Ferda, nové vizuální prostředí pro dobývání znalostí

Michal Kováč, Tomáš Kuchař, Alexander Kuzmin, Martin Ralbovský

Univerzita Karlova v Praze, Matematicko-fyzikální fakulta
Ke Karlovu 3, 121 16 Praha 2
ferda-users@lists.sourceforge.net

Abstrakt. Cílem návrhu programu Ferda bylo vytvořit aplikaci, která nahradí stávající uživatelské prostředí systému na dobývání znalostí LISp-Miner (lispminer.vse.cz) novým, vizuálně a uživatelsky přívětivějším pracovním prostředím. Ferda přináší pohled na zadání úlohy dobývání znalostí jako na zapsání funkce pomocí vizuálních objektů – krabiček. Krabičky mají parametry a vyhodnocují své funkce nad hodnotami těchto parametrů. Program je navržen modulárně tak, aby jej bylo možné snadno rozšiřovat o další krabičky. Podařilo se vytvořit prostředí, které zpřehledňuje a zjednodušuje uživatelskou práci se systémem LISp-Miner a nabízí další možnosti výzkumu a rozšiřování.

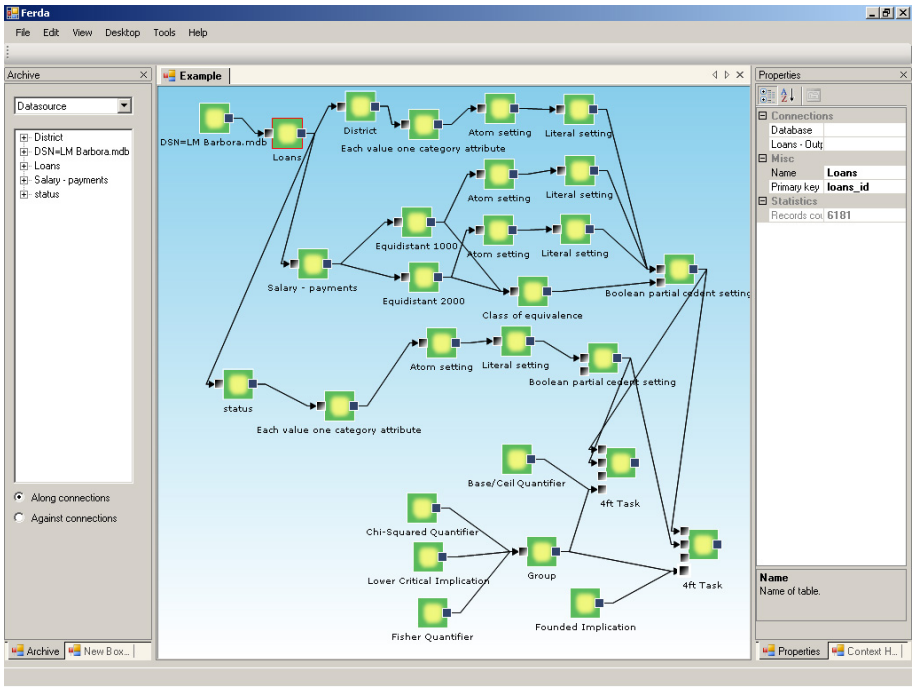
Klíčová slova: Ferda, LISp-Miner, krabička, zásuvka

1 Úvod

Historie systému LISp-Miner[1][2] sahá do roku 1996, kdy byl navržen. Jde o akademický softwarový systém pro výuku (zejména FIS VŠE a MFF UK) a výzkum v oblasti dobývání znalostí z databází. Systém obsahuje řadu modulů, které umožňují potřebným způsobem transformovat prvotní data, provádět vlastní analýzy a vhodným způsobem prezentovat výsledky. Systém je stále ve vývoji, který pokrývá jak implementaci nových modulů – v posledních dvou letech byl rozšířen o nové procedury – tak i prohlubování a rozšiřování modulů již implementovaných. Vedlejším důsledkem tohoto vývoje jsou stále větší nároky na uživatele systému pokud chce využít všech možností systému LISp-Miner. Pro snadné porozumění jistým částem článku předpokládáme, že je čtenář již se seznámen se systémem LISp-Miner.

