Ferda's tutorial for creating first project

Martin Ralbovský

<martin.ralbovsky@gmail.com>

2006-03-11

Table of Contents

Abstract

This document was created for beginners in Ferda that don't know how to work with the system. It contains a tutorial to create a simple 4ft task on a trial database of a fictional bank Barbora.

Introduction

Welcome in Ferda :) If you just launched the Ferda program and don't know what to do with it, or if you need to learn some of the Ferda basic operations, then read carefully this document. We hope that in the end thing will become clearer.

The Ferda and LISp-Miner system in brief

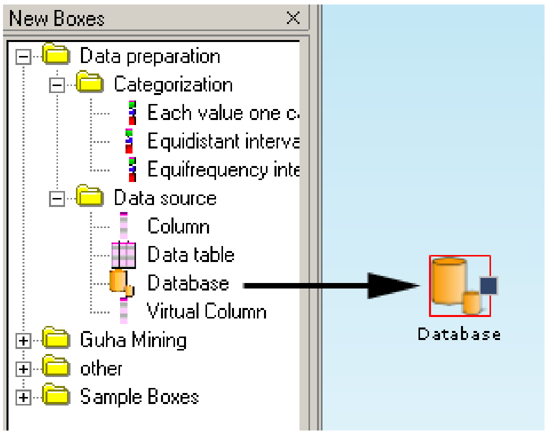
Ferda is system for mining knowledge from data. It is a successor of an older LISp-Miner system. The LISp-Miner system has been successfully under development for many years at VŠE (economics university) in Prague. Ferda as well as LISp-Miner uses association rule based data mining with improved GUHA style procedures. For more theoretical information about the procedures, see the LISp-Miner web pages.

Our example

In this tutorial we try to mine some knowledge from a fictional Barbora bank data containing information about loan profitability. We use the most known GUHA procedure implemented in Ferda, the 4FT. This procedure uses a fourfold contingency table of individual hypotheses verifies its value against defined quantifiers. You will find more about the procedure and its parameters later in this tutorial.

1. Creation of a first box

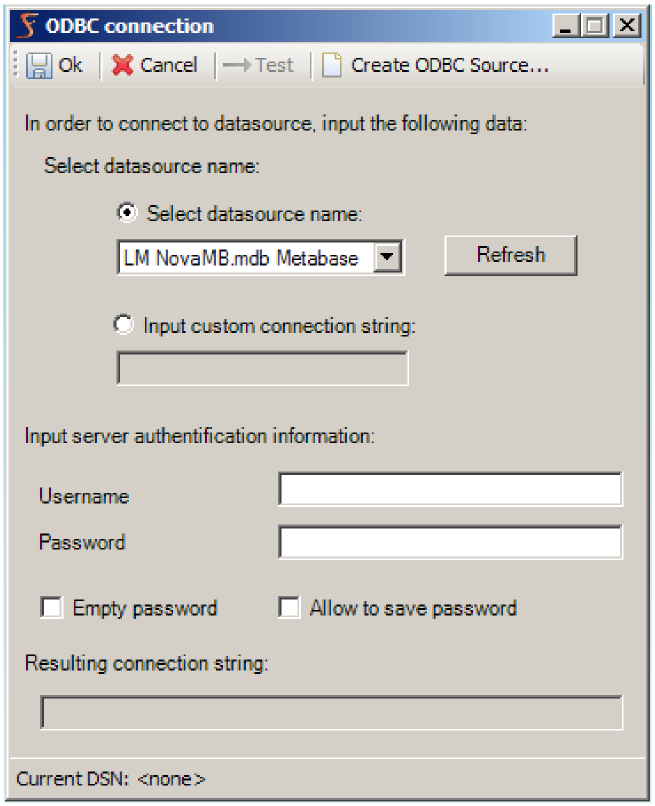
Ferda is a system that uses so called visual programming for creation of its tasks. Visual programming means connecting visual elements on the desktop and setting properties of the elements. In Ferda the visual element is called a `box`. The panel with all different boxes we can create is located on the left bottom edge of the docking environment. It contains folders and in the folders, there are boxes that we can create. The data mining process usually starts with identification of the data source. For this, we use the `Database` box in the `Datasource` folder. We create our first box by clicking on the `Database` box and dragging it onto the desktop.



