Table of Contents

ITAT'07

Information Technology – Applications and Theory

STUDENT PAPERS

Udržiavanie charakteristík v modeli používateľa kladením otázok využívajúcich koncepty doménovej ontológie*

Tomáš Klempa, Anton Andrejko, and Mária Bieliková

Ústav informatiky a softvérového inžinierstva, Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 3, 842 16 Bratislava, Slovensko
tomas.k@zmail.sk, {andrejko,bielik}@fiit.stuba.sk

Abstrakt *V adaptívnych webových aplikáciách je proces personalizácie založený na modeli používateľa, ktorý je tvorený charakteristikami používateľa. Aby bolo prispôsobovanie efektívne, je potrebné, aby model používateľa vždy odrážal reálneho používateľa. Preto je potrebné model používateľa neustále aktualizovať. V príspevku opisujeme metódu explicitného získavania a udržiavania charakteristík v modeli používateľa kladením otázok v prirodzenom jazyku. Kontext otázky závisí na doménovej ontológii, ktorej koncepty sú viazané prostredníctvom internej reprezentácie. Celý proces získavania a udržiavania charakteristík je riadený používateľom definovanými pravidlami. Výsledkom navrhnutej metódy je softvérový prototyp.*

1 Úvod

Web predstavuje rozsiahly zdroj informácií z mnohých oblastí, pričom množstvo informácií neustále rastie. Informačný obsah webu nie je vždy vhodne rozmiestnený, čo spôsobuje problémy s navigáciou, prípadne sa používateľ brodí obsahom, o ktorý nemá záujem a pod. Spôsob, ako zefektívniť prácu s informáciami ponúka personalizácia, kde sa vybrané črty používateľa reprezentujú v modeli používateľa a použijú sa na prispôsobenie rôznych viditeľných aspektov systému.

Model používateľa uchováva abstrahované charakteristiky používateľa. Charakteristiky používateľa je možné získavať explicitným alebo implicitným spôsobom. Explicitný spôsob je založený na priamej odozve od používateľa a implicitný spôsob využíva sledovanie aktivity používateľa. Nevýhodou explicitného spôsobu je, že môže vyrušovať používateľa pri práci. Naopak získané informácie sú spravidla presnejšie ako pri implicitnom spôsobe, kedy je nevyhnuté sledované charakteristiky správne interpretovať, aby mohli byť uložené v modeli používateľa. Preto sa obidva prístupy často kombinujú.

Väčšina webových adaptívnych aplikácií kladie dôraz najmä na prispôsobovanie viditeľných aspektov systému. Charakteristiky v modeli používateľa sa, podobne ako aj používateľ, v čase menia. Dôsledkom neudržiavania charakteristík je neefektívne prispôsobovanie, keďže model neodráža aktuálne charakteristiky reálneho používateľa.

Spôsob aktualizácie modelu používateľa založený na sledovaní aktivity používateľa je opísaný v [9]. Využíva informáciu o návšteve konceptu, ako aj kombináciu počtu návštev konceptu s časom stráveným v koncepte, čím eliminuje možnosť chyby vzniknutej na základe nesprávnych uzáverov ohľadom času zobrazenia príslušného konceptu. Získavanie charakteristík procesom učenia využíva systém UPRE [7].

Explicitnú spätnú väzbu využíva väčšina adaptívnych výučbových aplikácií na zistenie vedomosti používateľa o zobrazenom koncepte [2,5], čo sa prejaví aj v modeli používateľa, keďže vo výučbových systémoch model používateľa spravidla prekrýva model aplikačnej domény.

V príspevku opisujeme metódu inicializácie a udržiavania charakteristík v modeli používateľa, ktorá používa explicitný spôsob – odpovede na otázky. Motiváciou pre tento spôsob bol projekt M-PIRO, ktorý využíva sémantiku poskytovanú ontologickou reprezentáciou na generovanie textu v prirodzenom jazyku [1]. Nástroj vytvorený v rámci projektu generuje texty z ontológie, ktorá poskytuje informácie o entitách v doméne. V [8] je opísaný prístup plánovania výrokov v prirodzenom jazyku, ktoré môžu dynamicky využiť kontextové informácie o prostredí, v ktorom prebieha dialóg. Iným príkladom je vzdelávací systém OntoAIMS [6], kde používateľ komunikuje so systémom prostredníctvom grafického rozhrania, ktoré pozostáva z dialógu (s vopred pripravenými otázkami) a grafu, ktorým možno vytvárať a modifikovať vyhlásenia grafickým spôsobom.

V navrhnutej metóde využívame model aplikačnej domény reprezentovaný ontológiou a jeho koncepty na vygenerovanie otázok v prirodzenom (anglickom) jazyku s cieľom aktualizovať charakteristiky v modeli používateľa. Odpovede na tieto otázky sa priamo prejavia zmenou charakteristík v modeli. Proces kladenia otázok je riadený definovanými pravidlami. Do-

* Tento príspevok vznikol za podpory Štátneho programu výskumu a vývoja "Budovanie informačnej spoločnosti" č. úlohy 1025/04 a Vedeckej grantovej agentúry VEGA v rámci grantovej úlohy č. VG1/3102/06.

plnením priority k jednotlivým charakteristikám vieme vygenerovanie otázky úplne potlačiť.