Ferda vzniká jako softwarový projekt na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze pod vedením doc. RNDr. Jana Raucha, CSc. (Vysoká škola ekonomická v Praze). Projekt má za úkol zpřehlednit a zjednodušit postup používání existujících a pokud možno i budoucích modulů systému LISp-Miner při dobývání znalostí. Stávající postup se skládá z postupného spouštění různých aplikací, a nastavování parametrů, mezi nimiž nejsou na první pohled vidět souvislosti. V důsledku toho se proces získávání dat stává pro běžné uživatele nejasným a nepřehledným.

Ferda představuje společné vizuální prostředí pro následující procedury systému LISp-Miner: *4ft-Miner*, *SD4ft-Miner*, *KL-Miner*, *SDKL-Miner*, *CF-Miner* a *SDCF-Miner*.



Obr. 1. Prostředí Ferda

Ferda přináší vizuální model postupu dobývání znalostí podobný svou zásadou *plocha jako pracovní nástroj dobývání znalostí* (obrázek 1) systémům SPSS Clementine[8], SAS Enterprise Miner[10] a Weka[9]. Srovnání těchto systému s Ferdou naleznete v sekci 2.2.

Ferda není pouze prostředí, které pracuje nad systémem LISp-Miner, ale jedná se o samostatnou jednoduše rozšiřitelnou aplikaci, která může být využita pro různé typy úloh.

V následujících odstavcích nejdříve popíšeme základní ideu hlavních objektů prostředí Ferda (krabiček). Poté nastíníme základy zásad implementace při nichž poukážeme na rozšiřitelnost, modularitu a distribuovatelnost prostředí Ferda. Bude následovat popis prvků prostředí Ferda, na kterých bude ukázáno, jak uživateli zjednoduší práci. K závěru bude popsán příklad dobývání znalostí pomocí prostředí Ferda.

2 Krabičky

Ferda přináší pohled na zadání úlohy dobývání znalostí jako na zapsání funkce pomocí jistých vizuálních objektů – tzv. krabiček. Každá krabička zastává také roli funkce nebo několika funkcí. Krabičky mají parametry a vyhodnocují své funkce nad hodnotami těchto parametrů. Parametrům se u krabiček říká zásuvky. Do zásuvek se zapojují jiné krabičky. Krabičky jsou rozděleny podle typů a do každé zásuvky lze zapojit pouze některé typy krabiček.

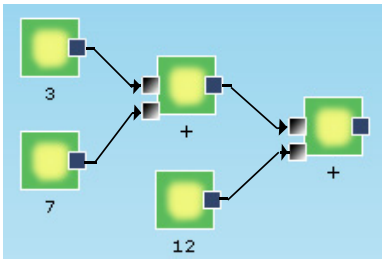
Pomocí řady jednoduchých krabiček je uživateli umožněno sestavovat vlastní algoritmy. Ferda tedy rovněž představuje něco jako uživatelský programovací jazyk. Na krabičky se lze však také dívat jako na konstrukce logiky či sémantiky, ke které existuje lambda-kategoriální gramatika.

2.1 Ukázka možností krabiček

Ukažme si krabičky na příkladu. Nebude však z oblasti dobývání znalostí, jak by se dalo očekávat. Místo toho budou krabičky předvedeny na jednoduché funkci sčítání, která je výhodná tím, že na ní lze ukázat několik možností krabiček. Bude to také názornější pro čtenáře, kteří neznají systém LISp-Miner.

Mějme krabičku *sčítání*. Funkcí této krabičky je součet dvou čísel. Pro každé z těchto čísel bude mít krabička *sčítání* zásuvku, do které bude možné zapojit krabičku typu *číslo*. Sama krabička *sčítání* je však také typu *číslo*. Nyní řekněme, že chceme spočítat $3 + 7 + 12$.

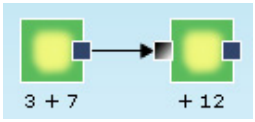
Za tím účelem definujme krabičku typu číslo, jejíž funkcí je číselná konstanta. Krabičky kromě zásuvek mohou mít vlastnosti. Vlastnosti jsou, stejně jako zásuvky, různých typů, například *řetězec*, *čísla* (ta jsou v programu rozdělena podle velikostí a toho, zda jsou s desetinou čárkou) či *datum*. Jedinou vlastností výše definované krabičky bude *hodnota číselné konstanty*. Výsledná zapojení pro výpočet $3 + 7 + 12$ je na obrázku 2.



