure $S_{18}$. One can easily check that $S_{18}$ is $K_{V'V''}$-dense and $V'$-interval-finite. As the $V''$-closed set $A = \{e_1\}$ is not compact in the lattice $(V''\text{Cl}(E_{18}), \subseteq)$, $S_{18}$ is not $V'V''$-algebraic.
References

5. E. Best. A theorem on the characteristics of non-sequential processes. Fundamenta Informaticae 3 (1980) 77-94
From Free Will Debate to Embodiment of Fuzzy Logic into Washing Machines: On Fuzzy and Rough Sets Approaches to Vagueness Modeling

Extended Abstract

Piotr Wasilewski
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
piotr@mimuw.edu.pl

Deep scientific ideas have at least one distinctive property: they can be applied both by philosophers in abstract fundamental debates and by engineers in concrete practical applications. Mathematical approaches to modeling of vagueness also possess this property. Problems connected with vagueness have been discussed at the beginning of XXth century by philosophers, logicians and mathematicians in developing foundations of mathematics leading to clarification of logical semantics and establishing of mathematical logic and set theory. Those investigations led also to big step in the history of logic: introduction of three-valued logic. In the second half of XXth century some mathematical theories based on vagueness idea and suitable for modeling vague concepts were introduced, including fuzzy set theory proposed by Lotfi Zadeh in 1965 [16] and rough set theory proposed by Zdzisław Pawlak in 1982 [4] having many practical applications in various areas from engineering and computer science such as control theory, data mining, machine learning, knowledge discovery, artificial intelligence.

Concepts in classical philosophy and in mathematics are not vague. Classical theory of concepts requires that definition of concept C hast to provide exact rules of the following form:

if object \( x \) belongs to concept \( C \), then \( x \) possess properties \( P_1, P_2, \ldots, P_n \);

if object \( x \) possess properties \( P_1, P_2, \ldots, P_n \), then \( x \) belongs to concept \( C \).

In classical set theory concepts are sets and their non-vagueness (crispness) is expressed by characteristic functions: with every set \( A \subseteq U \) there is function \( \chi_A : U \rightarrow \{0, 1\} \) such that \( \chi_A(x) = 1 \) iff \( x \in A \), otherwise \( \chi_A(x) = 0 \).

Lotfi Zadeh introduced fuzzy sets as generalizations of characteristic functions together with operations based on Łukasiewicz’s logical operations taken from three-valued logic: a fuzzy set \( X \) consisting of object from domain \( U \) is defined by membership functions \( \mu_X : U \rightarrow [0, 1] \), where \( \mu_X(a) \) reflects a grade/degree in which object

\* The research has been supported by grant 2011/01/D/ST6/06981 from the Polish National Science Centre and by grant 2011-3/12 from Homing Plus programme of Foundation for Polish Science, co-financed from EU, Regional Development Fund. Author would like to express his gratitude to Andrzej Skowron for his inspiration and ceaseless support during conducting research presented in the paper.
a belongs to fuzzy set $X$ what corresponds to vagueness of concepts. In particular if
$\mu_X(a) = 0$, then object $a$ does not belong to fuzzy set $X$ and if $\mu_X(a) = 1$, then
object $a$ belongs to fuzzy set $X$ in full degree what corresponds to classical notion of
belonging to a set.

In rough set theory concepts are represented by subsets of a given space $U$ and
objects are represented by granules, some collections of objects, in classical rough set
theory these granules are equivalence classes of some equivalence relation $R$ on $U$. In
rough set theory sets are represented and analyzed by two operators: lower and upper
approximations, denoted respectively by $R_*$, $R^*$ and defined for set $X \subseteq U$ as follows:

$$R_*(X) = \bigcup \{ Y \in U/R : Y \subseteq X \} \quad R^*(X) = \bigcup \{ Y \in U/R : Y \cap X \neq \emptyset \}.$$  

Set $X \subseteq U$ is rough iff $R_*(X) \neq R^*(X)$. With every set $X \subseteq U$ there are associated three sets called regions:

positive region

$$POS(X) := R_*(X),$$

negative region

$$NEG(X) := U \setminus R^*(X) = R_*(X') = \bigcup \{ E \in U/R : E \cap X = \emptyset \}$$

where $X' = U \setminus X$ and boundary region

$$BND(X) := R^*(X) \setminus R_*(X) = U \setminus (POS(X) \cup NEG(X)).$$

Set $X \subseteq U$ is rough iff $BND(X) \neq \emptyset$. In the case of any set $X \subseteq U$ positive region
of $X$ can be interpreted as a set of objects from $U$ which surely belongs to $X$, negative
region can be interpreted as a set of objects from $U$ which surely do not belong to $X$, whereas boundary region can be interpreted as a set of objects from $U$ which possibly
belong to $X$.

