Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Konvertor datových formátů ve FCA a veřejných repozitářích

2016

Jan Nováček

Vedoucí práce: Mgr. Jan Outrata, Ph.D.

Studijní obor: Informatika pro vzdělávání, prezenční forma

Bibliografické údaje

Autor:	Jan Nováček
Název práce:	Konvertor datových formátů ve FCA a veřejných repozitářích
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2016
Studijní obor:	Informatika pro vzdělávání, prezenční forma
Vedoucí práce:	Mgr. Jan Outrata, Ph.D.
Počet stran:	31
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jan Nováček
Title:	Convertor of data formats used in FCA and public repositories
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2016
Study field:	Computer Science for Education, full-time form
Supervisor:	Mgr. Jan Outrata, Ph.D.
Page count:	31
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Program Swift – Relational Data Converter umožňuje převod dat mezi datovými formáty, které jsou využívané ve veřejných repozitářích (CSV, ARFF, C4.5) a formáty pro formální konceptuální analýzu (Burmeister, FIMI/DAT a DTL). V úvodní části práce jsou vysvětleny základní pojmy jako tabulková data, datové typy a škálování, které jsou nezbytné pro správné pochopení funkcí programu. V dalším textu jsou podrobně popsány podporované datové formáty. Závěrečná kapitola je věnována návrhu programu, jeho implementaci a vlastnostem. Výsledný program poskytuje grafické i konzolové uživatelské rozhraní, je multiplatformní a volně dostupný. Součástí textu, jako příloha, je manuál programu v angličtině.

Synopsis

Swift – Relational Data Converter is a program that converts data between the file formats used in public repositories (CSV, ARFF, C4.5) and formats used in Formal Conceptual Analysis (Burmeister, FIMI/DAT, DTL). In the first part, the main concepts which are necessary for understanding of the program are being introduced (table data, data types and scaling). This is followed by the chapter describing data formats supported by the program. The final chapter is focused on the design of the program, its implementation and properties. The program provides a Graphical user interface and a Command-line user interface. Swift is a multi-platform and an open source. An English manual of the program is part of the text, as an appendix.

Klíčová slova: formální konceptuální analýza, FKA, škálování, datový formát, CSV, ARFF, Burmeister, C4.5, FIMI/DAT, DTL, veřejný repozitář

Keywords: Formal Conceptual Analysis, FCA, scaling, data format, CSV, ARFF, Burmeister, C4.5, FIMI/DAT, DTL, public repository

Děkuji Mgr. Janu Outratovi, Ph.D. za cenné rady, připomínky a ochotu. Děkuji Anetě Čapkové za psychickou podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Cílem práce bylo vytvořit multiplatformní aplikaci pro převod dat mezi datovými formáty využívaných ve formální konceptuální analýze (FKA) a veřejných repozitářích. Formální konceptuální analýza je moderní metoda analýzy dat, jejímž vstupem jsou tabulková data v určitém textovém formátu, který záleží na konkrétní implementaci FKA algoritmů. Data, která bychom chtěli pomocí této metody zpracovávat (například z odvětví medicíny, psychologie, ekologie a dalších oborů), jsou ale často v odlišných formátech a je tedy potřeba mít nástroj, který je schopen data mezi formáty převádět.

Veřejné repozitáře jsou volně přístupné online databáze určené pro vědecké účely. Jeden z nejznámějších repozitářů je Machine Learning Repository [1] založený v roce 1987 studentem Kalifornské Univerzity v Irvine [2]. Dalším, pro tuto práci relevantním repozitářem je Frequent Itemset Mining Dataset Repository [3], který byl vytvořen v rámci FIMI workshopů na vědeckých konferencích IEEE ICDM 2003 a 2004.

Datových formátů je velké množství, proto se budeme zabývat pouze těmi nejpoužívanějšími, mezi které patří CSV, ARFF, C4.5, Burmeister, FIMI/DAT a DTL, a ty později detailně rozebereme. Nyní uvedeme pouze základní rozdělení a to na formáty podporující pouze logické bivalentní atributy (ano/ne) a vícehodnotové atributy. Formální konceptuální analýza pracuje pouze s bivalentními daty a proto je potřeba mít k dispozici metodu, kterou lze vícehodnotová data převádět na bivalentní. Tato metoda se nazývá konceptuální škálovaní (dále jen škálování) a hraje při převodech významnou roli. Speciálním případem škálovaní je úplná binarizace dat. Obě metody převodu vícehodnotových dat na bivalentní jsou podrobně popsány v následujících kapitolách.

Samotná práce se zabývá pouze přípravou dat pro formální konceptuální analýzu, nikoli samotnou formální konceptuální analýzou a není tedy dále v textu rozebírána. Zájemce o tuto problematiku odkazuji na [4].

2 Data

Vstupem pro převod jsou v našem případě vždy data v textové podobě. Nebude pro nás podstatné z jakého odvětví byla data získána, jestli se jedná o meteorologické údaje nebo o databázi registrovaných vozidel. Zajímáme se zejména o to, jakým způsobem jsou data uspořádána, jakých hodnot mohou nabývat a v jakém jsou datovém formátu. Na první pohled by se mohlo zdát, že pojem uspořádání dat a pojem datový formát mají stejný význam, ale není tomu tak. Pojem uspořádání dat vyjadřuje, jak obecně chápeme data, bez ohledu na jejich zápis. Datový formát vyjadřuje soubor pravidel, podle kterých byla data zapsána, tedy jejich syntaxi. My se budeme se zabývat celkem šesti datovými formáty, ve kterých jsou data uspořádána do tabulky. Část zabývající se datovými formáty je pro tuto práci klíčová a proto je pro ni vyhrazena samostatná kapitola.

2.1 Uspořádání dat

Při převodech vždy pracujeme s tabulkovými daty[1], která se skládají z řádků a sloupců. Řádky chápeme jako objekty a sloupce jako jejich atributy neboli vlastnosti. Jako příklad si můžeme uvést studenty vysoké školy s atributy: jméno, příjmení, věk, datum narození, pohlaví, obor, studijní průměr a zda studuje. Příklad je znázorněn v tabulce 1 na straně 8. Nakonec je třeba dodat, že na pořadí objektů a atributů z hlediska významu tabulkových dat nezáleží.

Tabulka 1: Příklad tabulkových dat – studenti vysoké školy

jméno	příjmení	věk	datum nar.	pohlaví	obor	průměr	studuje
Tomáš	Vavruša	21	1990-02-11	muž	MI	1.55	ano
Otakar	Panáček	19	1992-08-22	muž	M-IV	2.99	ne
Lenka	Pokorná	22	1989-12-01	žena	MI	1.74	ano
Jan	Veselý	21	1990-12-04	muž	BINF	2.46	ano
Eliška	Dostálová	20	1991-06-25	žena	MI	1.12	ano

2.2 Datové typy

Jednotlivým atributům je možné přiřadit datový typ, který přesně určuje, jakých hodnot může atribut nabývat. Tyto hodnoty se nazývají *obor hodnot* datového typu. Datových typů existuje velké množství a často se od sebe liší jen mírně. Pro naše účely si představíme pouze několik základních, které pro nás budou důležité. Protože datový typ je v podstatě množina, budeme s ním jako s množinou v následujícím popisu zacházet.