2 Princíp generovania otázky

Pri explicitnom získavaní a udržiavaní charakteristík v modeli používateľa využívame priamu odozvu od používateľa, ktorá môže byť realizovaná napr. vyplnením ankety, formulára, odpoveďou na otázku a pod. Pri kladení otázok rozlišujeme (1) otázky, na ktoré možno odpovedať chýbajúcou časťou a (2) otázky s kladnou alebo zápornou odpoveďou. Pri návrhu metódy vychádzame z princípu, kde názov charakteristiky používateľa je priamo časťou otázky. Názov charakteristiky úzko súvisí s jej reprezentáciou v modeli používateľa prostredníctvom elementu vlastnosť (*property*).

Na obrázku 1 je uvedený príklad štyroch otázok a odpovedí, ktorý demonštruje opísaný spôsob generovania otázok. V otázke, kde očakávame nejaký údaj (napr. číselný, textový, ordinálny alebo booleovský) sa použije názov charakteristiky (zvýraznený podčiarknutým písmom). Odpoveď môže opäť obsahovať názov charakteristiky a jej príslušnú hodnotu, ktorú zadá používateľ (zvýraznené tučným šikmým písmom). Táto hodnota sa zapíše do modelu používateľa.

Q: What is your name?

A: My name is ***John***.

Q: Have you got any driving experiences?

A: ***Yes***, I have got.

Q: Please enter your expected salary range.

A: I expect salary from ***25.000*** to ***40.000***.

Q: How often are you willing to travel for a job?

A: ***Every day***.

Obrázok 1. Príklad otázok a odpovedí.

Pre takto navrhnutý spôsob generovania otázok, sme zvolili princíp šablón otázok. Šablóna otázky je vopred definovaná reprezentácia otázky, ktorá využíva slovnú zásobu. Postupnosť slov v šablóne otázky je vopred definovaná autorom šablóny. Slovo, ktoré má nahradiť názov charakteristiky, je označené špecifickou značkou. S kladením otázky súvisí nasledujúci kontext:

- *okolnosti* – čo musí nastať, aby bola pre používateľa vygenerovaná otázka,
- *predmet otázky* – aká otázka sa má vygenerovať, resp. na čo sa má systém "opýtať".

Okolnosti, kedy bude používateľovi vygenerovaná otázka, sú kontrolované pomocou pravidiel a predmet otázky závisí od charakteristiky.

Na obrázku 2 je znázornený princíp generovania otázky. Obrázok slúži iba na základné vysvetlenie navrhnutej metódy, a preto neobsahuje úplný tok spracovania údajov.

Predpokladajme, že modelujeme znalosti používateľa v rámci aplikačnej domény pracovných ponúk. Na začiatku vytvoríme zoznam charakteristík pracovnej ponuky, na ktoré sa chceme vygenerovanými otázkami opýtať a takto inicializovať nové, prípadne aktualizovať existujúce charakteristiky v modeli používateľa. Postup získavania a následného vytvorenia novej (neexistujúcej) inštancie charakteristiky sa odlišuje od udržiavania (aktualizácie) existujúcej charakteristiky.

V prípade vytvorenia novej inštancie charakteristiky sú z *Úložiska viazania* vybrané charakteristiky, ktoré nemajú doposiaľ vytvorenú inštanciu charakteristiky v modeli používateľa. Pri udržiavaní sú zvolené charakteristiky na základe vyhodnotenia pravidiel. V obidvoch prípadoch je však vytvorený zoznam jednoznačných pomenovaní charakteristík, usporiadaných podľa priority. Priorita rieši aj prípady, kedy je vhodné vygenerovanie otázky potlačiť (napr. pohlavie používateľa je charakteristika, ktorú stačí zistiť raz).