Obr. 2. Jednoduché sčítání

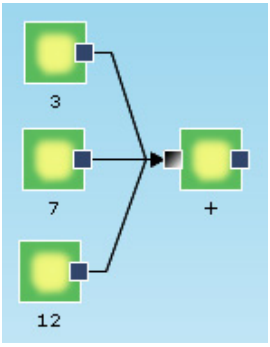
Vlastnosti a zásuvky představují vstupní parametry pro funkce, které krabička reprezentuje. Hodnoty zásuvek se nastavují vždy pomocí připojení jiné

krabičky, zatím co hodnoty vlastností se mohou nastavovat pomocí speciální komponenty v pracovním prostředím – panelu vlastností. U vlastností lze přepínat mezi nastavováním hodnoty pomocí krabiček a pomocí panelu vlastností. Vlastnostem totiž odpovídají zásuvky stejného názvu a typu, které jsou však ve výchozím nastavení uživateli skryty, ale je možné si je nechat zobrazit. Díky tomu, může mít naše krabička sčítání namísto dvou zásuvek dvě vlastnosti typu číslo a tak náš vzorec $3+7+12$ lze zapsat, jak je znázorněno na obrázku 3, místo pomocí pěti krabiček pouze pomocí krabiček dvou.



Obr. 3. Sčítání pomocí dvou krabiček

Do některých zásuvek je možné zapojit více než jednu krabičku. Proto lze vytvořit krabičku sčítání, která má jednu zásuvku a sečte všechna čísla reprezentovaná krabičkami, které jsou zapojeny do této zásuvky, viz. obrázek 4.



Obr. 4. Krabička pro multisčítání

V definici krabičky stálo, že krabička může reprezentovat více funkcí. Demonstrujme tuto schopnost krabičky již na příkladu z dobývání znalostí. Ferda obsahuje pro dobývání znalostí mimo jiné tři základní krabičky – *Databáze*, *Tabulka* a *Sloupec*. Krabička *Databáze* zastává roli několika funkcí. Jedna vrací seznam tabulek v databázi, jiná vrací způsob, jakým se má připojovat k datovému zdroji. Tento způsob nastavuje uživatel ve vlastnosti *Připojovací řetězec* této krabičky.

2.2 Porovnání s podobnými systémy

Porovnáme-li pohled na úlohu dobývání znalostí ve Ferdovi a v podobných aplikací (Clementine, SAS Enterprise Miner či Weka), ve všech těchto aplikací se pracuje s podobnými objekty jako jsou krabičky, nicméně zatímco ve Ferdovi krabička reprezentuje funkce, v ostatních aplikacích je to spíše část procesu dobývání znalostí. Proto pojem zásuvka jak ho známe z Ferdy nemá v těchto aplikacích smysl. Jejich krabička má pouze jeden vstupní bod a veliké množství parametrů se nastavuje pomocí různých dialogů. To má za následek menší opakovatelnost použití a větší složitost aplikace z hlediska uživatele. Na druhou stranu zpracování úlohy ve Ferdovi vede k většímu množství typů krabiček, kterým by měl uživatel rozumět, i k většímu množství krabiček na pracovní ploše, které může vést k obtížnější orientaci uživatele. Tomu všemu však byla věnována pozornost a vznikla patřičná vylepšení pro uživatele. Ta budou popsána dále.

3 Zásady implementace

Ferda je psán pod licencí GNU General Public License (GPL). Licence umožňuje program zdarma používat, zaručuje, že jsou k dispozici zdrojové soubory, a umožňuje rozšiřování programu někým jiným, pokud výsledný kód je také pod licencí GPL.