- **Číselný datový typ** (numeric) jsou všechna reálná čísla. Nerozlišujeme číselné atributy na celočíselné a s desetinou čárkou, jako je tomu například ve většině programovacích jazyků.
- **Řetězec** (string) je množina všech slov libovolné délky nad Unicode abecedou. Patří do něj prázdný řetězec (slovo délky 0) i jednotlivé znaky (slova délky 1).
- **Datum** (date) je množina všech dat v jednotném formátu.
- **Výčet** (enumeration/nominal) je určen konečnou množinou hodnot, kterých atribut může nabývat.
- **Logická hodnota** (bool) je speciálním případem výčtového typu, který je definován symboly pro logickou jedničku a nulu (např. ano/ne). Logická jednička vyjadřuje skutečnost, že objekt má určitou vlastnost a naopak logická nula, že objekt vlastnost nemá.

[1] Také nazývanými relační data.

- **Složený datový typ** může být tvořen prvky různých datových typů včetně sebe sama. Můžeme ho tedy označit jako rekurzivní datový typ.

Na základě předchozího popisu můžeme jednotlivé datové typy (kromě složeného datového typu, který si ukážeme na příkladu později) přiřadit k atributům z tabulky 1, což je znázorněno v tabulce 2. V dalším textu budeme využívat rozdělení atributů na dvě skupiny podle jejich typu. První skupinou jsou bivalentní atributy, které jsou typu logická hodnota. Druhou skupinou jsou vícehodnotové atributy, které mohou být jakéhokoli datového typu kromě typu logická hodnota. Na závěr je důležité uvést speciální hodnotu NULL, která se v datech může vyskytovat u jakéhokoli atributu a má význam nespecifikované hodnoty.

Tabulka 2: Přiřazení datových typů k atributům z tabulky 1

	numeric	string	date	enum	bool
jméno	·	×	·	·	·
příjmení	·	×	·	·	·
věk	×	·	·	·	·
datum nar.	·	·	×	·	·
pohlaví	·	·	·	×	·
obor	·	·	·	×	·
studuje	·	·	·	·	×

2.3 Škálování dat

Pod pojmem škálování dat myslíme způsob, kterým je možné převést vícehodnotové atributy na bivalentní atributy. Formálně lze zavést pojem škála, což je zobrazení s z oboru hodnot daného datového typu $dom(T)$ do oboru hodnot typu logická hodnota $s : dom(T) \to dom(bool)$. Škálování dat je tedy aplikace škály na atribut a výsledkem škálování je logická hodnota. V programovacích jazycích se takové funkce často označují jako predikáty.

Pojem škála je definován ve formální konceptuální analýze mnohem sofistikovaněji (zájemce opět odkazuji na [4]), my si ale vystačíme s tímto vysvětlením.

Jako příklad uvedeme aplikaci škály s_1 a s_2 na atributy věk a pohlaví z tabulky 1 na straně 8.

- s_1 pro x: je $x > 20$?
- s_2 pro x: je x „žena“?

Výsledek škálování je uveden v tabulce 3 na straně 10.

Tabulka 3: Výsledek aplikace škály s_1 a s_2 na atributy věk a pohlaví z tabulky 1

s_1(věk)	s_2(pohlaví)
true	false
false	false
true	true
true	false
false	true

2.4 Úplná binarizace dat

Úplná binarizace dat je operace, kterou je možné aplikovat na jednotlivé atributy objektů v datech. Pro každou hodnotu vybraného atributu, která se vyskytuje u některého z objektů, je vytvořen nový binární atribut. Nové atributy jsou vytvořeny v pořadí v jakém se vyskytovaly v datech a pro každou hodnotu je vytvořen pouze jeden nový atribut. Tedy pro vybraný atribut je počet nových binárních atributů roven počtu hodnot, které se v datech u vybraného atributu vyskytují a navzájem se liší. Nový binární atribut je pro určitý objekt pravdivý, pokud hodnota vybraného (původního) atributu toho objektu je rovna hodnotě, podle které byl nový atribut vytvořen. Nyní ukážeme, v jakém pořadí budou atributy výsledných dat po aplikaci binarizace na dva zvolené atributy z původních dat.

Pokud pracujeme s daty o atributech $a_0, ..., a_{n-1}$ a budeme aplikovat úplnou binarizaci na atributy a_k a a_s, kde $0 \leq k \leq (n-1)$, $0 \leq s \leq (n-1)$ a $k < s$, pak výsledná data budou obsahovat atributy

$$a_0, \ldots, a_k, \ldots, a_{k+j-1}, \ldots, a_{s+j-1}, \ldots, a_{s+(j-1)+(l-1)}, \ldots, a_{(j-1)+(l-1)+(n-1)},$$

kde j je počet nových binárních atributů vytvořených dle atributu a_k a l je počet nových binárních atributů vytvořených dle atributu a_s. Situace pro libovolný počet binarizovaných atributů by vypadala obdobně.

Dodejme, že pokud bychom úplnou binarizaci dat neměli k dispozici, stejného výsledku bychom mohli dosáhnout pouze využitím škálování. Pro každou poprvé vyskytující se hodnotu v vybraného atributu, bychom vytvořili nový binární atribut pomocí škálování, kde škála s by byla ve tvaru: s pro x: platí $x = v$?. Tento způsob by však byl poměrně nepraktický, pokud bychom dopředu nevěděli jakých hodnot daný atribut nabývá a vstupní data by obsahovala velké množství objektů[2]. Proto je zavedení této operace výhodné, i když bychom se bez ní obešli.

Jako příklad uvedeme aplikaci úplné binarizace na atributy věk a obor z tabulky 1 na straně 8. Výsledná data jsou zobrazena v tabulce 4 na straně 11. Pro přehlednost jsou v tabulce 4 vynechány hodnoty atributů, které nebyly binarizovány.

[2]Například pro data obsahující pouze 1000 objektů, bychom se bez pomocného programu, který nám by nám vrátil množinu hodnot vyskytujících se v atributu nejspíše neobešli.

Tabulka 4: Výsledek aplikace úplné binarizace na atributy věk a obor z tabulky 1

jm.	přij.	21	19	22	20	d. r.	poh.	MI	M-IV	BINF	prum.	stud.
		1	0	0	0			1	0	0		
		0	1	0	0			0	1	0		
		0	0	1	0			1	0	0		
		1	0	0	0			0	0	1		
		0	0	0	1			1	0	0		

3 Datové formáty

Datový formát je určen souborem pravidel, podle kterých jsou data zapsána v datovém souboru[3]. Datových formátů v současné době existuje obrovské množství a proto z nich vybereme pouze ty, které jsou nejpoužívanější a také ty, které mají praktické využití ve formální konceptuální analýze. Celkem se budeme zabývat šesti datovými formáty: CSV, ARFF, C4.5, Burmeister, FIMI/DAT a DTL.

Při popisu jednotlivých formátů se budeme snažit být co nejpřesnější a proto pokud bude daný formát složitější, uvedeme jeho gramatiku. Neformálně řečeno je gramatika soubor pravidel, podle kterých je možné vygenerovat jakýkoli datový soubor v daném formátu. Pojem gramatika lze zavést mnohem přesněji, zájemce odkazuji na [5].

Příklad gramatiky datového typu numeric 1 je uveden na straně 11. Pomocí této gramatiky je možné vygenerovat například čísla: -457, 11.78, 1.602e-31 ($1.602 \cdot 10^{-31}$), 34e11 ($34 \cdot 10^{11}$) apod. Gramatika 1 se nám bude později hodit při popisu gramatik jednotlivých formátů, kdy ji budeme používat jako proměnnou[4] `<numeric>`.

```
1  <numeric> ::= <int> ("." <number>?)? ("e" <int>)?
2  <int> ::= ("+" | "-")? <number>
3  <number> ::= \d+
```

Zdrojový kód 1: BNF gramatika datového typu numeric

Datové formáty, kterými se budeme zabývat lze rozdělit na dvě části.