2. ODBC connection setting and Setting modules

The ODBC interface provides a unified way of connecting to a data source for various data types. For our example, we use a fictional Barbora bank database with information about loans. You can download the database together with Ferda instalation program at Ferda home page.

In order for the `Database` box to work properly, we need to connect it to a data source. The `ODBC connection string` property of the box is designed for this purpose. When we click with the left mouse button on the box at the desktop, all the properties of the box appear in the `Property grid`, the component in the upper right part of the docking environment. When we click on the property `ODBC connection string`, we can write the connection string in there right away, or we can click on the button with three dots and use a `Settings module` instead. Ferda uses setting modules, when the property setting is somehow complicated. Then an ODBC connection dialog appears.



In order to connect to datasource, input the following data:

Select datasource name:

● Select datasource name:

LM NovaMB.mdb Metabase

○ Input custom connection string:

Input server authentification information:

Username

Password

□ Empty password □ Allow to save password

Resulting connection string:

Current DSN: <none>

We can see all the different data sources connected via the ODBC interface. If you hadn't established the Barbora as a data source, click on `Create ODBC source` on the toolbar. Moreover, you can also test the data sources in this dialog by clicking on the `Test` button. Please select the data source representing Barbora form the list.

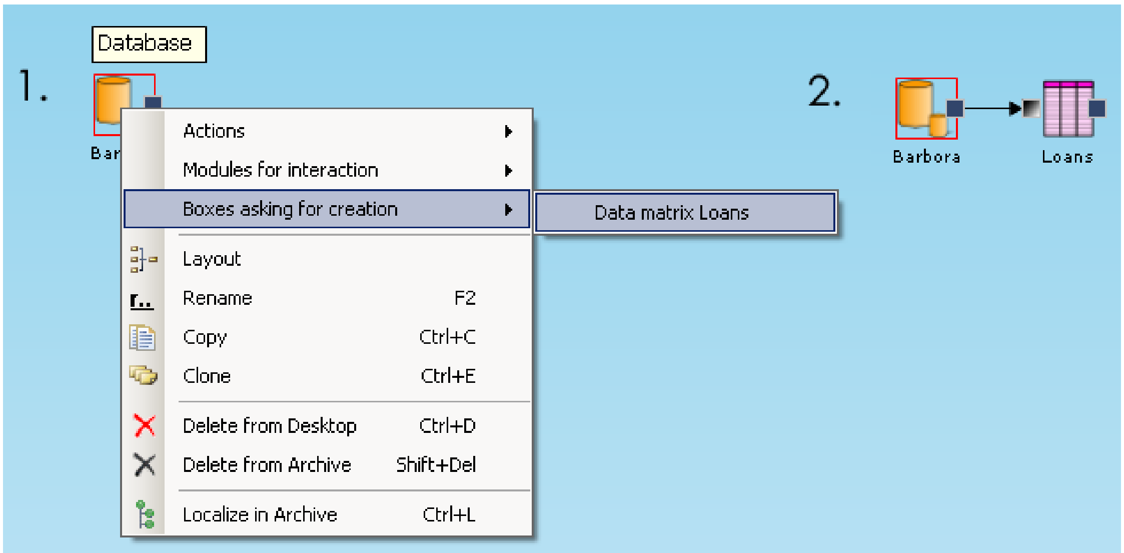
Note

Ferda supports only creation of the User and System data source in the first step of tutorial that emerges when you click on `Create ODBC source` - it does not support the File data source.

> For the LISp-Miner experts: While working in Ferda, you do not need to define a metabase (usually a Microsoft Access file - LM_Empty.mdb) for saving a project and mutual data exchange between different modules of the system. User in Ferda does not notice metabase usage, he works only with the project files (*.xfp). (Implementation: whenever the Ferda boxes execute LISp-Miner modules, they give metabase as a parameter to the modules generated before. After the LISp module finishes its work, the metabase is deleted.)