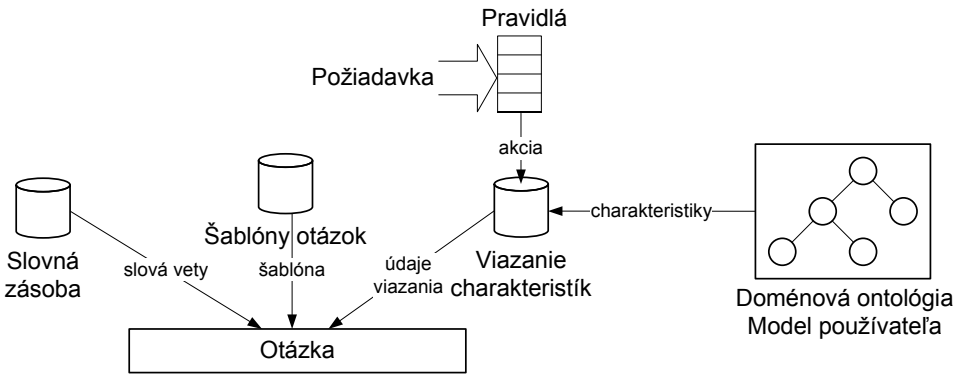
Pomenovanie a priorita charakteristiky, spolu s jej štruktúrou, sú uložené v *Úložisku viazania*. K charakteristike je tiež naviazaná šablóna otázky a podstatné meno, ktoré je vo vygenerovanej otázke použité ako názov charakteristiky. Podľa jedinečného identifikátora charakteristiky sa vyhľadá v *Úložisku šablón* otázok príslušná šablóna. Jednotlivé slová sú uložené v *Slovnej zásobe*, odkiaľ sa pre príslušnú šablónu načítajú. Týmto spôsobom vznikne otázka, ktorá je položená používateľovi. Používateľ taktiež získa časť vopred pripravenej odpovede v prípade ak je odpoveďou hodnota z množiny typov ordinálna hodnota alebo voľba.

2.1 Slovná zásoba

Pre slovnú zásobu sme pôvodne navrhli a vytvorili ontológiu *Part of Speech*, ktorá uchováva slová vo forme inštancií, pričom každé slovo má špecifikovaný slovný druh a gramatickú kategóriu. Od tohto prístupu sme ustúpili z dôvodu časovej náročnosti potrebnej na naplnenie ontológie a nízkej efektívnosti softvérových prostriedkov využívajúcich ontológie, čo by sa prejavilo časovo kritickým generovaním otázok.

Používame slovnú zásobu, ktorá predstavuje použitie výsledkov existujúceho projektu – trénovacie a testovacie údaje využité pre určenie slovných druhov v texte (Text Chunking[1]). Údaje sú reprezentované vo

[1] http://www.cnts.ua.ac.be/conll2000/chunking/



Obrázok 2. Doménovo závislá časť modelu používateľa.

forme *identifikátor;slovo;značka slovného druhu*, napr. *25;ability;NN* (*POS tag*) v anglickom jazyku, kde prvý stĺpec reprezentuje identifikátor slova, na ktorý sa odkazuje v šablóne otázok, nasleduje samotné slovo a značka slovného druhu. Značka slovného druhu v niektorých prípadoch určuje nielen slovný druh, ale aj gramatickú kategóriu slova (*NN* – podstatné meno v jednotnom čísle). Po vyčistení slovnej zásoby (odstránenie čísel, prebytočných členov a pod.) obsahuje približne 17.000 slov. Značky slovných druhov špecifikujú nasledovné gramatické kategórie:

1. *číslo* – použitie pri podstatných menách, zámenách; môže byť jednotné a množné,
2. *čas* – pri slovesách; minulý, prítomný a budúci,
3. *osoba* – pri podstatných menách, zámenách a slovesách; prvá, druhá a tretia osoba.

Hlavnou výhodou tohto spôsobu reprezentácie je skutočnosť, že poznáme sémantiku jednotlivých slov, z ktorých sa skladajú otázky a vyššia efektivita spracovania oproti využitiu ontológie *Part Of Speech*. Slovná zásoba navyše obsahuje dostatočný počet slov, takže nepredpokladáme potrebu jej udržiavania. Údržba je vzhľadom na použitý formát slovnej zásoby jednoduchá.

2.2 Šablóny

Z dôvodu znovupoužitia šablón, ako aj možnosti kedykoľvek vymeniť slovnú zásobu, sme sa rozhodli oddeliť slovnú zásobu od šablón otázok. Šablóna otázky pozostáva zo slov, ktoré sú reprezentované v XML súbore prostredníctvom identifikátora, tak ako sú uložené v slovnej zásobe.

Na nasledujúcom výpise je znázornený príklad šablóny otázky *ValueTemplate01*:

```
<question name="ValueTemplate01">
  <pos order="1" instanceName="15432"/>
  <pos order="2" instanceName="7398"/>
  <pos order="3" instanceName="12456"/>
  <pos order="4" instanceName="n01"/>
</question>
```

Element *question* obsahuje elementy *pos*, ktoré špecifikujú identifikátory slov v slovnej zásobe. Identifikátory 15432, 7398, 12456 budú neskôr nahradené slovami *what*, *is*, *your*. Špecifikované je aj poradie slov v otázke prostredníctvom atribútu *order*.

Vytvorili sme niekoľko šablón otázok. So zvyšujúcim sa počtom šablón otázok, ktoré budú k dispozícii, bude konverzácia s používateľom prirodzenejšia. Avšak veľký počet šablón, t.j. stav, kedy je pre každú vygenerovanú otázku vytvorená samostatná šablóna, nie je veľmi vhodný z hľadiska efektívnosti ich vytvárania. Z tohto dôvodu zohľadňujeme pri vytváraní šablón otázok a ich viazaní na konkrétnu charakteristiku v modeli používateľa univerzálnosť šablóny.