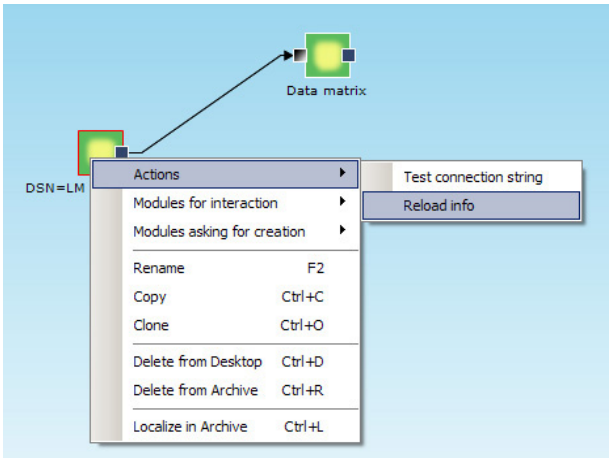
Většina současného kódu je napsána v druhé verzi jazyka C#. Výsledný program běží nad prostředím .NET Framework. Některé části také je možné spustit nad prostředím Mono[7], které běží nejen na operačním systému Windows, ale například i na systému Linux či Mac OS. Autoři by chtěli docílit toho, aby všechny nynější části prostředí Ferda bylo možné spustit nad prostředím Mono.

Program je navržen modulárně tak, aby jej bylo možné relativně snadno rozšiřovat o další krabičky. Krabička může být samostatný program napsaný v jednom z mnoha programovacích jazyků. Je možné nechat distribuovat výpočet po síti, například každá krabička může běžet na jiném stroji. To vše lze díky tomu, že je Ferda postaven nad technologií Internet Communications Engine (Ice)[6]. Jedná se o rychlý objektový otevřený middleware podobný uznávanému standardu v této oblasti – CORBA.

4 Pracovní prostředí

Při návrhu uživatelské práce v systému Ferda byla snaha držet se co nejvíce standardů v oblasti programování a vizuálního programování. Byly proto pro prostředí zvoleny standardní komponenty (části pracovního prostředí), na které je uživatel zvyklý z různých vývojových nástrojů a může je proto ihned začít používat i ve Ferdovi. Všechny komponenty, u kterých to má smysl, je možné přesouvat po pracovní ploše pomocí mechanizmu dokování oken. Dokování je další standard používaný v uživatelských prostředích mnohých produktů, za všechny jmenujme například Microsoft Visual Studio.

Vraťme se ke krabičkám. Pro uživatele je krabička grafický objekt, který se dá používat na pracovní ploše. Krabička na pracovní ploše je malý čtverec s ikonou, který má na stranách menší čtverce – zásuvky. Jednotlivé zásuvky propojuje uživatel na ploše přetáhnutím myši. Krabička má nadefinovaný seznam metod, které může provádět jak je vidět na obrázku 5.



Obr. 5. Krabičky a kontextové menu

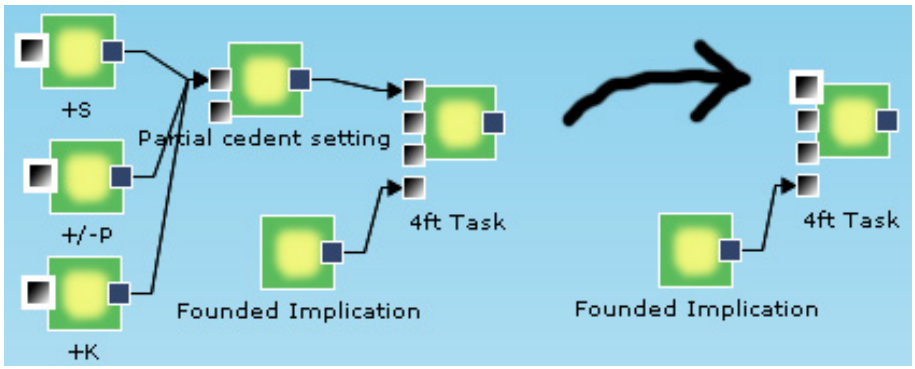
Funkčnost krabičky je někdy tak složitá, že samotné nastavování vlastností a propojování zásuvek nestačí na pohodlnou uživatelskou práci. Proto krabička může mít nadefinované moduly pro interakci. Modul pro interakci znamená volání akce, kterou zajišťuje jiný modul než samotná krabička. Často to může být část pracovního prostředí: dialogy, tabulky grafy...