3. Data matrix creation and Boxes asking for creation

We have already a database created and correctly connected to a data source. Now we can create other boxes out of these boxes. To do this, we use a mechanism called `Boxes asking for creation`. For example the `Database` box represents a data source that can contain several tables. In this case, the Barbora database contains one table named `Loans`. Therefore, we can create a box `Data matrix` out of the `Database` box. We do it by clicking on Barbora box with the left button and from the context menu of the box we choose the `Boxes asking for creation` group and from there the `Data matrix Loans` item. A new box representing the table is going to be created.



The `Loans` box can also be created without using the Boxes asking for creation. An equivalent procedure would be dragging a new (blank) `Data matrix` box from the New boxes component, then connecting the output connector of the `Barbora` box with the input connetor of the new `Data matrix` box and setting the `Name` value to `Loans`.

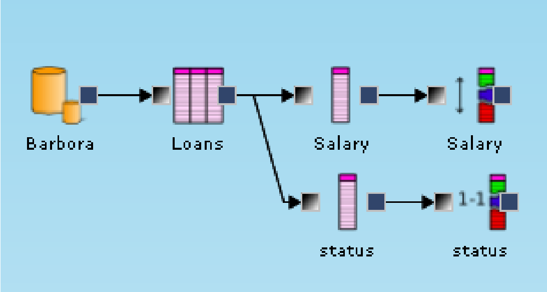
For the Ferda data mining procedures to work properly, we have to select the primary key of the data matrix we are working with. Therefore we select the `loan_id` value of the `Primary key` property.

5. Attributes creation

The boxes asking for creation principle can be applied also to other boxes then `Database`. For example the `Data matrix` table offers all the containing columns to be created. The table `Loans` contains information about the profitability of loans to clients with several characteristics. A manager of our fictional bank could be interested for instance in the relationship of income height and the profitability of the loan. We can help him to solve this task in Ferda.

At first we create the `Column` boxes for the columns `Salary` providing the income facts and `status` for the loan profitability. Then we create attributes out of the columns. Attribute is a suitable categorization of the domain values. In the `Salary` column properties, we can see that there are 76 different values in the column. Apparently it is not suitable to take all the different values into account. Instead we could create only some intervals that would cover all the possible values. The `Equidistant intervals attribute` box asking for creation from a `Column` box serves just this purpose. It takes all the values in the column and divides them into x different intervals where x can be edited in the Property grid. Well, let us create this box and fill number 1000 into the `Length` property. Now we have the 1000 czech crowns long intervals.

On the other hand, the `status` column contains only 4 different values. So it makes sense to create a different category for each value. We use the `Each value one category` box asking for creation. After the creation we have an attribute containing 4 categories.

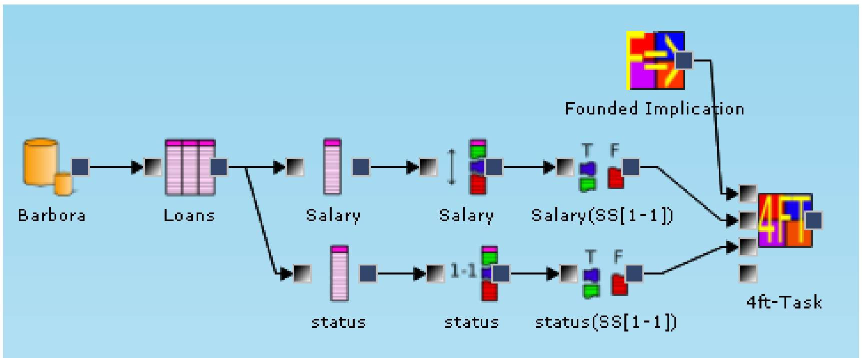


6. The 4FT task completion

Let us continue creating our task. We need to define the atoms, which are the basic building blocks for creation of *boolean attributes* that serve as an input for the 4FT procedure. We choose the `Atom setting` out of the `Boxes asking for creation` menu of both of the attributes. We can leave the default setting of the properties.