3 Viazanie charakteristík

Pre viazanie sme sa rozhodli z praktického dôvodu, keďže priamo do ontológie nemusíme pridávať koncepty, ktoré by špecifikovali údaje potrebné pre generovanie otázky, ako napríklad priority charakteristík. Do ontológie pristupujeme iba vtedy, ak potrebujeme načítať konkrétne hodnoty charakteristiky alebo sa vytvoria, resp. aktualizujú príslušné inštancie charakteristiky.

Viazaním obohacujeme charakteristiky modelu používateľa o špecifické údaje potrebné pre navrhnutú metódu. Špecifickými údajmi sú:

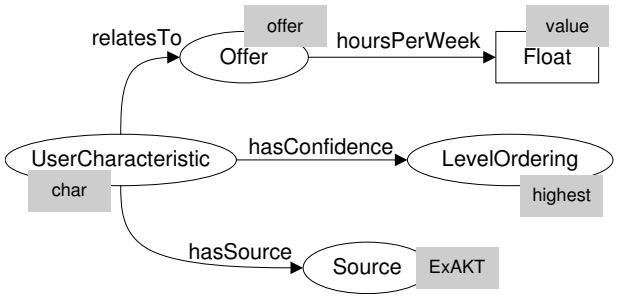
- jedinečný identifikátor charakteristiky,
- priorita charakteristiky,
- šablóna otázky a
- podstatné meno použité vo vete, ktoré predstavuje názov charakteristiky (nemusí byť jedinečné).

Charakteristika je v ontológii zložená z rôznych konceptov. Pri viazaní štruktúry charakteristiky viažeme nasledujúce koncepty:

- priestory mien (*namespaces*),
- triedy (*classes*),
- vlastnosti (*properties*) dátové alebo objektové a
- inštancie.

Jednoduché charakteristiky majú tvar RDF trojice, t.j. subjekt, predikát a objekt, kde subjekt odpovedá triede charakteristiky, predikát názvu a objekt hodnote charakteristiky. Takáto štruktúra je jednoducho viazateľná. Pri experimentoch s ontológiou pracovných ponúk sme zistili, že uvedený spôsob nepostačuje pre reprezentáciu komplexnejších charakteristík, ktoré môžu spolu súvisieť, môžu vzájomne vytvárať taxonómiu alebo ak je charakteristika vyjadrená ordinálnou hodnotou (jednotlivé ordinálne hodnoty môžu byť reprezentované triedou alebo inštanciou).

Pri takýchto charakteristikách je zložitejší aj spôsob vytvorenia, resp. aktualizácie charakteristiky. Obrázok 3 znázorňuje štruktúru časti modelu používateľa pre charakteristiku vyjadrujúcu používateľom preferovaný počet pracovných hodín v týždni, pričom pri pridaní charakteristiky sa určí aj jej ďalší atribút – dôveryhodnosť.



Obrázok 3. Príklad štruktúry konceptov pre reprezentáciu počtu pracovných hodín do týždňa.

Pre ľahšie pochopenie štruktúry sme názvy tried a atribútov nahradili jednoduchými symbolickými menami. Trieda *UserCharacteristic* predstavuje samotnú charakteristiku, má reláciu s pracovnou ponukou (*Offer*), ktorá má dátovú vlastnosť *hoursPerWeek* typu reálne číslo (*Float*). Charakteristika má reláciu vyjadrujúcu jej dôveryhodnosť, ktorá je ordinálnou hodnotou reprezentovanou inštanciami triedy *LevelOrdering* a reláciu *hasSource* s triedou vyjadrujúcou zdroj, ktorý charakteristiku do modelu používateľa pridal. V sivých obdĺžnikoch sú uvedené názvy inštancií, ktoré majú byť vytvorené po zadaní používateľovej odpovede.

Podobne sme navrhli spôsob pomenovania aj pre inštancie, aby bola zachovaná štruktúra konceptov pri vytvorení inštancie. Napríklad pre triedu *UserCharacteristic* a jej inštanciu *char* sme určili, že bude obsahovať dodatočný sufix: časovú pečiatku. Tento spôsob vytvárania inštancie sme pomenovali vzor (*pattern*). Rovnaký princíp uplatníme aj pre inštanciu *offer*. Inštancia *highest* (v rámci ontológie už existuje) vyjadrujúca úroveň dôveryhodnosti bude prepojená s inštanciou *char*. Inštancia triedy *Source* predstavuje názov nástroja, ktorý urobil zmenu charakteristiky ako posledný, bude mať názov *ExAKT*.