Krabička může nabízet v kontextovém menu nebo v hlavním menu aplikace seznam krabiček, které se mohou automaticky vytvořit s přednastavenými hodnotami vlastnosti a zapojení. Nejlépe to lze ukázat na příkladu. Mějme krabičku *Databáze*, ta může navrhovat vytvoření krabičky *Tabulka*. V kontextovém menu této krabičky bude tedy pod položkou *Navrhované krabičky na vytvoření* seznam názvů tabulek v databázi. Jestliže uživatel klikne, na některou tabulku, vytvoří se vedle krabičky *Databáze* krabička *Tabulka*, nastaví se u ní vlastnost *jméno tabulky v databázi* na jméno, které si uživatel vybral. Dále se krabička *Databáze* propojí s touto novou krabičkou *Tabulka*.

Jestliže srovnáme výše popsané způsoby uživatelské práce s Ferdou se způsobem, jakým se formulují úlohy v původním uživatelském rozhraní systému LISp-Miner, vidíme, že je Ferda v mnohých ohledech přívětivější. Zatímco v původním rozhraní, bylo k zadání celé úlohy a k procházení jejích výsledků zapotřebí použití třech různých programů (například pro úlohu pro 4ft-Miner to jsou programy *LMDataSource*, *4ftTask* a *4ftResult*), ve Ferdovi je vše integrováno do jednoho

nástroje. Je nutné však počítat s tím, že poměrně velké množství krabiček může být pro uživatele začátečníka matoucí a i zkušený uživatel starého systému může v počátcích práce tápat, proto uživateli neustále pomáhá kontextová nápověda a osvědčuje se i koncept krabiček navrhovaných na vytvoření popsaný výše.

Komplikovanější úlohy dobývání znalostí z dat vyžadují, aby na ploše bylo velké množství krabiček zapojených do sebe. Uživatel, který pracuje s touto úlohou, často nechce znát celé komplikované zapojení všech krabiček a zajímá ho jenom výsledek. Pro tyto případy Ferda nabízí koncept zabalování a rozbalování krabiček. Při něm uživatel klikne na zásuvku, kterou chce zabalit, popřípadě na celou krabičku, jestliže chce zabalit všechny zásuvky krabičky, a zvolí patřičnou akci v kontextovém menu. Poté z pracovní plochy zmizí krabičky a na zásuvce se objeví symbol, že je v ní něco zabaleno viz. obrázek 6.



Obr. 6. Zabalování krabiček

Uživatel není v pracovním prostředí Ferda omezen jednou pracovní plochou. Může tvořit a ukládat libovolný počet pracovních ploch. Toto pomáhá rozdělit si komplikované úlohy na menší podčásti a pracovat na nich odděleně.

4.1 Archiv krabiček

Archiv je vedle pracovní plochy hlavní koncept, který má uživateli usnadnit a především zefektivnit jeho práci. Všechny krabičky, které uživatel vytvořil, budou zaznamenány v archivu. Uživatel se tímto může lehce vracet k dřívějším postupům a dále je používat. Při výchozím nastavení bude panel s archivem umístěn v levé části pracovního prostředí, jak je možné jej vidět na obrázku 1.

Uživatel může procházet archiv horizontálně nebo vertikálně. Při vertikálním prohlížení uživatel přepíná mezi typy krabiček, které má v archivu. Systém potom zobrazí pod sebou všechny krabičky v archivu, které jsou stejného typu. Horizontální prohlížení archivu znamená to, že uživatel zkoumá archiv po nebo proti směru zapojení šipek propojujících jednotlivé zásuvky krabiček.

Pro propojení archivu a pracovní plochy slouží funkce pracovní plochy nazvaná *Lokalizovat v archivu*. Při této akci kontextového menu nad krabičkou na pracovní ploše se vyhledá daná krabička v archivu a označí se. Tato funkce se opět hodí když uživatel ztrácí přehled nad množstvím krabiček, které vytvořil.

Všechny uživatelem vytvořené krabičky jsou automaticky ukládány do archivu, odkud je může uživatel kdykoliv vyzvedávat, upravovat jejich nastaveni, používat je v nových a nových kontextech. Tento koncept dramaticky zvyšuje efektivitu uživatelské práce, neboť v původním uživatelském rozhraní citelně chyběly archivy zadání cedentů, zadání literálů, tříd ekvivalence atd. Síla archivu krabiček spočívá mimo jiné i v tom, že jsou veškerá nastavení archivována jednotně a dají se smysluplně procházet.