1. Hlavička (Header)
2. Data

Hlavička se vyskytuje na začátku datového souboru a obsahuje informace o jednotlivých atributech (datový typ, jméno apod.). U některých formátů může

[3] Chápeme ho jako fyzický soubor uložený v souborovém systému i jako soubor dat označovaný v literatuře jako dataset.

[4] V terminologii gramatik se používá označení neterminál.

být volitelná a u některých se nevyskytuje vůbec. Data jsou umístěna pod hlavičkou a tvoří zbytek datového souboru.

3.1 Comma-separated values (.csv)

CSV [6, 7] formát je jeden z nejpoužívanějších vícehodnotových datových formátů a to hlavně kvůli jeho jednoduchosti. Využívá se například jako výstupní formát z databází (jako jedna z možných alternativ) a nebo jako vstup pro software pracující s tabulkovými daty jako je například Microsoft Excel nebo LibreOffice Calc.

Hlavní nevýhodou CSV je, že pro něj neexistuje standard, což je způsobeno jeho jednoduchostí a tedy tím, že je velmi snadné jej vhodně upravit pro konkrétní využití. V praxi je možné se setkat s velkým množstvím modifikací jako například s využitím odlišných oddělovačů. Z toho důvodu jsme se snažili zavést CSV formát maximálně variabilní, aby se nám podařilo postihnout většinu často používaných forem CSV.

3.1.1 Specifikace

Každý řádek datového souboru je složen z hodnot oddělených oddělovačem. Výchozí oddělovač je čárka, ale je možné použít libovolný znak. Hodnoty mohou být libovolného datového typu, který ale není nikde v souboru explicitně uveden. Pokud hodnota obsahuje oddělovač, je nutné oddělovač escapovat (zapsat před něj zpětné lomítko) např. `foo\,bar`.

Hlavička je umístěna na prvním řádku a skládá se ze jmen jednotlivých atributů. Pořadí jmen určuje příslušnost k jednotlivým atributům. Hlavičku je možné vynechat, v tom případě datový soubor začíná daty hned od prvního řádku.

S využitím dat z tabulky 1 na straně 8 ukážeme příklad použití datového souboru v CSV formátu 2. Popis formátu je možné nalézt například na [7].

```
1 jméno, příjmení, věk, datum nar., pohlaví, obor, průměr, studuje
2 Tomáš, Vavruša, 21, 1990-02-11, muž, MI, 1.55, ano
3 Otakar, Panáček, 19, 1992-08-22, muž, M-IV, 2.99, ne
4 Lenka, Pokorná, 22, 1989-12-01, žena, MI, 1.74, ano
5 Jan, Veselý, 21, 1990-12-04, muž, BINF, 2.46, ano
6 Eliška, Dostálová, 20, 1991-06-25, žena, MI, 1.12, ano
```

Zdrojový kód 2: Data z tabulky 1 v CSV formátu

3.2 Attribute-Relation File Format (.arff)

ARFF [8, 9] byl vytvořen v rámci projektu zaměřeného na strojové učení (Machine Learning Project) na katedře Informatiky University of Waikato pro použití

v jimi vytvořeném software WEKA (Waikato Environment for Knowledge Analysis) [10]. WEKA je soubor algoritmů strojového učení pro získávání informací z dat. Jedná se o open source projekt spadající pod GNU licenci napsaný v jazyce Java. Více informací o projektu lze získat na oficiálních stránkách [11].

Protože ARFF má složitější syntaxi, na straně 13 uvádíme gramatiku 3 pro jeho jednoznačný popis.

```
1  <file> ::= <relation-part> <attributes-part> <data-part>
2  <relation-part> ::= <comment-line>* <relation> "\n" <comment-line>*
3  <attributes-part> ::= <attribute-line> | ("\n" <attribute-line>)*
4  <data-part> ::= "@data" "\n" (<data-line> | ("\n" <data-line>)*)
5  <data-line> ::= <comment> | <instance> | <blank>
6  <instance> ::= <string> | ("," <string>)*
7  <attribute-line> ::= <comment> | <attribute> | <blank>
8  <attribute> ::= "@attribute" <string> <type>
9  <type> ::= "numeric" | "string" | <nominal> | <date> | <relational>
10 <nominal> ::= "{" <string> | ("," <string>)* "}"
11 <date> ::= "date" <string>?
12 <relational> ::= "relational" "\n" <attribute-part> "@end" <string>
13 <relation> ::= "@relation" <string>?
14 <string> ::= [^%,{}]+ | "'" .+ "'"
15 <blank> ::= \s*
16 <comment> ::= "%".*
17 <comment-line> ::= <comment> "\n"
```

Zdrojový kód 3: BNF gramatika ARFF

3.2.1 Specifikace

Hlavička musí být uvedena a skládá se ze dvou částí. První část tvoří jméno relace a druhou definice atributů.

Jméno relace se zapisuje ve tvaru: `@relation <name>`, kde `<name>` je libovolný řetězec, který nemusí být uveden a nesmí obsahovat symboly pro složené závorky ({}), čárku (,) a procento (%).

Definice atributů je složena z řádků ve tvaru `@attribute <name> <type>`, kde `<name>` je název atributu pro který platí stejná omezení jako pro název relace a `<type>` je datový typ příslušného atributu. ARFF podporuje datové typy:

- Numeric (číselný datový typ), např. `@attribute age numeric`.
- String (řetězec), např. `@attribute name string`.
- Date (datum) je složen z klíčového slova `date` a volitelné části specifikující formát dat. Výchozí formát je ISO-8601 kombinující datum a čas: `YYYY-MM-DDThh:mm:ss`, např. `2016-01-19T16:07:37`. Definice atributu typu datum může vypadat například následovně:
  `@attribute date-of-birth date 2016-01-19`.

- Nominal (výčet) přímo definuje množinu možných hodnot daného atributu ve tvaru `{value1, value2, ...}`.
  Např. `@attribute color {red, green, blue}`.
- Relational (složený datový typ) se zapisuje dle tvaru 4 na straně 14, kde `<further-attributes>` jsou definice libovolného množství atributů libovolného typu.

```
1 @attribute <name> relational
2   <further-attributes>
3 @end <name>
```

Zdrojový kód 4: Datový typ relational ARFF

V předchozím výčtu jsou uvedeny názvy datových typů dle ARFF a v závorce za nimi příslušný datový typ podle definic které jsme uvedli v kapitole 2.2.

Část s daty je umístěna pod hlavičkou a začíná deklarací `@data`, pod kterou jsou uvedeny jednotlivé instance. Každá instance je zapsána na jednom řádku a pořadí jejich atributů určuje příslušnost k definicím atributů uvedených v hlavičce. Jednotlivé hodnoty atributů na řádku jsou odděleny čárkou. Hodnoty datového typu String a Nominal který obsahuje znak pro procento nebo čárku, musí být uzavřeny do uvozovek a jsou case sensitive[5]. Hodnoty datového typu Date musí být ve formátu, který je uveden v hlavičce. U datového typu Relational musí být hodnoty uzavřeny do dvojitých uvozovek. Pro nespecifikovanou hodnotu atributu se používá otazník (pro NULL hodnoty).