Pre zhrnutie, definovali sme tri spôsoby pomenovania inštancie:

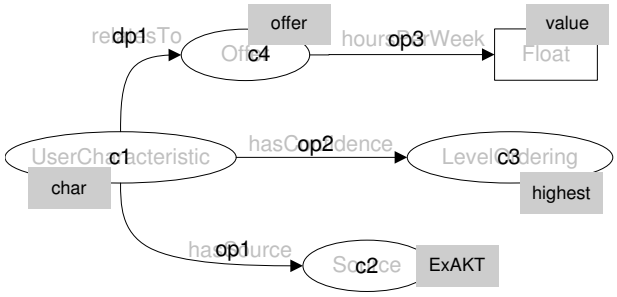
- *pomenovanie podľa vzoru* – vytvorí sa inštancia podľa stanoveného názvu so sufixom časovej pečiatky, pričom jej formát je konfigurovateľný,
- *pomenovanie podľa názvu* – vytvorí sa inštancia podľa definovaného názvu,
- *nastavenie želanej hodnoty* (ordinálna, ale aj textová, numerická) – k relácii bude priradená želaná hodnota.

Pre reprezentáciu viazaných konceptov sme zvolili spôsob pomenovania podľa jednoznačných identifikátorov – tzv. kľúčov, ktoré sú uložené v *Úložisku viazania*. Názov kľúča pozostáva z písmena (c – trieda, op – objektová vlastnosť, dt – dátová vlastnosť, i – inštancia) a *postfixu* – číslo poradia v rámci *Úložiska viazania*.

Každý kľúč má priradený názov, podľa ktorého sa odvolávame na daný koncept, k nemu príslušný názov konceptu, priestor mien konceptu (v prefixovom tvare) a názov a typ inštancie, ktorá sa má vytvoriť. Využitie viazania prostredníctvom kľúčov zvyšuje flexibilitu viazania štruktúry v prípade, ak dôjde k zmenám v doménovej ontológii, resp. v modeli používateľa. Úložisko viazania zároveň obsahuje aj zoznam priestorov mien viazanej ontológie (pre viazanie prefixového tvaru na kompletný priestor mien vo forme URI). Obrázok 4 ilustruje štruktúru ontológie s uvedením jednotlivých kľúčov priradených ku konceptom.

Ak by charakteristika obsahovala kompletnú štruktúru konceptov, t.j. relácie medzi triedami a inštanciami, bolo by viazanie neefektívne z hľadiska veľkej závislosti na štruktúre charakteristiky v modeli používateľa. Namiesto tohto sa načítajú iba objektové alebo dátové vlastnosti. Ich doménové triedy a triedy rozsahu sa načítajú prostredníctvom špecifických dopytov.

Môže však nastať prípad, kedy jedna viazaná vlastnosť (objektová alebo dátová) obsahuje viacero doménových tried alebo tried rozsahu. Vtedy je potrebné pre konkrétnu vlastnosť definovať, ktorá jej doménová



Obrázok 4. Kľúče konceptov.

trieda alebo trieda rozsahu bude použitá pre vytvorenie inštancie. Keďže táto definícia je uvedená iba v konkrétnej charakteristike, neovplyvní to ostatné viazané charakteristiky, ktoré obsahujú rovnakú vlastnosť.

Počas experimentov s viazaním štruktúry charakteristiky sme zistili, že v niektorých prípadoch je nevyhnutné špecifikovať konkrétnu doménovú triedu alebo triedu rozsahu (napr. ak je vlastnosť špecifikovaná pre konkrétnu triedu, ale trieda má niekoľko potomkov, pričom chceme špecifikovať vlastnosť pre konkrétneho potomka).

Na nasledujúcom výpise je znázornená časť charakteristiky *hoursPerWeek*:

```
<properties characteristic="hoursPerWeek"
     priority="2">
  <property name="op3" noun="none"
     sentenceTemplate="none" domainConcept="c1"
     rangeConcept="none" increment="false"/>
  <property name="op8" noun="none"
     sentenceTemplate="none"
     domainConcept="c2" rangeConcept="c100"
     increment="false">
    <setOptionValue name="c100" value="dp4"/>
  </property>
  <property name="op9" noun="hours per week"
     sentenceTemplate="QuantityTemplate"
     domainConcept="c2" rangeConcept="dp5"/>
</properties>
```

Špecifikácia štruktúry charakteristiky je ohraničená elementom *properties*. Atribút *characteristic* definuje názov charakteristiky. Vlastnosti (dátové a objektové) sú reprezentované elementom *property*. Atribút *name* je identifikátor konceptu, *noun* špecifikuje podstatné meno, ktoré sa použije v otázke, *sentenceTemplate* špecifikuje názov šablóny otázky, *domainConcept* a *rangeConcept* špecifikujú triedy domény a rozsahu – implicitne sú nastavené na *none*, v prípade ak sa požaduje explicitne špecifikovať triedy, uvedie sa názov konceptu. Atribút *increment* je aplikovateľný iba v prípade číselných literálov, napr. počet aktualizácií charakteristiky. Ak atribút *increment* obsahuje hodnotu *true*, zodpovedajúci literál bude inkrementovaný pri každej aktualizácií charakteristiky. Element *setOptionValue* nastavuje hodnotu typu voľba ako triedu rozsahu v rámci elementu *property*.