One can note that both approaches to modelling vagueness are some generalizations
in which crisp sets are particular cases: in fuzzy set theory fuzzy set $X$ of objects from
$U$ is crisp iff for all $a \in U$ either $\mu_X(a) = 1$ or $\mu_X(a) = 0$; in rough set theory set
$X \subseteq U$ is crisp iff $BND(X) = \emptyset$. Both theories are also essentially connected with
Łukasiewicz’s ideas.

The main object of the paper is to present and compare fuzzy sets and rough
sets approaches to vagueness and uncertainty modelling and analysis, in particular we
will discuss representation of vague concepts in both theories. We will also present
Łukasiewicz’s arithmetization of propositional calculus semantics and Łukasiewicz’s
involvement in discussion on meaning and logical values of propositions about future
which led to introduction of the third logical value and to proposing three-valued logic.
Our comparison of fuzzy set theory and rough set theory approaches to vagueness
modelling will be made with respect to a characterization of vagueness proposed in contem-
porary philosophy. This characterization includes a second order vagueness condition
which, roughly speaking, requires that a boundary of a concept cannot be a crisp set.
We will conclude the paper with presenting and discussion solutions to that problem
in the rough set approach to vagueness modelling including that proposed by Andrzej
References

Global Optimization of Exact Association Rules Relative to Length

Beata Zielosko
Institute of Computer Science
University of Silesia
39, Bedzinska St., 41-200 Sosnowiec, Poland
beata.zielosko@us.edu.pl

Abstract. In the paper, an application of dynamic programming approach to global optimization of exact association rules relative to length is presented. It is an extension of the dynamic programming approach to optimization of decision rules to inconsistent tables. An information system \( I \) is transformed into a set of decision tables \( \{I_{f_1}, \ldots, I_{f_{n+1}}\} \). The algorithm constructs, for each decision table from the set \( \{I_{f_1}, \ldots, I_{f_{n+1}}\} \), a directed acyclic graph \( \Delta(I_{f_i}) \), \( i = 1, \ldots, n+1 \). Based on the graph, the set of so-called irredundant \( (f_i) \)-association rules can be described. The union of sets of \( (f_i) \)-association rules, \( i = 1, \ldots, n+1 \), is considered as a set of association rules for information system \( I \). Then, global optimization relative to length is made and sets of association rules with minimum length, for each row of information system \( I \), are obtained. Preliminary experimental results with data sets from UCI Machine Learning Repository are included.

Key words: rough sets, association rules, length, dynamic programming, decision rules

1 Introduction

Association rule mining is one of the important fields of data mining and knowledge discovery. It aims to extract interesting correlations, associations, or frequent patterns among sets of items in data set.

There are many algorithms for construction of association rules. The most popular algorithm for mining association rules is Apriori algorithm [1]. During years, based on the Apriori, many new algorithms were designed with some modifications or improvements, e.g., hash based technique [15], transaction reduction [7] and others [8, 13, 18]. Another approaches are frequent pattern growth [5] that adopts divide and conquer strategy, and algorithms that uses vertical data format [6].

In the paper, an application of dynamic programming approach to optimization of exact association rules relative to length is presented. Construction of short rules is connected with the Minimum Description Length principle introduced by Rissanen [17]: the best hypothesis for a given set of data is the one that leads to the largest compression of the data. Short rules are more understandable and easier for interpreting by experts. Unfortunately, the problem of construction of rules with minimum length is
Application of rough sets theory to the construction of rules for knowledge representation or classification tasks are usually connected with the usage of decision table [16] as a form of input data representation. In such a table one attribute is distinguished as a decision attribute and it relates to a rule’s consequence. However, in the last years, associative mechanism of rule construction, where all attributes can occur as premises or consequences of particular rules, is popular. Association rules can be defined in many ways. In the paper, a special kind of association rules is studied, i.e., they relate to decision rules. Similar approach was considered in [11, 12], where greedy algorithm for minimization of length of association rules was studied.

This paper is an essential extension of the paper [3] in which consistent decision tables were considered, i.e., they do not contain equal rows with different decisions. When association rules for information systems are studied and each attribute is sequentially considered as the decision one, inconsistent tables are often obtained. So, the approach considered in [3] is extended to the case of inconsistent decision tables. It required changes in definitions, algorithms (new conditions of stop), proofs of algorithm correctness, and, especially, in the software.

Proposed approach by comparison with Apriori algorithm allows one to derive a required number of rules for a given row only. If we consider sequential optimization relative to coverage and length it is possible to find so-called totally-optimal rules, i.e., rules with maximum coverage and minimum length. Experimental resuls show that such rules have often good coverage and small length.