Dodejme, že všechna klíčová slova ARFF: `@relation`, `@attribute`, `@data`, `@end`, `string`, `numeric`, `date`, `relational` jsou case insensitive[6]. Každý řádek začínající symbolem procento je komentář a může být uveden na kterémkoli místě v datovém souboru. Soubor také může obsahovat libovolný počet prázdných řádků, které se přeskakují. Nakonec uvedeme příklad 5 na straně 15 datového souboru zapsaného v ARFF. Přesný popis formátu je možné nalézt na [8].

3.3 C4.5 formát (.names, .data, .test)

C4.5 [13, 14] je program implementující C4.5 algoritmus, který slouží k vytváření rozhodovacích stromů (více informací o algoritmu a související tématice najdete např. v [12]). Vstupem pro tento program jsou tři soubory které tvoří C4.5 formát:

- `file.names`
- `file.data`
- `file.test`

[5] Záleží na použitých velkých a malých písmenech.
[6] Nezáleží na použitých velkých a malých písmenech.

```
% 1. Title: Students Database
%
% 2. Sources:
%     (a) Creator: Jan Novacek
%     (b) Donor: John Marshall
%     (c) Date: July, 2015
%
@relation students

@attribute name string
@attribute surname string
@attribute age numeric
@attribute date-of-birth date YYYY-MM-DD
@attribute sex {man, woman}
@attribute school-info relational
  @attribute course {MI, BINF, M-IV}
  @attribute average numeric
  @attribute is-studying {True, False}
@end school-info

@data
Tomáš, Vavruša, 21, 1990-02-11, muž, "MI, 1.55, ano"
Otakar, Panáček, 19, 1992-08-22, muž, "M-IV, ?, ne"
Lenka, Pokorná, 22, 1989-12-01, žena, "MI, 1.74, ano"
Jan, Veselý, 21, 1990-12-04, muž, "BINF, 2.46, ano"
Eliška, Dostálová, 20, 1991-06-25, žena, "MI, 1.12, ano"
```

Zdrojový kód 5: Data z tabulky 1 v ARFF

V souboru s příponou `.names` je pouze definice hlavičky. Gramatika hlavičky 6 je uvedena na straně 16. Soubor `.data` obsahuje jednotlivé instance s atributy a `.test` testovací sadu dat (bývá občas vynechán) pro C4.5 program.

3.3.1 Specifikace

Hlavička je složena z jednotlivých záznamů, které definují jména atributů, hodnoty atributů a třídy. Záznamy jsou složeny z identifikátorů, ve kterých se znaky pro čárku, otazník nebo dvojtečku nesmí vyskytovat a nebo musí být escapovány. Pokud se na řádku vyskytuje znak `"|"`, zbytek řádku je ignorován a tedy svislá čára je znak pro komentáře. Identifikátory mohou obsahovat tečku i bílá místa s omezením, že za tečkou nesmí následovat bílé místo. Každý záznam je ukončen tečkou, která může být vynechána v případě, že se jedná o poslední znak na daném řádku.

První řádek v souboru definuje jména tříd, které jsou odděleny čárkou a ukončeny tečkou např. `class1, class2, class3.`. Záznamy na dalších řádcích definují atributy v pořadí ve kterém jsou uvedeny v souborech `.data` a `.test` a jsou zapsány ve tvaru: `attribute-name : attribute-type.`. Řetězec `attribute-name` je jméno atributu pro který platí omezení popsaná výše a `attribute-type`

```
<names> ::= <entry> | (<delimiter> <entry>)*
<entry> ::= <classes> | <attribute> | <blank>
<classes> ::= <string> | <string> "," <classes> | <comment>
<attribute> ::= <string> ":" <type> <comment>? | <comment>
<blank> ::= \s*
<delimiter> ::= "." | "\n"
<string> ::= [^|?,.\s]+
<type> ::= "continuous" | "ignore"| <discrete> | <enum>
<discrete> ::= "discrete" \d+
<enum> ::= <string> | ("," <string>)*
<comment> ::= "|".*
```

Zdrojový kód 6: BNF gramatika souboru definující hlavičku C4.5 formátu

definuje datový typ daného atributu. C4.5 formát podporuje následující datové typy:

- Continuous (numeric) např. `age : 24.`.
- Discrete <n> - definuje typ který může nabývat maximálně n různých hodnot, např. `grade : discrete 6.`.
- Seznam hodnot (výčet) má stejný význam jako typ discrete <n>, s tím rozdílem, že přímo definuje možné hodnoty, které jsou odděleny čárkami. Tento způsob zápisu je preferovaný pro diskrétní atributy.
  Např. `grade : A, B, C, D, E, F.`.
- Ignore typ vyjadřuje, že daný atribut bude ignorován (nebude využit), např. `grade : ignore.`.

Příklad souboru `.names` 7 je uveden na straně 16.

```
day, night.
temperature : continuous.
humidity : ignore.
sky : sunny, cloudy, mostly sunny, clear.
wind-speed : continuous.
fog : yes, no.
weatherstation-active: discrete 5.
personal-count : continuous.
```

Zdrojový kód 7: Příklad souboru C4.5 formátu definující hlavičku: weather.names

Soubory s příponami `.data` a `.test` obsahují na každém řádku jednu instanci s atributy oddělenými čárkami. Třídy jsou zapsány jako poslední hodnota na řádku a chybějící hodnoty (NULL) jsou označeny otazníkem. Příklad 8 souboru `.data` je uveden na straně 17. Přesný popis formátu je možné nalézt na [14].

```
21.81, sunny, 48, no, station4, 12, day.
12.55, cloudy, 102, yes, station1, 4, night.
25.97, mostly sunny, 64, yes, station3, 11, day.
18.45, clear, 54, no, station4, 1, night.
17.46, sunny, 99, no, station3, 5, day.
```

Zdrojový kód 8: Příklad souboru C4.5 formátu obsahující data: weather.data

3.4 Burmeister (.cxt)

Burmeister [15] je jeden z nejpopulárnějších datových formátů pro FKA, který využívají například aplikace: ConImp, ConExp a conexp-clj. Jedná se o formát podporující pouze bivalentní atributy s poměrně jednoduchou syntaxí.

3.4.1 Specifikace

Hlavička datového souboru v Burmeister formátu začíná písmenem „B“, které podle autorů nemá žádný speciální význam, ale je zvykem psát jej na začátek souboru [16]. Na dalším řádku je zapsáno jméno datového souboru, které může být vynecháno. Poté následuje řádek s počtem objektů a řádek s počtem atributů. Na dalších řádcích jsou pod sebou zapsána jména objektů a pod nimi jména atributů.

Datová část se nachází pod hlavičkou a je složena z řádků, které reprezentují jednotlivé objekty s atributy v pořadí, které je dané definicí atributů v hlavičce datového souboru. Každý řádek je složen ze symbolů `"X"` pro logickou jedničku a symbolů `"."` pro logickou nulu. Nakonec dodejme, že v datovém souboru se na libovolném místě může vyskytovat libovolný počet prázdných řádků, které jsou ignorovány.

Jako příklad využijeme data z tabulky 1 ze strany 8, které pomocí škálování převedeme do Burmeister formátu. Škála pro jednotlivé atributy bude vypadat následovně:

- **jméno** - s_{name} pro x: obsahuje x znak „k“?
- **příjmení** - $s_{surname}$ pro x: obsahuje x podřetězec „korn“?
- **věk** - s_{age} pro x: je $x > 20$?
- **datum narození** - s_{date} pro x: je $x >$ 1991-01-01?
- **pohlaví** - s_{sex} pro x: je x „muž“?
- **obor** - s_{course} pro x: je x „MI“?
- **průměr** - $s_{average}$ pro x: je $x < 1.5$?
- **studuje** - s_{study} pro x: je x „ano“?