4 Pravidlá

Pre efektívne udržiavanie charakteristík v modeli používateľa sme zvolili využitie pravidiel. Výhodou tohto prístupu je jednoduchosť vytvárania a údržby. Využívame klasický koncept pravidiel: *IF* [*podmienka*] *THEN* [*akcia*]. Súčasťou podmienky je vlastnosť z modelu používateľa a k nej príslušný operand a parameter podmienky. Akcia je štandardne určená ako vygenerovanie otázky. Pre definovanie podmienky sme navrhli využitie údajových typov – literálov a objektov, ktoré môžu reprezentovať hodnotu charakteristiky. Literály sú reprezentované údajovými typmi jazyka XML. Objektové hodnoty sú nasledujúce:

- *ordinálna hodnota* – vyjadruje mieru charakteristiky, ktorá nie je reprezentovaná číslom, ale jej hodnoty možno zoradiť a vzájomne porovnávať a
- *voľba (option)* – vyjadruje nemerateľnú charakteristiku (napr. znalosti z programovacieho jazyka Java).

V podmienke môžu byť použité známe operandy (menší, väčší, rovný) a špeciálny operand inštancia (*instanceOf*), ktorý sa využije iba pri hodnote typu voľba a môže vyjadrovať podmienku relácie medzi inštanciou a triedou. V prípade literálov musí byť parametrom podmienky zodpovedajúci typ, t.j. pre číselný literál (patrí sem aj dátum) sa použije číselná hodnota parametra a pre reťazec hodnota typu reťazec. V prípade literálu typu dátum je nevyhnutné uviesť časovú jednotku, ktorá môže byť v rozsahu minúty až roky. V prípade objektových hodnôt sú to existujúce názvy inštancií alebo tried.

Vhodne navrhnutým pravidlom môžeme vyvolať vygenerovanie otázky, napr. ak špecifická charakteristika je staršia ako definované časové obdobie. Ďalším príkladom môže byť vygenerovanie otázky pre charakteristiku v závislosti od toho, ktorý nástroj zabezpečil jej vytvorenie, resp. aktualizáciu.

Nasledujúci výpis predstavuje vybrané pravidlá:

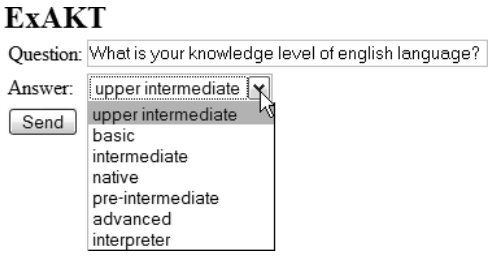
```
<rule type="dateTime">
  <condition parameter="gu:hasTimeStamp"
    operator="less" value="1" unit="minute"/>
  <action type="ask"/>
</rule>
<rule type="ordinalValue">
  <condition parameter="c:hasConfidence"
    operator="less" value="c:_loAverage"/>
  <action type="ask"/>
</rule>
```

5 Zhodnotenie

Väčšina adaptívnych webových systémov sa zameriava na prispôsobovanie viditeľných aspektov (obsahu, navigácie, prezentácie) na základe modelu používateľa. Prínosom tohto príspevku je metóda na prispôsobovanie samotného modelu používateľa, ktorého aktuálnosť je nevyhnutná pre efektívne prispôsobovanie.

Metóda explicitného získavania a udržiavania charakteristík v modeli používateľa je založená na kladení otázok v prirodzenom jazyku. Na vygenerovanie otázok využívame koncepty modelu aplikačnej domény reprezentovaného ontológiou.

Navrhnutá metóda bola overená softvérovým nástrojom ExAKT, ktorý je implementovaný v jazyku Java. Na obrázku 5 je zobrazený výstup z nástroja – vygenerovaná otázka s možnosťou voľby odpovede. Alternatívy odpovede sú získané z doménového modelu. Nástroj pri práci s ontologickými modelmi využíva rámec Sesame (*http://www.openrdf.org*). Modely sú reprezentované v jazyku OWL DL.



Obrázok 5. Používateľské rozhranie pre odpoveď s viacerými hodnotami.

Overenie prebehlo na doménovej ontológii pracovných ponúk zúženej na oblasť informatiky (vytvorená v rámci projektu NÁZOU [4]), ontológii publikácií (projekt MAPEKUS [3]) a odpovedajúcich modeloch používateľa. V obidvoch prípadoch sme naviazali vybrané charakteristiky, vytvorili pre ne šablóny otázok a definovali pravidlá pre udržiavanie. Overili sme vytvorenie nových charakteristík (vhodné na inicializáciu modelu používateľa pri prvom prihlásení používateľa do systému) a aktualizáciu existujúcich charakteristík v modeli používateľa.