The aim of the paper is to create a research tool which is applicable to medium sized decision tables and allows one to construct exact association rules with minimum length. To this end, an information system $I$ with attributes $\{f_1, \ldots, f_{n+1}\}$ is transformed into a set of decision tables $\{I_{f_1}, \ldots, I_{f_{n+1}}\}$. The algorithm constructs, for each decision table from the set $\{I_{f_1}, \ldots, I_{f_{n+1}}\}$, a directed acyclic graph $\Delta(I_{f_i})$, $i = 1, \ldots, n + 1$. Based on the graph $\Delta(I_{f_i})$, the set of so-called irredundant $(f_i)$-association rules, $i = 1, \ldots, n + 1$, can be described. The union of sets of irredundant $(f_i)$-association rules, $i = 1, \ldots, n + 1$, is considered as the set of irredundant association rules for information system $I$. Using global optimization relative to length it is possible to obtain the set of irredundant association rules for information system $I$ with minimum length. In [20], global optimization of association rules relative to coverage was presented.

The paper consists of six sections. Section 2 contains main notions. In Section 3, an algorithm for construction of a directed acyclic graph is presented. Section 4 contains a description of optimization procedure relative to length. Section 5 contains experimental results for decision tables from UCI Machine Learning Repository, and Section 6 - conclusions.
2 Main Notions

An information system $I$ is a table with $n + 1$ columns labeled with attributes $f_1, \ldots, f_{n+1}$. Rows of this table are filled by nonnegative integers which are interpreted as values of attributes.

An association rule for $I$ is a rule of the kind

$$f_{i_1} = a_1 \land \ldots \land f_{i_m} = a_m \rightarrow f_j = a,$$

where $f_j \in \{f_1, \ldots, f_{n+1}\}$, $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_{n+1}\} \setminus \{f_j\}$, and $a, a_1, \ldots, a_m$ are nonnegative integers.

The notion of an association rule for $I$ is based on the notion of a decision table and decision rule.

A decision table $T$ is a table with $n$ columns labeled with (conditional) attributes $f_1, \ldots, f_n$. Rows of this table are filled by nonnegative integers which are interpreted as values of conditional attributes. Each row is labeled with a nonnegative integer (decision) which is interpreted as a value of a decision attribute. It is possible that $T$ contains equal rows with the same or different decisions.

For each attribute $f_i \in \{f_1, \ldots, f_{n+1}\}$, the information system $I$ is transformed into a decision table $I_{f_i}$. The column $f_i$ is removed from $I$ and a table with $n$ columns labeled with attributes $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_{n+1}$ is obtained. Values of the attribute $f_i$ are attached to the rows of the obtained table which will be denoted by $I_{f_i}$.

\[
\begin{array}{ccc}
I &=& \begin{array}{ccc}
  f_1 & f_2 & f_3 \\
  r_1 & 1 & 1 & 2 \\
r_2 & 0 & 0 & 2 \\
r_3 & 1 & 1 & 1 \\
\end{array} \\
\Rightarrow I_{f_1} &=& \begin{array}{ccc}
  f_2 & f_3 \\
  r_1 & 1 & 2 & 1 \\
r_2 & 0 & 2 & 0 \\
r_3 & 1 & 1 & 1 \\
\end{array}
\end{array}
\]

Fig. 1. Decision table $I_{f_1}$ obtained from information system $I$

The set $\{I_{f_1}, \ldots, I_{f_{n+1}}\}$ of decision tables obtained from the information system $I$ is denoted by $\Phi$. Let $T \in \Phi$. For simplicity, it is assumed that $T = I_{f_{n+1}}$.

The table $T$ is called degenerate if it is empty or all rows of $T$ are labeled with the same decision, or all rows of $T$ are equal.

A minimum decision value that is attached to the maximum number of rows in $T$ will be called the most common decision for $T$.

A table obtained from $T$ by the removal of some rows is called a subtable of the table $T$. Let $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_n\}$ and $a_1, \ldots, a_m$ be nonnegative integers. A subtable of the table $T$, which contains only rows of $T$ that have numbers $a_1, \ldots, a_m$ at the intersection with columns $f_{i_1}, \ldots, f_{i_m}$, is denoted by $T(f_{i_1}, a_1) \ldots (f_{i_m}, a_m)$. Such subtables (including the table $T$) are called separable subtables of $T$.