With the atoms created, let us explain the 4FT procedure a little more. The 4ft procedure is the most widely used and the most intuitive procedure out of all the GUHA procedures implemented in Ferda. There are 3 boolean attributes named `antecedent`, `succedent` and `condition` coming into the procedure as parameters. The procedure examines the relation between antecedent and succedent under a given condition. We also use a `4FT quantifier` for the task. It is a formula defined on the fourfold table and the hypotheses are verified against this formula. We use the most common `Founded implication` 4F quantifier. This quantifier contains one input parameter `p` and test if at least `p` percent of objects that satisfies the antecedent satisfies also the succedent. In other words, if when the object satisfies the antecedent, it also satisfies the succedent in at least `p` percent of the cases.

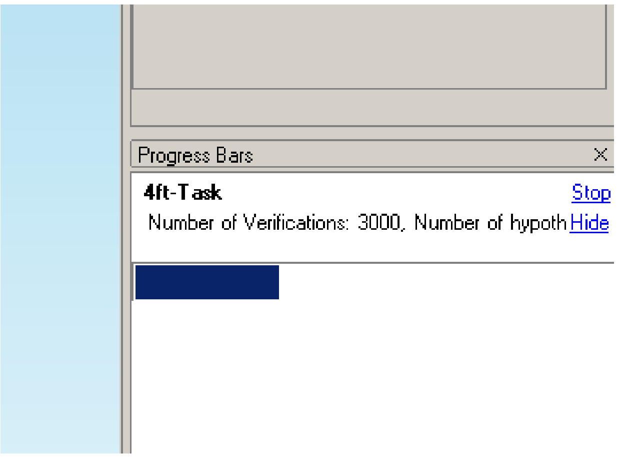
Now we have everything set for completing our task. From the `New boxes` component from the `GUHA Mining->Tasks` we drag the box `4FT-Task` behind the already constructed atoms on the desktop. This box has four input sockets: `Antecedent Succedent Condition` and the `Quantifiers`. We connect into the `Antecedent` the atom setting box created from the `Salary` column, into the `Succedent` we connect the atom setting box created from the `status` column. We won't use the condition in our task. Finally we connect a quantifier into the first socket. We connect there the `Founded implication` quantifier that can be found in the `Quantifiers` folder of the `4ft-Miner` folder. The task is ready to run.



7. Running the 4FT and Actions

Executing the action of boxes is yet another usage of the box. Actions of the box are accessible from the `Actions` item of the box context menu. Let us choose the `Run` item from the `Actions` submenu of the box.

When we run the task, a progress bar with generating information is shown. Then we select the `4ft Task` box on the desktop once again. As one can notice, the property values of the box have changed. The `State` property shows the `Completed` value. Unfortunately, none of the 20 verifications passed the quantifier conditions. We change the quantifier parameter `p` little from 0,9 to 0,8 and run the task again. Great we got a hypothesis.