Pri testovaní metódy na doménovej ontológii cestovného ruchu sme narazili na ohraničenie metódy, ktoré spočíva vo viazaní konceptov. Napr. chceli by sme otázkou zistiť ohodnotenie ubytovanie (trieda *Accomodation*) v rámci destinácie (trieda *Destination*) a zistiť kontaktné údaje. Vytvorené by však boli dve nezávislé inštancie. To odporuje metóde, ktorá očakáva v otázke názov jednej charakteristiky.

Pri vyhodnotení univerzálnosti šablón sme sledovali počet otázok, ktoré možno vygenerovať z jednej šablóny. Pre doménu pracovných ponúk sme vytvorili 7 šablón, pomocou ktorých sme vygenerovali otázky pre 22 charakteristík. Pre doménu publikácií sme vytvorili 3 šablóny a vygenerovali 5 otázok. V prípade podobnosti štruktúry viacerých charakteristík sa miera univerzálnosti využitia šablón zvyšuje.

Aktualizácia jednotlivých charakteristík používateľa priamo ovplyvnila 10 – 16 výrokov v modeli používateľa. Preto sme sa zamerali aj na vyhodnotenie času potrebného na aktualizáciu charakteristiky. Ten sa pohyboval rádovo v jednotkách sekúnd v závislosti od komplexnosti charakteristiky. Pre 10 výrokov sme dosiahli priemerný čas 4,7 sekundy a 6,4 sekundy pre 16 výrokov.

Pri overení metódy sme sa zamerali na udržiavanie charakteristík jedného používateľa. V ďalšej práci plánujeme rozšírenie pre viacerých používateľov. Zároveň pracujeme na implementácii nástroja pre poloautomatizované viazanie charakteristík modelu používateľa.

Referencie

1. Androutsopoulos, I., Kallonis, S., Karkaletsis, V.: Exploiting OWL Ontologies in Multilingual Generation of Object Descriptions. In Proceedings of the 10th European Workshop on Natural Language Generation, Aberdeen Scotland, August (2005) 150–155
2. Bieliková, M.: An adaptive web-based system for learning programming. Int. J. Continuing Engineering Education and Life-Long Learning, Inderscience, **16** (2006) 122–136
3. Bieliková, M., Návrat. P.: Modelovanie a získavanie, spracovanie a využívanie znalostí o konaní používateľa v hyperpriestore Internetu. In Proc. of Znalosti 2007, Ostrava, Česká republika, Február (2007) 368–371
4. Bieliková, M., Návrat, P., Vojtáš, P., Hluchý, L., Bartoš, P.: Softvérové nástroje pre získavanie, organizovanie a prezentáciu pracovných ponúk na webe. In Proc. of Datakon 2006, Brno, Česká republika (2006) 1–20
5. De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, S., Stash, N.: AHA! The adaptive hypermedia architecture, In Proceedings of the ACM Conf. on Hypertext and Hypermedia, Nottingham, UK (2003) 81–84
6. Denaux, R., Aroyo, L., Dimitrova, V.: An Approach for Ontology based Elicitation of User Models to Enable Personalization on the Semantic Web. In Proceedings of WWW 2005 conference, ACM Press (2005) 1170–1171
7. Gurský, P., Horváth, T., Novotný, R., Vaneková, V., Vojtáš, P.: UPRE: User preference based search system. In Int. Conf. on Web Intelligence. (2006) 841–844
8. Kruijff, G.M.: Context sensitive utterance planning for CCG. In Proc. of the 10th European Workshop on Natural Language Generation, Scotland (2005) 83–90
9. Šimún, M., Andrejko, A., Bieliková, M.: Inicializácia a aktualizácia modelu používateľa pri hľadaní pracovných ponúk na webe. In Proc. of Znalosti 2007. Ostrava, Česká republika, Február (2007) 109–120

Dynamická zmena harmonogramu pomocou minimálneho rezu grafu*

Marián Lekavý and Pavol Návrat

Slovak University of Technology, Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
{lekavy,navrat}@fiit.stuba.sk

Abstrakt *Systémy pre riadenie toku práce majú za úlohu zefektívniť postup projektu tým, že na seba preberú veľké množstvo administratívnej práce. Hlavnou úlohou takéhoto systému je plánovať, monitorovať a podporovať aktivity vykonávané agentmi (či už ľuďmi alebo softvérovými agentmi). Dôležitou úlohou je zaistiť splnenie termínov projektu. Tento článok sa venuje algoritmu, ktorého úlohou je vytvorenie nového harmonogramu v prípade, že niektorá aktivita nebola dokončená v plánovanom čase. Nový harmonogram je vytváraný skrátením a posunutím závislých aktivít. Algoritmus je založený na algoritme pre minimálny rez grafu, ktorý zaisťuje, že cena zmeny harmonogramu je minimálna.*

1 Úvod

Riadenie toku práce je automatizácia procesu alebo jeho časti, v ktorom dochádza k prenosu dokumentov, informácií alebo úloh medzi jednotlivými účastníkmi podľa sady procedurálnych pravidiel [4]. Riadenie toku práce skvalitňuje administratívne procesy tým, že šetrí čas a náklady zvýšením kvality práce [14]. Systém riadenia toku práce preberá veľké množstvo práce tým, že spracováva e-maily a iné dokumenty, monitoruje a kontroluje procesy, poskytuje účastníkom toku práce užitočné kontextovo závislé informácie a pamätá si úspešné riešenia z predchádzajúcich procesov.