4.2 Panel vlastností

Panel vlastností je ovládací prvek, pomocí něhož nastavujeme vlastnosti všech krabiček. Jednoduché vlastnosti krabiček se nastavují přímo v něm, pro složitější bude panel volat dialogy, které vlastnost nastaví. Dále může uživatel pomocí panelu vlastností propojit prostřednictvím speciálního dialogu dvě krabičky v archivu, aniž by je propojoval myší na pracovní ploše. Panel vlastností (PropertyGrid) je standardizovaná komponenta .NET Framework a používá se v mnoha jiných programech. Důležité zlepšení oproti původní verzi uživatelského rozhraní systému LISp-Miner je také v jednotném způsobu nastavování vlastností všech krabiček.

4.3 Kontextová nápověda

Kontextová nápověda má za úkol jednoduše seznámit uživatele s funkčností krabičky právě označené na pracovní ploše. Při označení krabičky na pracovní ploše se pro krabičku vygeneruje dynamická nápověda v komponentě Kontextová nápověda. Obsahuje základní informace o funkci krabičky, jejích vlastnostech a možnostech připojení. Nápověda může obsahovat i odkazy na stránky v externích dokumentech v různých formátech, za všechny jmenujme alespoň HTML či PDF.

5 Implementované krabičky

Ve Ferdovi jsou implementovány i krabičky, které rozšiřují možnosti systému.

5.1 Krabičky jednoduchých datových typů

Jsou vytvořeny krabičky pro nastavování jednoduchých datových typů (textový řetězec, číslo, datum, ...), které pak lze zapojit do zásuvek nastavujících tyto vlastnosti. Například krabička *Zadání booleovského dílčího cedentu* má mimo jiné vlastnosti pro nastavení minimálního a maximálního počtu literálů v cedentu. Tyto délky se standardně nastavují v panelu vlastností krabičky, lze z nich

však vytvořit zásuvky akceptující krabičky stejného datového typu. To například umožňuje analytikovi nastavit výše zmíněnou maximální délku v krabičce *Číslo*, kterou pak připojí do patřičné zásuvky. Totéž může udělat u více vlastností různých krabiček a pak pouze jednou změnou hodnoty v krabičce *Číslo* změnit nastavení mnoha krabiček najednou.

5.2 Krabička skupina

Krabička *Skupina* je dalším praktickým pomocníkem analytika při práci s Ferdou. Příklad: Analytik připraví několik krabiček různých 4ft-kvantifikátorů, které zapojí do krabičky *4ft-Task*. Nyní, bude-li chtít použít stejné kvantifikátory u jiné 4ft úlohy, musí znovu vytvořit pro všechny tyto kvantifikátory spojení s novou krabičkou *4ft-Task*. Tuto neefektivitu práce řeší Ferda krabičkou skupina. *Skupina* umožňuje seskupování krabiček jakéhokoli typu, tuto skupinu pak může analytik snadno znovu použít na jiném místě v jiném kontextu. Sama o sobě *Skupina* neposkytuje žádné funkce, při jejím zapojování do nějaké zásuvky vystupuje stejně jako krabičky do ní zapojené tj. jestliže zásuvka vyžaduje nějaké funkce, *Skupina* do něj nepůjde zapojit, pokud každá krabička uvnitř *Skupina* neposkytuje tyto funkce. Rovněž platí, že pokud lze do zásuvek zapojit jen omezený počet krabiček, nebude možné do něj zapojit *Skupina* pokud do ní není zapojen stejně omezený počet krabiček. *Skupina* je tak pro uživatele vedle archivu dalším snadno použitelným pomocníkem pro vytváření a uchovávání pracovních postupů.

5.3 Dynamické, a nedynamické atributy

V modulu *LMDataSource* systému LISp-Miner existuje několik možností pro vytvoření atributu. Uživatel může ručně vytvořit seznamy hodnot či intervalů hodnot, tvořící jednotlivé kategorie, nebo při jejich vytváření může použít tři vestavěné algoritmy pro automatické generování kategorií a tyto pak ručně upravit. Jde o tyto algoritmy:

Each value one category pro každou hodnotu z domény sloupce vytvoří právě jednu kategorii.