Jako jména objektů jsme využili atribut jméno. Výsledek škálování a následné převedení do Burmeister formátu je uvedeno v příkladu 9 na straně 18. Přesný popis formátu je možné nalézt na [15].

```
1  B
2  Students
3  5
4  8
5  Tomas
6  Otakar
7  Lenka
8  Jan
9  Eliska
10 name
11 surname
12 age
13 date-of-birth
14 sex
15 course
16 average
17 is-studying
18 ..X.XX.X
19 X..XX...
20 XXX..X.X
21 ..X.X..X
22 X..X.XXX
```

Zdrojový kód 9: Příklad datového souboru students.cxt v Burmeister formátu

3.5 FIMI/DAT (.dat)

Frequent Itemset Mining Implementations (FIMI/DAT) [17] je další často používaný datový formát pro FKA programy jako například FCALGS [18]. Stejně jako Burmeister formát, podporuje pouze bivalentní atributy a ve srovnání s výše uvedenými formáty poskytuje způsob pro minimální možný zápis bivalentních dat. To je způsobeno jednak tím, že FIMI/DAT nemá hlavičku a tedy v datovém souboru jsou zapsána pouze samotná data, ale hlavně tím, jakým způsobem jsou data na řádku zapsána, což popíšeme v následujícím textu.

3.5.1 Specifikace

V datovém souboru ve FIMI/DAT formátu je zapsána na každém řádku jedna instance. Jednotlivé řádky se skládají z indexů atributů, které pro danou instanci nabývají hodnoty logické jedničky. Indexy jsou od sebe odděleny libovolným množstvím bílých míst. Atributy které nabývají hodnoty logické nuly nejsou v souboru vůbec uvedeny, což je hlavní důvod pro úsporu prostoru, která byla zmíněna v úvodu této kapitoly.

Příklad 10 datového souboru ve FIMI/DAT formátu je uveden na straně 19, ve kterém jsme využili data z příkladu 9 na straně 18. Přesný popis formátu je možné nalézt na [17].

```
2 4 5 7
0 3 4
0 1 2 5 7
2 4 7
0 3 5 6 7
```

Zdrojový kód 10: Příklad datového souboru students.dat ve FIMI/DAT formátu

3.6 DTL (.dtl)

Datový formát DTL je velmi podobný FIMI/DAT, ale navíc umožňuje specifikovat třídy pro jednotlivé objekty.

3.6.1 Specifikace

Datový soubor v DTL formátu je složen z řádků ve tvaru `data | třídy`, kde data jsou indexy pravdivých bivalentních atributů (tato část je stejná jako u FIMI/DAT formátu) a `"|"` je oddělovač pro hodnoty atributů a tříd. Část třídy obsahuje libovolný počet tříd (ale u všech objektů stejný), což mohou být libovolné řetězce oddělené stejným oddělovačem jako část data (výchozí oddělovač je mezera).

Příklad 11 datového souboru v DTL formátu je uveden na straně 19, ve kterém se vyskytují třídy: `class1={a, b} class2={aa, bb}`.

```
0|a aa
0 1|a bb
0 1 2|a aa
0 1 2 3|b aa
0 1 2 3 4|a bb
```

Zdrojový kód 11: Příklad datového souboru students.dtl v DTL formátu

4 Konvertor datových formátů

V předchozích kapitolách jsme vysvětlili, s jakými daty budeme pracovat, jak jsou data organizována a v jakých formátech je můžeme zapsat. Toto vysvětlení bylo nezbytné pro správné pochopení funkcí konvertoru datových formátů, pojmenovaného Swift – Relational Data Converter, jehož implementace je hlavním předmětem této práce.

V následujícím textu popíšeme, na základě jakých požadavků byla aplikace vystavěna, jakých jsme při implementaci využili prostředků, jak je aplikace navržena a jaké jsou její hlavní funkce. Nakonec stručně popíšeme podobné existující programy.

4.1 Základní požadavky

Hlavní požadavek na aplikaci přímo vyplývá z jejího názvu, jedná se tedy o konverzi datových formátů CSV, ARFF, C4.5, Burmeister, FIMI/DAT a DTL, které jsme specifikovali v kapitole 3. Aplikace musí být schopna převést každý formát na libovolný jiný formát (z vyjmenovaných formátů), a protože formátů, které naše aplikace podporuje je 6, dostáváme celkem 36 možných převodů. Z tohoto požadavku přirozeně vyplývají další dva požadavky. Prvním je poskytnout uživateli možnost specifikovat škály pro jednotlivé atributy při převodech z vícehodnotového formátu do dvouhodnotového formátu např. z ARFF do FIMI/DAT formátu. Druhým je umožnit uživateli dodatečně specifikovat informace při převodu z formátu obsahujícího menší množství informací do formátu s větším množstvím informací. Například při převodu z CSV do Burmeister formátu je kromě škály nutné specifikovat ještě názvy jednotlivých objektů.

Aplikace by měla být multiplatformní, což znamená, že by ji mělo být možné spustit minimálně na třech nejrozšířenějších operačních systémech, kterými jsou Windows, Mac OS X a Linux/UNIX. Tento požadavek je nutné zohlednit hlavně pří výběru programovacího jazyka, ve kterém je aplikace implementována.

Dále by uživateli mělo být umožněno pracovat s aplikací jednak prostřednictvím příkazového řádku a jednak přes grafické rozhraní, přičemž by tyto dvě části měly být nezávislé. Tím je myšleno, že by uživatel neměl být nucen instalovat grafické knihovny v případě, že hodlá použít pouze příkazový řádek k ovládání aplikace. Stejně jako v předchozím případě je nutné tento požadavek brát v úvahu při výběru programovacího jazyka.

4.2 Použité nástroje

Jako základní nástroj pro implementaci aplikace byl vybrán programovací jazyk Python. Jedná se multiparadigmový[7] a multiplatformní interpretovaný programovací jazyk, který podporuje mimo nejběžnějších operačních systémů jako jsou Windows, Mac OS X a Linux/UNIX například také Solaris nebo BeOS[8]. Python je volně použitelný open source projekt, ve kterém je napsán například populární webový framework Django nebo knihovny PyGame pro tvorbu her. Více informací o jazyku Python je možné získat například na [20].

Pro tvorbu grafického uživatelského rozhraní (GUI) byl vybrán nástroj PyQt, což je Python nadstavba nad Qt frameworkem. Stejně jako Python je PyQt

[7] Podporuje procedurální, funkcionální i objektový přístup.

[8] Všechny operační systémy, které Python podporuje lze najít na [19].

multiplatformní, ale volně použitelný pouze pro nekomerční účely. Více informací o PyQt je možné získat na [21].