The set of attributes from $\{f_1, \ldots, f_n\}$ which are nonconstant in $T$ is denoted by $E(T)$. For any $f_i \in E(T)$, the set of values of the attribute $f_i$ in $T$, is denoted by $E(T; f_i)$. 
The expression
\[ f_{i_1} = a_1 \land \ldots \land f_{i_m} = a_m \rightarrow f_{n+1} = d \]  
(1)
is called a decision rule over \( T \) if \( f_{i_1}, \ldots, f_{i_m} \in \{ f_1, \ldots, f_n \} \), and \( a_1, \ldots, a_m, d \) are nonnegative integers. It is possible that \( m = 0 \). In this case (1) is equal to the rule
\[ \rightarrow f_{n+1} = d. \]  
(2)

Let \( r = (b_1, \ldots, b_n) \) be a row of \( T \). Rule (1) will be called realizable for \( r \), if \( a_1 = b_1, \ldots, a_m = b_m \). If \( m = 0 \) then rule (2) is realizable for any row from \( T \).

Rule (1) will be called true for \( T \) if the table \( T' = T(f_{i_1}, a_1) \ldots (f_{i_m}, a_m) \) is degenerate and \( d \) is the most common decision for \( T' \). If \( m = 0 \) then rule (2) is true for \( T \) if \( T \) is degenerate and \( d \) is the most common decision for \( T \).

If rule (1) is true for \( T \) and realizable for \( r \), then (1) will be called a decision rule for \( T \) and \( r \).

Decision rules for \( T \) and \( r \) will be called \((f_{n+1})\)-association rules for \( I \) and \( r \). In general case, the notion of \((f_i)\)-association rule for \( I \) and \( r \) coincides with the notion of decision rule for \( I_f \) and \( r \), \( i = 1, \ldots, n+1 \). The union of sets of \((f_i)\)-association rules, \( i = 1, \ldots, n+1 \), will be considered as the set of association rules for \( I \) and \( r \).

Let \( T = I_{f_n+1} \) and (1) be a decision rule over \( T \). Rule (1) will be called an irredundant rule for \( T \) and \( r \) if (1) is a decision rule for \( T \) and \( r \) and the following conditions hold if \( m > 0 \):

(i) \( f_{i_1} \in E(T) \), and if \( m > 1 \) then \( f_{i_j} \in E(T(f_{i_1}, a_1) \ldots (f_{i_{j-1}}, a_{j-1})) \) for \( j = 2, \ldots, m \);

(ii) if \( m = 1 \) then the table \( T \) is nondegenerate, and if \( m > 1 \) then the table \( T(f_{i_1}, a_1) \ldots (f_{i_{m-1}}, a_{m-1}) \) is nondegenerate.

If \( m = 0 \) then rule (2) is an irredundant decision rule for \( T \) and \( r \) if (2) is a decision rule for \( T \) and \( r \), i.e., if \( T \) is degenerate and \( d \) is the most common decision for \( T \).

Let \( T = I_{f_n+1} \), \( \tau \) be a decision rule over \( T \), and \( \tau \) be equal to (1). The length of \( \tau \) is the number \( m \) of descriptors (pairs attribute=value) on the left-hand side of \( \tau \). It is denoted by \( l(\tau) \). If \( m = 0 \) then the length of the rule \( \tau \) is equal to 0.

### 3 Algorithm for Directed Acyclic Graph Construction

In this section, an algorithm for construction of a directed acyclic graph for a given decision table \( T \) is presented. Based on this graph it is possible to describe the set of irredundant rules for \( T \) and for each row \( r \) of \( T \). This algorithm is repeated for each decision table \( I_{f_i} \), \( i = 1, \ldots, n+1 \), obtained from the information system \( I \).

Let \( T = I_{f_n+1} \). The constructed graph is denoted by \( \Delta(T) \). Nodes of the graph are some separable subtables of the table \( T \). During each step, the algorithm processes one node and marks it with the symbol *. At the first step, the algorithm constructs a graph containing a single node \( T \) which is not marked with *. Let the algorithm have already performed \( p \) steps. Now the step \((p+1)\) will be described. If all nodes are marked with the symbol * as processed, the algorithm finishes its work and presents the resulting graph as \( \Delta(T) \). Otherwise, choose a node (table) \( \Theta \), which has not been processed yet. If \( \Theta \) is degenerate then mark the considered node with symbol * and proceed to the step
is degenerate. As a result, a graph $G(\Theta)$, with the same sets of nodes and edges as in $\Delta(T)$, is obtained. The only difference is that any row $r$ of each nonterminal node $\Theta$ from $G(T)$ is labeled with a nonempty set of attributes $E_{G(T)}(\Theta, r) \subseteq E(\Theta)$, possibly different from $E(\Theta)$.

Let $G \in \{\Delta(T), G(T)\}$. Now, for each node $\Theta$ of $G$ and for each row $r$ of $\Theta$, a set of rules $\text{Rul}_G(\Theta, r)$ will be described.