Equidistant intervals generuje intervaly stejné délky od zadané počáteční hodnoty prvního intervalu, po poslední hodnotu v doméně sloupce.

Equifrequency intervals generuje zadaný počet kategorií s přibližně stejným počtem objektů z tabulky v každé kategorii.

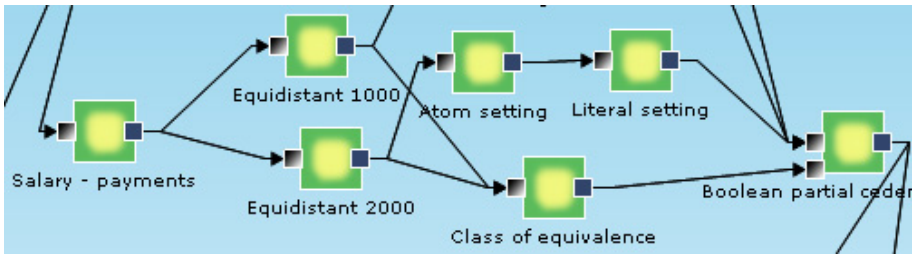
Pokud se však data pro analýzu změní, je často třeba pro stejnou analýzu znovu měnit či vytvářet nové kategorie. Systém Ferda tuto problematiku řeší tak, že je uživateli ponechána možnost ručně vytvářet kategorie a nechat si při tom pomáhat těmito algoritmy, ale navíc jsou zde krabičky tzv. dynamických atributů, které (zvolí-li si tak uživatel), ve chvíli kdy od něj jiná krabička vyžaduje množinu kategorií, ověří zda nedošlo ke změně dat a pokud ano, provedou automaticky znovu generování pro aktuální hodnoty ve zkoumaném sloupci. To, mimo jiné, umožní analytikovi snadno zkoumat rozdíly ve výsledcích nad měnící se databází v průběhu času.

6 Příklad řešení úlohy

Na obrázku 1 je naznačeno zadání dvou úloh pro *4ft-Miner*, které mají téměř stejné parametry, liší se pouze zadanými kvantifikátory. Z obrázku je patrné, že již nebylo nutné znovu vytvářet zadání cedentů pro obě úlohy zvlášť, tak jak tomu je v původním uživatelském rozhraní systému LISp-Miner, ale zde již uživatel snadno opakovaně použil dříve nadefinované – archivované – zadání cedentů.

Další práci, která by vznikala v důsledku četného vytváření spojení mezi krabičkami, si může analytik ušetřit, pomocí dříve zmíněné krabičky *Skupina*. Například obě úlohy na obrázku mají definovány čtyři kvantifikátory, z nichž tři jsou společné, ale jeden kvantifikátor má každá úloha jiný. Analytik vytvořil a nastavil tři společné kvantifikátory, ty umístil do krabičky *Skupina* a tu zapojil do příslušné zásuvky obou 4ft-úloh, poté už jen vytvořil krabičky lišících se kvantifikátorů a také obě zapojil do zásuvek příslušných krabiček.

Na tomto příkladu je rovněž naznačeno používání krabiček dynamických atributů. Takto formulovaná úloha totiž umožňuje pozdější spuštění s tím, že se, bude-li si to uživatel přát, u dynamických atributů znovu dynamicky vygenerují množiny kategorií, což v případě, že se v databázi změnily domény či frekvence u sloupců (případně odvozených), z nichž jsou odvozeny tyto atributy, ušetří analytikovi mnoho jeho práce, kterou by v dřívější implementaci grafického rozhraní systému LISp-Miner musel udělat tj. znovu ručně u každého atributu provést generování množiny kategorií.



Obr. 7. Vylepšení třídy ekvivalence

Čtenář, který již má zkušenosti se třídami ekvivalence v systému LISp-Miner, si může povšimnout, že další změny oproti původnímu rozhraní systému LISp-Miner doznala *Třída ekvivalence*, do níž šlo původně zařazovat pouze zadání literálů. Ve Ferdovi je možné navíc vytvářet už třídy ekvivalence atributů. To znamená, že všechna zadání literálů, která budou odvozena z těchto atributů jsou ve stejné třídě ekvivalence. V tomto příkladě jsou z krabičky *Odvozený sloupec* pojmenované *Salary – payments* odvozeny dva dynamické atributy typu *Equidistantní intervaly* pojmenované *Equidistant 1000* a *Equidistant 2000* (číslo

v názvu označuje délku intervalů). Zřejmě nemá smysl, aby v cedentech vystupovali zároveň literály odvozené z obou těchto atributům, proto jsou oba atributy zařazeny do jedné třídy ekvivalence.