Jako poslední zmíníme nástroj pyparsing, což je modul poskytující alternativní přístup pro vytváření a spouštění jednoduchých gramatik oproti tradičnímu lex/yacc přístupu nebo oproti využití regulárních výrazů [22]. Naše aplikace využívá modul pyparsing v několika případech, jedním z nich je parsování hlavičky datového souboru v ARFF formátu, ve kterém využíváme gramatiku pro parsování datového typu numeric 1. Tuto gramatiku 12 na straně 21 definovanou pomocí modulu pyparsing uvádíme jako příklad, abychom ukázali, že je možné téměř přímo přepsat gramatiku v BNF formě, do gramatiky definovanou pomocí modulu pyparsing.

```
1 point = Literal('.')
2 e = CaselessLiteral('E')
3 plusorminus = Literal('+') | Literal('-')
4 number = Word(nums)
5 integer = Combine(Optional(plusorminus) + number)
6 numeric = Combine(integer +
7                   Optional(point + Optional(number)) +
8                   Optional(e + integer))
```

Zdrojový kód 12: Gramatika datového typu numeric 1 definovaná pomocí modulu pyparsing

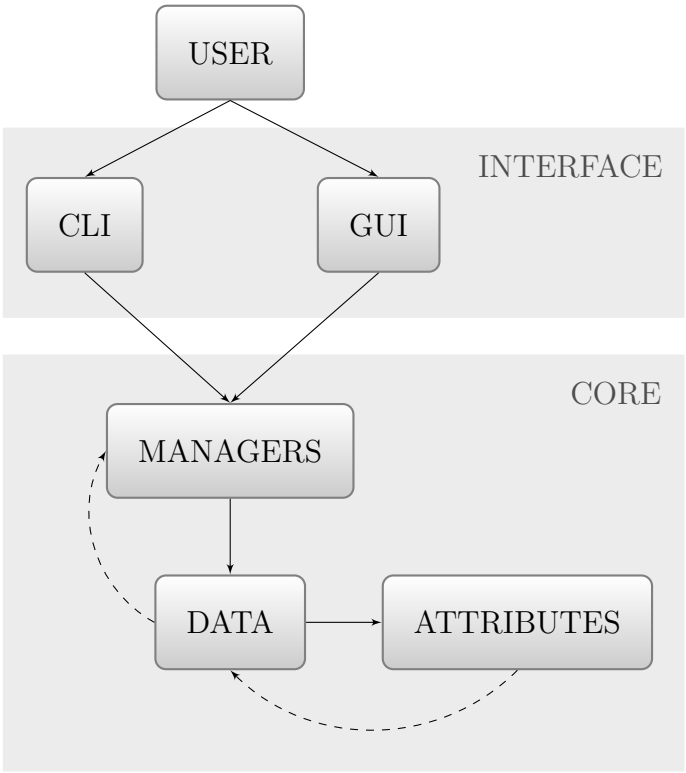
4.3 Struktura

Strukturu aplikace můžeme rozdělit na dvě hlavní části. První část tvoří rozhraní pomocí kterého uživatel komunikuje s aplikací, druhou část tvoří jádro ve kterém je implementována nejpodstatnější část funkcionality[9]. V obrázku 1 na straně 22 je znázorněno schéma struktury aplikace, ve kterém každý uzel (z části INTERFACE nebo CORE) tvoří logický celek struktury, který má svou specifickou funkci. Vztahy mezi uzly jsou znázorněny pomocí plných a přerušovaných šipek. Samotné plné šipky vyjadřují předání řízení programu jinému celku a plná v kombinaci s přerušovanou šipkou vyjadřuje vytvoření celku (ten, ke kterému vede plná šipka) a jeho využívání celkem (ten, ke kterému vede přerušovaná šipka). V další části uvedeme základní popis činnosti aplikace v interakci s uživatelem.

4.3.1 Základní popis činnosti

Uživatel předá aplikaci argumenty, které určují, jak bude samotný převod vypadat, přičemž může využít buď příkazového řádku (uzel CLI) nebo grafického rozhraní (uzel GUI). Rozhraní zpracuje získané argumenty a vytvoří příslušný

[9]Dodatečné funkce jako například vyhledávání v tabulce dat jsou implementovány přímo v části GUI.



Obrázek 1: Schéma struktury aplikace

manager (uzel MANAGERS) podle akce, kterou chce uživatel provést. Manager dále řídí činnost aplikace a je zodpovědný za vytvoření objektu reprezentujícího datový soubor (uzel DATA), jeho zpracování a vrácení korektního výstupu uživateli. Objekt reprezentující datový soubor je zodpovědný za korektní čtení/zápis hlavičky a datové části, získání potřebných informací z dat a vytvoření kolekce atributů podle argumentů od uživatele. Hlavním úkolem atributů (uzel ATTRIBUTES) je řídit škálovaní hodnot při převodech a uchovávat informace o výskytu hodnot v datovém souboru.

Nakonec dodejme, že výše zmíněný popis je velmi zjednodušený a slouží pouze pro základní představu o struktuře a činnosti aplikace. Ve skutečnosti aplikace poskytuje podstatně více funkcí než pouze převody, tyto funkce budou uvedeny později.

4.3.2 Algoritmus zpracování dat

Nejdůležitější částí celé aplikace je průchod řádků tabulkových dat a zpracování vybraných hodnot. Z toho důvodu uvádíme příklad 13 na straně 23 na kterém ukážeme ideu základního algoritmu zpracování dat.

Příklad 13 je napsán v jazyce Python a je zcela funkční. Na řádku 1 až 5 je definice třídy `Attr`, která reprezentuje atribut (sloupec) tabulkových dat. Atribut obsahuje vlastnost `key`, což může být buď index atributu nebo jeho jméno a

```
1  class Attr:
2     def __init__(self, key):
3        self.key = key
4     def scale(self, value):
5        return value > 18
6
7  data = [[23, 22, 12], [13, 15, 6], [2, 57, 90]]
8  attributes = [Attr("id"),Attr("2"),Attr("age"),Attr("id")]
9  template = {"1": 10, "2": 2, "id": 0, "age": 1}
10
11 def process_data(data, attributes, template):
12   for line in data:
13      scaled_values = []
14      for attr in attributes:
15         # index hodnoty v řádku, která přísluší akurálnímu atributu
16         index = template[attr.key]
17         # hodnota získaná z řádku dle indexu
18         value = line[index]
19         # atribut řídí škálovaní hodnoty
20         scaled = attr.scale(value)
21         scaled_values.append(scaled)
22      # vrácení/zapsání zpracovaných dat
23      print(scaled_values)
24
25 process_data(data, attributes, template)
26
27 # výstup:
28 # [True, False, True, True, True]
29 # [False, False, False, False, False]
30 # [False, True, True, False, True]
```

Zdrojový kód 13: Příklad průchodu řádků tabulkových dat a zpracování jednotlivých hodnot

metodu `scale()`, která aplikuje škálu na aktuální hodnotu a vrátí pravdivostní hodnotu. V našem příkladě je pro jednoduchost zadána škála staticky, ale v programu se samozřejmě vytváří dle argumentů zadaných uživatelem.

Na řádku 7 je seznam obsahující seznamy, které reprezentují řádky dat. Na řádku 8 je seznam atributů definovaný uživatelem. Všimněte si, že atributy v tomto seznamu mohou být v libovolném pořadí, mohou se opakovat nebo mohou být vynechány. Na řádku 9 je slovník, který se vytváří dle argumentů od uživatele a dle hlavičky vstupního datového souboru. Klíče ve slovníku odpovídají klíčům jednotlivých atributů a hodnoty odpovídají indexům, na kterých se nacházejí hodnoty příslušící k danému atributu.

Řádky 12 až 24 obsahují hlavní algoritmus, dle kterého probíhá zpracování dat. Algoritmus postupně prochází všechny řádky dat (`for line in data`) a pro každý řádek dále projde všechny atributy definované uživatelem (`for attr in attributes`). Dle klíče atributu je získán index, na kterém se nachází

hodnota příslušící aktuálnímu atributu (`index = template[attr.key]`). Index je dále využit k získání hodnoty z aktuálního řádku (`value = line[index]`), která je předána atributu. Atribut na hodnotu následně aplikuje škálu (`scaled = attr.scale(value)`) a výsledek této aplikace je uložen do seznamu výsledných škálovaných hodnot (`scaled_values.append(scaled)`). Škálovaný řádek je nakonec zapsán na výstup `print(scaled_values)`).