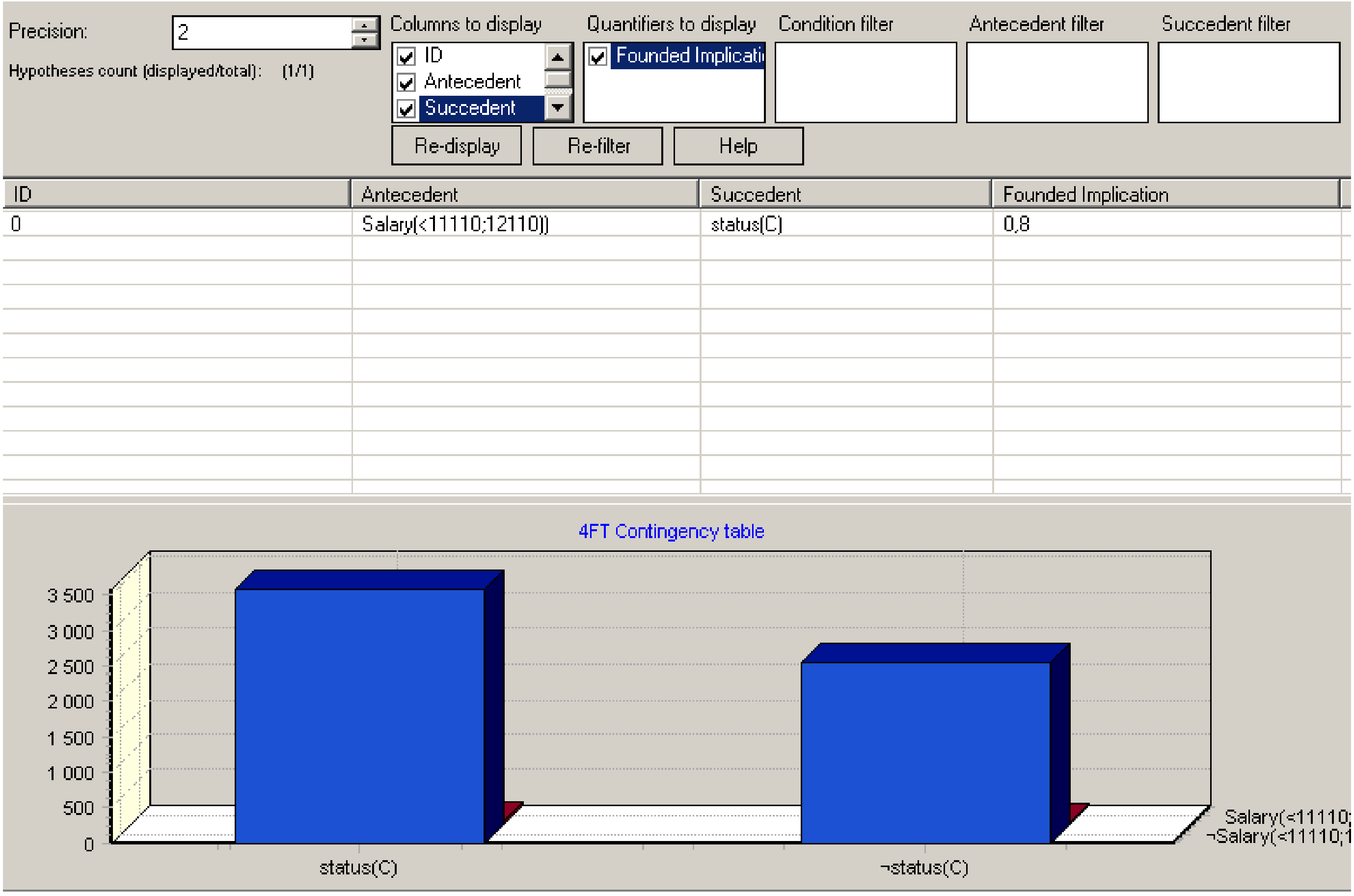
Progress Bars ×

4ft-Task

Number of Verifications: 3000, Number of hypoth

8. Displaying the results and the Modules for Interaction

With the aid of `Modules for interaction` the boxes show their content to the user. Therefore, in order to display our hypothesis, we use the `Result browser module` of the `4ft Task` box. In the boxes context menu, we choose it in the `Modules for interaction` submenu. Afterwards, a table with hypotheses is shown and underneath a place for graph. If we choose our (only) hypothesis, the graph showing the contingency table emerges in the lower part and hypothesis details are displayed in the Property grid.



Precision: 2

Hypotheses count (displayed/total): (1/1)

Columns to display: ☑ ID ☑ Antecedent ☑ Succedent

Quantifiers to display: ☑ Founded Implicati

Condition filter

Antecedent filter

Succedent filter

ID	Antecedent	Succedent	Founded Implication
0	Salary(<11110;12110))	status(C)	0,8

10. Result interpretation

Out hypothesis says that the couple antecedent (salary between 11110 and 12110) and succedent (status C) satisfies the founded implication quantifier on 80%. Which means if a client has this salary, it is a quite high probability that his final status will be C. On the first sight, this may seem as a good result. But when we look closer at the contingency table, we see that the support of the hypotheses is very small - only 45 out of (about) 6000 objects satisfy the antecedent (even less then 1 percent). When we display the the `Attribute frequency module` of the salary attribute we see unequal distribution of values. One thing to do is to use equifrequency attribute or we can modify the quantifiers... Now you know how to work with Ferda to get the best results. So good luck!