7 Závěr

Ačkoli prostředí Ferda je teprve v beta verzi, naše první praktické zkušenosti s ním přivedly uspokojivé výsledky. Kombinace vizuálního zobrazení, které převažuje v moderních systémech na dobývání znalostí, s konceptem krabičky dokázala sjednotit dříve roztříštěný postup v systému LISp-Miner. Možnosti pracovního prostředí popsané v tomto článku na čele s archivem krabiček, pracovní plochou nebo zabalováním a rozbalováním krabiček pravděpodobně zjednoduší a zefektivní práci analytika při dobývání znalostí. Osvědčily se i nové nápady v čele s krabičkou *Skupina* a dynamickými atributy popsané ve shodně pojmenovaných částech tohoto článku, které rozšiřují záběr stávajícího systému.

Za největší výhodu prostředí však jeho tvůrci považují jeho modularitu a rozšiřitelnost. Například není příliš těžké vytvořit krabičku, která bude sekvenčně měnit nastavení mezí dynamických atributů a bude si tak sama simulovat data měnící se v čase. Otevírá se také prostor pro vytváření jednoduchých krabiček heuristicky simulujících práci analytika. Například jednoduchá krabička dynamicky upravující nastavení kvantifikátorů úlohy tak, aby byly vlastnosti i počet výsledných hypotéz uspokojující pro uživatele.

V budoucnu by Ferda měl být rozšířen o nové procedury dobývání znalostí. Za zmínku stojí procedura RelMiner[5], která počítá asociační pravidla nad hvězdicovitým relačním schématem. Ferda je také vhodným kandidátem na prostředí pro projekt EverMiner[3].

Reference

1. J. Rauch, M. Šimůnek. Systém LISp-Miner *ZNALOSTI 2003, 2. ročník konference* Ostrava 2003. ISBN 80-248-0229-5.
2. Seznam publikací týkajících se systému LISp-Miner. http://lispminer.vse.cz/references.html.
3. J. Rauch, M. Šimůnek. *GUHA Method and Granular Computing.* Beijing 25.07.2005 - 27.07.2005. In: HU, Xiaohua, LIU, Qing, SKOWRON, Andrzej, LIN, Tsau Young, YAGER, Ronald R., ZANG, Bo (ed.). IEEE 2005. Piscataway: IEEE, 2005, s. 630-635.
4. P. Aubrecht, M. Kejkula, P. Kremen, L. Novakova, J. Rauch, M. Simunek, O. Stepankova. *Mining in Hepatitis Data by LISp-Miner and SumatraTT.* Accepted for publicvation in proceedings of Discovery Challenge 2005 see http://lisp.vse.cz/challenge/ecmlpkdd2005/chall2005.htm.
5. Martin Ducháček. Diplomová práce 2005 *Nástroj pro správu databází s využitím pro multi-relační DataMining.*
6. ZeroC. http://www.zeroc.com, Internet Communications Engine (Ice).
7. Mono Project. http://www.mono-project.com.

8. Clementine. http://www.spss.com/es/noticias/npclementine.htm. Clementine Data Mining System.
9. Weka. http://www.cs.waikato.ac.nz/~ml/weka. Weka - Java Programs for Machine Learning, University of Waikato.
10. SAS Enterprise Miner. http://www.sas.com/technologies/analytics/datamining/miner/

Annotation:

Ferda, new visual environment for data mining

The goal of project Ferda was to create an application which will replace the current user environment for the data mining system LISp-Miner (lispminer.vse.cz) by a new, more visually and user friendly working environment. Ferda brings in the conception of data mining task as the record of the function using visual objects – boxes. Boxes have their parameters and evaluate their functions based on the values of the parameters. The program is modular and is designed to make the addition of new boxes easy. We have created the environment that organizes and simplifies user interaction with the LISp-Miner system and offers further possibilities for research and extensions.