Let $\Theta$ be a terminal node of $G$. In this case $\Theta$ is a degenerate table and

$$\text{Rul}_G(\Theta, r) = \{ \rightarrow f_{n+1} = d \},$$

where $d$ is the most common decision for $\Theta$.

Let now $\Theta$ be a nonterminal node of $G$ such that, for each child $\Theta'$ of $\Theta$ and for each row $r'$ of $\Theta'$, the set of rules $\text{Rul}_G(\Theta', r')$ is already defined. Let $r = (b_1, \ldots, b_n)$ be a row of $\Theta$. For any $f_i \in E_G(\Theta, r)$, the set of rules $\text{Rul}_G(\Theta, r, f_i)$ is defined as follows:

$$\text{Rul}_G(\Theta, r, f_i) = \{ f_i = b_i \land \gamma \rightarrow f_{n+1} = s : \gamma \rightarrow f_{n+1} = s \in \text{Rul}_G(\Theta(b_i), r) \}.$$  

Then $\text{Rul}_G(\Theta, r) = \bigcup_{f_i \in E_G(\Theta, r)} \text{Rul}_G(\Theta, r, f_i)$.

One can prove the following statement.

**Theorem 1.** For any node $\Theta$ of $\Delta(T)$ and for any row $r$ of $\Theta$, $\text{Rul}_{\Delta(T)}(\Theta, r)$ is equal to the set of all irredundant rules for $\Theta$ and $r$.

The algorithm for the directed acyclic graph construction is repeated for each decision table $I_{f_i}, i = 1, \ldots, n+1$, obtained from the information system $I$. In general, the obtained graph is denoted by $\Delta(I_{f_i}), i = 1, \ldots, n+1$. As a result, for $i = 1, \ldots, n+1$, the set $\text{Rul}_{\Delta(I_{f_i})}(I_{f_i}, r)$ of irredundant decision rules for $I_{f_i}$ and $r$ is obtained. This set will be called the set of irredundant $(f_i)$-association rules for $I$ and $r$, $i = 1, \ldots, n+1$. The union of sets $\text{Rul}_{\Delta(I_{f_i})}(I_{f_i}, r)$ forms the set $\text{Rul}(I, r)$ of irredundant association rules for $I$ and $r$:

$$\text{Rul}(I, r) = \bigcup_{i=1,\ldots,n+1} \text{Rul}_{\Delta(I_{f_i})}(I_{f_i}, r).$$

**Example 1.** To illustrate the presented algorithm the information system $I$ depicted in Fig. 1 will be considered. Set $\Phi = \{I_{f_1}, I_{f_2}, I_{f_3}\}$ contains three decision tables obtained from $I$. Figure 2 presents a directed acyclic graph for decision table $I_{f_1}$. Based on the graph $\Delta(I_{f_1})$ the sets of rules attached to rows of $I_{f_1}$ are described.

$\text{Rul}_{\Delta(I_{f_1})}(I_{f_1}, r_1) = \{ f_2 = 1 \rightarrow f_1 = 1, f_3 = 2 \land f_2 = 1 \rightarrow f_1 = 1 \}$:
Let \( f(\Theta, r) = \) the number of irredundant association rules for \( \Theta, r \).

\[ f^i(\Theta, r) = \begin{cases} 1 & \text{if } f(\Theta, r) = 1, \\ 0 & \text{otherwise}. \end{cases} \]

Global optimization relative to the length is made for an information system \( I \), i.e., among all graphs \( G(I_{f_1}), \ldots, G(I_{f_{n+1}}) \) constructed for decision tables from the set \( \Phi \) and optimized locally relative to the length.

Let \( T = I_{f_{n+1}} \). Now, a procedure of local optimization of the graph \( \Delta(T) \) relative to the length will be described. For each node \( \Theta \) in the graph \( \Delta(T) \), this procedure assigns to each row \( r \) of \( \Theta \) the set \( Rul^{l\Delta(T)}(\Theta, r) \) of decision rules with the minimum length from \( Rul_{\Delta(T)}(\Theta, r) \) and the number \( Opt_l^{\Delta(T)}(\Theta, r) \) – the minimum length of decision rule from \( Rul_{\Delta(T)}(\Theta, r) \).

The algorithm moves from the terminal nodes of the graph \( \Delta(T) \), which are degenerate tables, to the node \( T \). It will attach the set \( E_{G(T)}(\Theta, r) \) to each row \( r \) in \( \Theta \) if \( \Theta \) is a nonterminal node of \( \Delta(T) \). The obtained graph is denoted by \( G(T) \).