4.3.3 Časová složitost

Časovou složitost [23] algoritmu `process_data` z příkladu 13 na straně 23 určíme v průměrném a nejhorším případě. Časové složitosti v O notaci pro všechny operace využité v algoritmu jsou uvedeny v tabulce 5 na straně 24. Tabulka byla vytvořena podle [24] a za předpokladu, že škálování probíhá v konstantním čase.

Pro časovou složitost v průměrném případě $T_a(n)$ může psát

$$T_a(n) = O(n) \cdot (4O(n) + O(n))$$

z čehož dostáváme

$$T_a(n) = O(n^2).$$

Pro časovou složitosti v nejhorším případě $T_w(n)$ může psát

$$T_w(n) = O(n) \cdot ((3O(n) \cdot O(n)) + O(n))$$

z čehož dostáváme

$$T_w(n) = O(n^3).$$

Odvození $T_a(n)$ a $T_w(n)$ se liší pouze v časové složitosti získání hodnoty ze slovníku, která je v průměrném případě $O(1)$ a v nejhorším případě $O(n)$.

Tabulka 5: Časové složitosti operací v O notaci, vyskytujících se v algoritmu 13

	průměrný případ	nejhorší případ
iterace přes seznam	$O(n)$	$O(n)$
získání hodnoty - seznam	$O(1)$	$O(1)$
získání hodnoty - slovník	$O(1)$	$O(n)$
škálování	$O(1)$	$O(1)$
přidání hodnoty - seznam	$O(1)$	$O(1)$
zapsání hodnot na výstup	$O(n)$	$O(n)$

4.4 Hlavní funkce

Program kromě převodů dat mezi datovými formáty poskytuje i několik dalších funkcí. Všechny hlavní funkce zde uvedeme a stručně popíšeme. Popis bude sloužit pouze pro představu o možnostech programu. Úplný popis je k dispozici v manuálu, který je součástí přílohy.

Konverze datových formátů
Konverze dat mezi šesti datovými formáty: CSV, ARFF, C4.5, Burmeister, FIMI/DAT a DTL, což je celkem 36 možných převodů.

Filtrace atributů
Změna pořadí atributů, vynechání atributů a opakování atributů při kterémkoli převodu.

Filtrace objektů
Vynechání objektů z jakéhokoli převodu.

Škálování atributů
Aplikace škály na vybrané atributy při kterémkoli převodu. Škálovat je možné atributy všech typů, uvedených v kapitole 2.2.

Škálování tříd
Aplikace škály na vybrané třídy při převodech, ve kterých jsou vstupní data ve formátů podporující třídy.

Úplná binarizace atributů
Úplná binarizace vybraných atributů při kterémkoli převodu.

Analýza dat
Získání statistických informací o vstupních datech.

Náhledy dat
Zobrazení libovolné části vstupních dat ve formě tabulky.

Nakonec dodejme, že všechny výše uvedené funkce je možné aplikovat na libovolně velký datový soubor[10].

4.5 Podobné programy

Volně dostupných programů, které se zaměřují na převod dat mezi datovými formáty používaných ve FKA a veřejných repozitářích není mnoho. V současné době existují dva, které jsou našemu programu Swift podobné. Jedná se o programy FcaBedrock a FcaStone.

4.5.1 FcaBedrock

Program FcaBedrock [25] je volně dostupný na [26] a je s ním možné pracovat přes grafické uživatelské rozhraní na platformě Windows.

Program umožňuje převody z formátů CSV a 3-column CSV do formátů Burmeister a FIMI/DAT (celkem 4 převody). Dále podporuje například funkce škálování číselných atributů, vynechání atributů, ukládání metadat vstupního souboru a jejich následné využívání a další. Jedná se o program, který se se svými funkcemi nejvíce blíží našemu.

[10] Velikost vstupního souboru je omezena pouze místem na disku, ne operační pamětí.

4.5.2 FcaStone

Druhým podobným programem je FcaStone [27], který je volně dostupný na [28]. Jedná se o konzolovou aplikaci, kterou je možné nainstalovat na operační systém Linux, Mac OS X nebo Windows.

Zaměřuje se na převod mezi formáty využívaných ve FKA nástrojích jako je například ToscanaJ, ConExp, Galicia a Colibri, převod mezi FKA formáty a grafickými formáty [28]. FcaStone je od našeho programu poměrně odlišný a to jak z hlediska funkcionality, tak i podporovaných formátů.

Závěr

Výsledkem práce je program Swift – Relational Data Converter pro převod dat mezi datovými formáty CSV, ARFF, C4.5, Burmeister, FIMI/DAT a DTL. Program podporuje převod všech kombinací těchto formátů, tedy celkem 36 převodů. Při všech převodech je možné filtrovat atributy, měnit jejich pořadí nebo je opakovat. Dále je k dispozici škálování a úplná binarizace atributů, vynechání objektů z převodu, analýza vstupních dat a náhledy dat ve formě tabulky. Program poskytuje grafické a konzolové uživatelské rozhraní a je možné jej spustit na systémech Windows, Mac OS X a Linux/UNIX. Všechny jeho možnosti jsou popsány v anglických manuálech, které jsou součástí přílohy této práce a také dostupné na webových stránkách projektu http://gnovis.github.io/swift/. Všechny požadavky definované zadáním práce se tedy podařilo naplnit.

V úvodní části práce jsou vysvětleny základní pojmy (tabulková data, škálování, datové typy a úplná binarizace), které jsou důležité pro pochopení činnosti a funkcí programu. Při vysvětlování je odhlédnuto od formálních definic těchto pojmů, protože pro práci s programem nejsou podstatné. V další části jsou podrobně rozebrány podporované datové formáty. Závěrečná kapitola je věnována samotnému programu, ve které je stručně popsána jeho struktura a činnost, nástroje využité pro implementaci, jeho vlastnosti a funkce.

Ve srovnání s podobným programem FcaBedrock, Swift přináší nový přístup k převodům a to hlavně v obecnosti řešení. Například FcaBedrock poskytuje pouze 4 možné převody, zatímco Swift 36 převodů, přičemž obtížnost přidání nového datového formátu nezávisí na počtu již podporovaných formátů. Dalším příkladem je škálovaní, u kterého FcaBedrock zavádí pojem diskrétního a progresivního škálování, zatímco Swift poskytuje pouze jeden obecný typ, který oba zmíněné typy snadno nahradí. Navíc FcaBedrock umožňuje škálovat pouze číselné atributy, ale Swift umožňuje škálovat atributy všech datových typů. Druhý vzdáleně podobný program FcaStone se zaměřuje převážně na převody jiného typu (hlavně do grafických formátů) a jeho funkcionalita se s naším programem prolíná jen málo (pouze převod Burmeister formátu na CSV a naopak).

Výsledný program je volně dostupný na https://github.com/gnovis/swift, kde se prostřednictvím systému Git může na jeho vývoji kdokoli podílet. Jedním z možných rozšíření je vytvoření webového uživatelského rozhraní, které by využívalo funkcí programu Swift a umožňovalo jeho využití bez nutnosti instalace.

Conclusions

The result of this thesis is a program Swift – Relational Data Converter, which converts data between data formats CSV, ARFF, C4.5, Burmeister, FIMI/DAT and DTL. The program can convert any combination of these formats, which means 36 possible conversions. In any conversion, it is possible to filter attributes, change their order or repeat them. The program also provides functions such as scaling, total binarization, the omission of selected objects, analyzing input data and data preview. The program provides Command-line interface and Graphical user interface. It is possible to use it on Windows, Mac OS X and Linux/UNIX platforms. All the functions of the program described in the English manuals are part of the thesis appendix and also available on the project website http://gnovis.github.io/swift/.

First of all, the main concepts that are necessary for the understanding of the program such as table data, scaling, data types and total binarization are being introduced. The formal definitions of these concepts aren't included, because they are of no importance for the use of the program. This is followed by the detail description of the data formats supported by the program. The final chapter is then focused on a design of the program, tools used for its implementation, its functions and properties.

In comparision with a program of similar character FcaBedrock, Swift brings more universal approach to the conversions. For instance, FcaBedrock provides only four conversions, whereas Swift 36 conversions. Moreover, in Swift, the difficulty of adding a new data format does not depend on a number of already supported data formats. Another example can be scaling. Where FcaBedrock defines two types of scaling – discrete and progressive – Swift provides only one universal type of scaling that can satisfy the function. Furthermore, FcaBedrock can scale only numeric attributes, whereas Swift can scale attributes of any supported type. Another slightly similar program is FcaStone, focused on different type of conversions, especially on graphical formats conversions. Its use is quite far from Swift, the only same conversions are CSV to Burmeister and vice versa.

The final product is freely available at https://github.com/gnovis/swift and anyone can contribute to it by using Git system.

There is a possibility for a future extension in form of Web-based user interface that would benefit from the Swift functions and enable its use without the need of an installation.

A Obsah přiloženého CD/DVD

doc/
: Bakalářská práce ve formátu PDF včetně všech příloh a souborů potřebných k vygenerování této práce.

src/swift_fca/
: Zdrojové kódy programu Swift.

src/swift-cli.py
: Skript pro použití konzolového rozhraní programu Swift.

src/swift.py
: Skript pro spuštění grafického rozhraní programu Swift.

README
: Stručný popis programu Swift, instrukce pro instalaci, příklad použití a odkaz na webové stránky projektu obsahující manuály.

LICENSE
: Licence GNU GPL v3.

Navíc CD/DVD obsahuje:

tests/
: Testovací a ukázková data se skripty pro testování správnosti převodů.

html/
: Zdrojové soubory webové stránky projektu obsahující manuály v HTML formátu.

html/manual.html
: Hlavní manuál k programu Swift v angličtině.

html/guimanual.html
: Dodatečný manuál ke grafickému rozhraní programu Swift v angličtině.

Literatura

[1] LICHMAN, M. UCI Machine Learning Repository [online]. Irvine, CA: University of California, School of Information and Computer Science, 2013 [cit. 2016-05-03]. Dostupné z: http://archive.ics.uci.edu/ml

[2] LICHMAN, M. UCI Machine Learning Repository [online]. Irvine, CA: University of California, School of Information and Computer Science, 2013 [cit. 2015-12-17]. Dostupné z: http://archive.ics.uci.edu/ml/about.html

[3] Frequent Itemset Mining Dataset Repository [online]. [cit. 2016-05-03]. Dostupné z: http://fimi.ua.ac.be/data/

[4] BĚLOHLÁVEK, Radim. Konceptuální svazy a formální konceptuální analýza [online]. [cit. 2015-12-16]. Dostupné z: http://belohlavek.inf.upol.cz/publications/Bel_Ksfka.pdf

[5] SIPSER, Michael. Introduction to the theory of computation. Boston: PWS Publishing, 1997, xv, 396 s. ISBN 05-349-4728-X.

[6] SHAFRANOVICH, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files [online]. SolidMatrix Technologies, Inc., 2005 [cit. 2016-02-27]. Dostupné z: http://www.ietf.org/rfc/rfc4180.txt

[7] Comma-separated values. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-04-30]. Dostupné z: https://en.wikipedia.org/wiki/Comma-separated_values

[8] ARFF (stable version) [online]. [cit. 2016-02-27]. Dostupné z: http://weka.wikispaces.com/ARFF+%28stable+version%29

[9] VINTERBO, Staal. The ARFF Package [online]. [cit. 2016-05-01]. Dostupné z: http://laats.github.io/sw/mit/arff/

[10] Attribute-Relation File Format (ARFF). The University of Waikato - Department of Computer Science [online]. 2008 [cit. 2016-01-24]. Dostupné z: http://www.cs.waikato.ac.nz/ml/weka/arff.html

[11] Weka 3: Data Mining Software in Java. Machine Learning Group at the University of Waikato [online]. [cit. 2016-01-24]. Dostupné z: http://www.cs.waikato.ac.nz/ml/weka/index.html

[12] QUINLAN, J. C4.5: programs for machine learning. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993, x, 302 p. ISBN 15-586-0238-0.

[13] C4.5 Manual Page [online]. UNIVERSITY OF REGINA DBD. [cit. 2016-01-26]. Dostupné z: http://www2.cs.uregina.ca/ dbd/cs831/notes/ml/dtrees/c4.5/c4.5.html

[14] C4.5 Format [online]. [cit. 2016-04-30].
Dostupné z: http://www.cs.washington.edu/dm/vfml/appendixes/c45.htm

[15] PRISS, Uta. FCA File Formats [online]. [cit. 2016-02-28].
Dostupné z: http://www.upriss.org.uk/fca/fcafileformats.html#Burmeister

[16] NIK BESSIS AND FATOS XHAFA (EDS.). Next generation data technologies for collective computational intelligence. Berlin: Springer, 2011. ISBN 978-364-2203-442.

[17] VYCHODIL, Vilem. FCALGS: Data Formats [online]. 2009 [cit. 2016-01-27].
Dostupné z: http://fcalgs.sourceforge.net/format.html

[18] VYCHODIL, Vilem. FCALGS [online]. 2009 [cit. 2016-05-03].
Dostupné z: http://fcalgs.sourceforge.net/

[19] Download Python for Other Platforms. Python.org [online]. [cit. 2016-01-30].
Dostupné z: https://www.python.org/download/other/

[20] Python. [online]. [cit. 2016-01-30]. Dostupné z: https://www.python.org/

[21] What is PyQt? Riverbank [online]. [cit. 2016-01-30].
Dostupné z: https://www.riverbankcomputing.com/software/pyqt/intro

[22] Pyparsing [online]. [cit. 2016-01-30].
Dostupné z: http://pyparsing.wikispaces.com/

[23] CORMEN, Thomas H. Introduction to algorithms. 2nd ed. Cambridge, Mass.: MIT Press, c2001. ISBN 00-701-3151-1.

[24] Python Time Complexity[online]. [cit. 2016-02-23].
Dostupné z: https://wiki.python.org/moin/TimeComplexity

[25] ANDREWS, S. and ORPHANIDES, C. (2010). FcaBedrock, a formal context creator [online]. [cit. 2016-02-26].
Dostupné z: http://shura.shu.ac.uk/2104/1/ICCS2010_accepted_paper.pdf

[26] FcaBedrock Formal Context Creator [online]. [cit. 2016-02-26].
Dostupné z: https://sourceforge.net/projects/fcabedrock/

[27] PRISS, Uta. FcaStone - FCA file format conversion and interoperability software [online]. , 11 [cit. 2016-02-26].
Dostupné z: http://www.upriss.org.uk/papers/ictools08.pdf

[28] PRISS, Uta. FcaStone: software for FCA file format conversion and interoperability [online]. [cit. 2016-02-26].
Dostupné z: http://fcastone.sourceforge.net/