Let \( \Theta \) be a terminal node of \( \Delta(T) \). Then the number \( Opt^{l\Delta(T)}(\Theta, r) = 0 \) is assigned to each row \( r \) of \( \Theta \).
Let $\Theta$ be a nonterminal node of $\Delta(T)$ and all children of $\Theta$ have already been treated. Let $r = (b_1, \ldots, b_n)$ be a row of $\Theta$. The number

$$Opt^l_{\Delta(T)}(\Theta, r) = \min\{Opt^l_{\Delta(T)}(\Theta(f_i, b_i), r) + 1 : f_i \in E(\Theta, r)\}$$

is assigned to the row $r$ in the table $\Theta$ and

$$E_\Delta(\Theta, r) = \{f_i : f_i \in E_{\Delta(T)}(\Theta, r), Opt^l_{\Delta(T)}(\Theta(f_i, b_i), r) + 1 = Opt^l_{\Delta(T)}(\Theta, r)\}.$$

**Theorem 2.** For each node $\Theta$ of the graph $G(T)$ and for each row $r$ of $\Theta$, the set $Rul_{G(T)}(\Theta, r)$ is equal to the set $Rul^l_{\Delta(T)}(\Theta, r)$ of all rules with the minimum length from the set $Rul_{\Delta(T)}(\Theta, r)$.

Now, a global optimization relative to the length is presented. It is made for the information system $I$.

The set of irredundant association rules for $I$ and $r$ with the minimum length from $Rul(I, r)$ is denoted by $Rul^l(I, r)$, and the minimum length of an association rule from $Rul(I, r)$ is denoted by $Opt^l(I, r)$.

To make global optimization relative to the length, the directed acyclic graph is constructed for each decision table $I_{f_i} \in \Phi$, and local optimization relative to the length of the graph $\Delta(I_{f_i}), i = 1, \ldots, n + 1$, is made. As a result, the graph $G(I_{f_i})$ is obtained and each row $r$ of $I_{f_i}, i = 1, \ldots, n + 1$, has assigned the set $Rul_{G(I_{f_i})}(I_{f_i}, r)$ of $(f_i)$-association rules for $I$ and $r$ with the minimum length from $Rul_{\Delta(I_{f_i})}(I_{f_i}, r)$ and the number $Opt^l_{\Delta(I_{f_i})}(I_{f_i}, r)$, which is the minimum length of $(f_i)$-association rule from $Rul_{\Delta(I_{f_i})}(I_{f_i}, r)$.

Then, the value $Opt^l(I, r)$ is obtained, such that

$$Opt^l(I, r) = \min\{Opt^l_{\Delta(I_{f_i})}(I_{f_i}, r) : i = 1, \ldots, n + 1\},$$

and among all numbers, $i = 1, \ldots, n + 1$, only these are selected, where

$$Opt^l_{\Delta(I_{f_i})}(I_{f_i}, r) = Opt^l(I, r).$$

These numbers forms the set $N(I)$. Then

$$Rul^l(I, r) = \bigcup_{i \in N(I)} Rul_{G(I_{f_i})}(I_{f_i}, r).$$

As a result of the global optimization relative to the length each row $r$ of $I$ has assigned the set $Rul^l(I, r)$ of association rules with the minimum length and the number $Opt^l(I, r)$.

Below one can find the set $Rul^l(I, r)$ and the value $Opt^l(I, r)$ for the information system $I$ depicted in Fig. 1 and each row $r$ of this system.

$Rul^l(I, r_1) = \{f_2 = 1 \rightarrow f_1 = 1, f_1 = 1 \rightarrow f_2 = 1, f_1 = 1 \rightarrow f_3 = 1, f_2 = 1 \rightarrow f_3 = 1\}, Opt^l(I, r_1) = 1$;

$Rul^l(I, r_2) = \{f_2 = 0 \rightarrow f_1 = 0, f_1 = 0 \rightarrow f_2 = 0, f_1 = 0 \rightarrow f_3 = 2, f_2 = 0 \rightarrow f_3 = 2\}, Opt^l(I, r_2) = 1$.
\[ \text{Rule}(I,r_3) = \{f_2 = 1 \rightarrow f_1 = 1, f_3 = 1 \rightarrow f_1 = 1, f_1 = 1 \rightarrow f_2 = 1, f_3 = 1 \rightarrow f_2 = 1, f_1 = 1 \rightarrow f_3 = 1, f_2 = 1 \rightarrow f_3 = 1 \}, \text{Opt}(I,r_3) = 1. \]

The problem of rule length minimization is NP-hard [11, 14]. The algorithms considered in this paper have polynomial time complexity depending on the size of decision table and the number of separable subtables in it. In general case, the number of separable subtables grows exponentially with the growth of table size. However, in [9, 10] classes of decision tables are described for each of which the number of separable subtables in tables from the class is bounded from above by a polynomial on the size of decision table.

5 Experimental Results

Experiments were made using data sets from UCI Machine Learning Repository [4] and modified software system Dagger [2].

Each data set was considered as information system \( I \) and, for each attribute \( f_i \in \{f_1, \ldots, f_{n+1}\} \), the system \( I \) was transformed into a decision table \( I_{f_i} \). The column \( f_i \) was removed from \( I \) and a table with \( n \) columns labeled with attributes \( f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_{n+1} \), was obtained. Values of the attribute \( f_i \) were attached to the rows of the obtained table \( I_{f_i} \). The set \( \{I_{f_1}, \ldots, I_{f_{n+1}}\} \) of decision tables obtained from the information system \( I \) is denoted by \( \Phi \).

Table 1 presents preliminary results of experiments connected with the minimum length of irredundant association rules (column “Association rules”). For each row \( r \) of \( I \), the minimum length of an irredundant association rule for \( I \) and \( r \) was obtained. After that, for rows of \( I \) the minimum length of an association rule for \( I \) and \( r \) with the minimum length (column “Min”), the maximum length of such rule (column “Max”) and the average length of association rules with the minimum length - one for each row (column “Avg”) were obtained. Column “Rows” contains the number of rows in \( I \), column “Attr” contains the number of attributes in \( I \). This table contains also, for the purpose of comparison, minimum, average and maximum length of exact irredundant decision rules (column “Decision rules”) obtained by the dynamic programming algorithm.

Based on the results in Table 1, it is possible to see that the proposed approach allows one to obtain short association rules. The minimum value of minimum length (column “Min”) is equal to 3 only for two data sets, for the rest of data sets, this value is equal to 1. In the case of comparison of length of association and decision rules, the minimum values (columns “Min”) of minimum length of rules are the same. The average values (columns “Avg”) and the maximum values (columns “Max”), often are smaller in the case of association rules. Only for data sets “Monks-1-test” and “Monks-3-test”, the obtained results are the same for association and decision rules.

Table 2 presents the average number of nodes (column “Nodes”) and the average number of edges (column “Edges”) related to the data set \( I \) and the graph \( \Delta(I_{f_i}) \), \( i = 1, \ldots, n+1 \). For each data set \( I \), the set \( \Phi \) was obtained. For each decision table \( I_{f_i}, i = 1, \ldots, n+1 \), the graph \( \Delta(I_{f_i}) \) was constructed and the number of nodes and edges were calculated. Then, the average number of nodes and edges in the directed acyclic graphs \( \Delta(I_{f_i}), i = 1, \ldots, n+1 \), were computed.
Table 1. Minimum length of rules

<table>
<thead>
<tr>
<th>Data set</th>
<th>Rows</th>
<th>Attr</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult-stretch</td>
<td>16</td>
<td>5</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1</td>
<td>1.25</td>
<td>2</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>625</td>
<td>5</td>
<td>3.14</td>
<td>4</td>
<td>3</td>
<td>3.20</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Breast-cancer</td>
<td>266</td>
<td>10</td>
<td>1.83</td>
<td>3</td>
<td>1</td>
<td>2.67</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Cars</td>
<td>1728</td>
<td>7</td>
<td>2.02</td>
<td>6</td>
<td>1</td>
<td>2.43</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Hayes-roth-data</td>
<td>69</td>
<td>5</td>
<td>1.62</td>
<td>3</td>
<td>1</td>
<td>2.15</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Lenses</td>
<td>24</td>
<td>5</td>
<td>1.25</td>
<td>3</td>
<td>1</td>
<td>1.40</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Monks-1-test</td>
<td>432</td>
<td>7</td>
<td>2.25</td>
<td>3</td>
<td>1</td>
<td>2.25</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Monks-3-test</td>
<td>432</td>
<td>7</td>
<td>1.75</td>
<td>2</td>
<td>1</td>
<td>1.75</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Shuttle-landing</td>
<td>15</td>
<td>7</td>
<td>1</td>
<td>1.00</td>
<td>1</td>
<td>1.40</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Teeth</td>
<td>23</td>
<td>9</td>
<td>1</td>
<td>1.00</td>
<td>1</td>
<td>2.26</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>958</td>
<td>10</td>
<td>3</td>
<td>3.00</td>
<td>4</td>
<td>3</td>
<td>3.02</td>
<td>4</td>
</tr>
<tr>
<td>Zoo-data</td>
<td>59</td>
<td>17</td>
<td>1</td>
<td>1.00</td>
<td>1</td>
<td>1.56</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Average number of nodes and edges

<table>
<thead>
<tr>
<th>Data set</th>
<th>Rows</th>
<th>Attr</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult-stretch</td>
<td>16</td>
<td>5</td>
<td>48.0</td>
<td>104.0</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>625</td>
<td>5</td>
<td>742.0</td>
<td>2386.0</td>
</tr>
<tr>
<td>Breast-cancer</td>
<td>266</td>
<td>10</td>
<td>6082.0</td>
<td>61063.6</td>
</tr>
<tr>
<td>Cars</td>
<td>1728</td>
<td>7</td>
<td>4335.3</td>
<td>17697.1</td>
</tr>
<tr>
<td>Hayes-roth-data</td>
<td>69</td>
<td>5</td>
<td>190.8</td>
<td>569.0</td>
</tr>
<tr>
<td>Lenses</td>
<td>24</td>
<td>5</td>
<td>70.8</td>
<td>174.8</td>
</tr>
<tr>
<td>Monks-1-test</td>
<td>432</td>
<td>7</td>
<td>1734.9</td>
<td>6760.1</td>
</tr>
<tr>
<td>Monks-3-test</td>
<td>432</td>
<td>7</td>
<td>1584.9</td>
<td>5770.4</td>
</tr>
<tr>
<td>Shuttle-landing</td>
<td>15</td>
<td>7</td>
<td>73.6</td>
<td>368.6</td>
</tr>
<tr>
<td>Teeth</td>
<td>23</td>
<td>9</td>
<td>112.3</td>
<td>952.7</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>958</td>
<td>10</td>
<td>31415.1</td>
<td>264362.9</td>
</tr>
<tr>
<td>Zoo-data</td>
<td>59</td>
<td>17</td>
<td>3595.4</td>
<td>57868.2</td>
</tr>
</tbody>
</table>

The proposed approach of rule induction is based on the analysis of the directed acyclic graph constructed for a given decision table. The structure of the graph depends on data set, i.e., number of attributes, distribution of values of attributes, number of rows. Such graph can be huge for larger data set. Therefore, possibilities of decreasing the size of the graph were studied by the author. In [19], the graph is constructed only for selected values of attributes contained in a decision table.

6 Conclusions

In the paper, an application of dynamic programming to global optimization of exact association rules relative to length was presented. It is based on the dynamic programming approach to optimization of decision rules. However, there are differences: (i) definitions are different, (ii) the information system is used, (ii) decision table can be
inconsistent, and (iv) global optimization relative to length was studied. The presented approach can be considered as a research tool which allows one to construct association rules with minimum length.

Possible applications of association rules obtained using presented approach are construction of classifiers, inference process in knowledge base system, filling missing values of attributes.

Future works will be connected with future selection, construction of classifiers and possibilities of decreasing the size of the directed acyclic graph.

References

Author Index

Bandyopadhyay, Sibasis, 177
Bazan, Jan G., 73
Bazan-Socha, Stanisława, 73
Bozhenkova, Elena, 222
Erofeev, Evgeny, 222
Grochowalski, Piotr, 177
Jankowski, Cezary, 34
Kolbusz, Janusz, 120
Kurach, Karol, 1
Lasek, Agnieszka, 187
Lasek, Piotr, 11
Lindemann-von Trzebiatowski, Gabriela, 147
Lomazova, Irina A., 19
Luba, Tadeusz, 34
Machnicka, Zofia, 69
Mankowski, Michal, 34
Nguyen, Hung Son, 207
Nguyen, Linh Anh, 44
Niemann, Matthias, 147
Niewiadomski, Artur, 173
Paja, Wiesław, 57
Palasinski, Marek, 69
Pancerz, Krzysztof, 57, 155
Pardel, Przemysław W., 73
Pawłowski, Krzysztof, 1
Penczek, Wojciech, 173
Podmiov, Vladislav, 85
Polkowski, Lech T., 97
Popova-Zeugmann, Louchka, 19
Provatot, Oleksandr, 111
Rozycki, Paweł, 120
Rytsar, Bohdan, 130
Schmidt, Daniło, 147
Schumann, Andrew, 155
Siminski, Roman, 161
Skaruz, Jarosław, 173
Suraj, Zbigniew, 177, 187
Swieboda, Wojciech, 207
Terletska, Dmytro, 211
Virbińskaite, Irina, 222
Wakulicz-Deja, Alicja, 161
Wasilewski, Piotr, 234
Wozna-Szczęśniak, Bozena, 196
Zarychta, Jacek, 73
Zbrzeźny, Agnieszka M., 196
Zbrzeźny, Andrzej, 196
Zielosko, Beata, 237