Dôležitou časťou toku práce je harmonogram. Bežným problémom je, že harmonogram nie je vždy rešpektovaný na 100%. Ak nie je nejaká aktivita dokončená včas, je potrebné posunúť závislé aktivity, ktoré čakajú na jej výstupy. Harmonogram je obzvlášť dôležitý, ak je potrebné dodržať konečný termín (Deň D) a nie je možné tento termín prekročiť. V takomto prípade je nutné posunúť a/alebo skrátiť čas závislých aktivít.

Zmena harmonogramu je často spojená s preplánovaním, a to obzvlášť vtedy, keď tok práce nie je dostatočne zadefinovaný a stabilný. Pod preplánovaním v tomto článku rozumieme pridanie alebo odobratie aktivít a závislostí medzi nimi. Zmena harmonogramu teda v tomto kontexte nie je preplánovaním. Tento článok sa venuje výhradne zmene harmonogramu bez preplánovania.

Vo všeobecnosti existujú 4 prístupy k zmene harmonogramu:

1. **Vynucovanie harmonogramu, zodpovednosť je prenechaná agentom.** Niektoré systémy pre riadenie toku práce (napríklad EVM použitý pre CERN [1]) sa vyhýbajú zmene harmonogramu výstupovo-orientovaným riadením, ktoré si vynucuje dodržanie termínov a nedovoľuje agentom zmeniť harmonogram akýmkoľvek spôsobom, ktorý by ovplyvnil ostatných. Ak agent nevytvorí požadovaný výstup v požadovanom čase, všetci agenti na ktorých to má vplyv sa musia dohodnúť na novom harmonograme. Je pri tom vkladaná veľká dôvera do schopností agentov splniť harmonogram a do schopnosti vytvoriť v prípade potreby nový harmonogram.
2. **Zaradiť do definície toku práce všetky alternatívy.** Niektoré prístupy (napríklad PROTEUS [2]) zahŕňajú všetky možné scenáre vykonania do definície toku práce. V tomto prípade odpadá nutnosť zmeny harmonogramu za behu. Manuálna definícia všetkých možností je zložitý proces, ale ak sa spraví správne, systém sa dokáže vyrovnať so všetkými situáciami. Je tiež možné nechať v harmonograme rezervy pre nepredvídané udalosti, čím vznikne robustný harmonogram [9].
3. **Nový harmonogram.** Asi najbežnejšia metóda zmeny harmonogramu (okrem manuálnej) je vytvorenie úplne nového harmonogramu (napríklad MicroBoss [12]). Má to však niekoľko nevýhod. Hlavnou nevýhodou je, že strácame informácie získané pri vytváraní pôvodného harmonogramu a tvorba úplne nového harmonogramu je tak stratou času. Ďalšou nevýhodou je to, že nový harmonogram nevychádza z pôvodného, takže množstvo zmien v harmonograme môže byť väčšie, než je potrebné.
4. **Upraviť pôvodný harmonogram.** Mnoho prístupov (napríklad ISIS [6]), vrátane manuálnej zmeny harmonogramu, používa heuristické pravidlá na opravu pôvodného harmonogramu. Tvorba

* Táto práca bola podporená APVT 51-024604; SPVV 1025/04; VG 1/3102/06.

nového harmonogramu potom vyžaduje menšie úsilie, ale nie je zaručené, že zmena harmonogramu bude optimálna vzhľadom na nejakú metriku (zmien je viac, než je nutné). Pri manuálnej zmene harmonogramu je možné použiť rôzne podporné metódy, ako sú metódy založené na CPM (Critical Path Method) [10] alebo PERT (Program Evaluation and Review Technique) [10], ktoré umožňujú optimalizovať harmonogram vzhľadom na dĺžku vykonávania harmonogramu (makespan), avšak neberú do úvahy cenu samotných zmien. Existuje tiež optimálny prístup ktorý minimalizuje cenu zmeny harmonogramu [13] založený na celkovej cene harmonogramu, ktorá zahŕňa aj cenu zmien. Tento prístup vytvorí všetky možné zmeny a vyberie z nich tú s najmenšou cenou. Umožní tak použitie ľubovoľnej cenovej funkcie, ale generovanie všetkých alternatív spôsobuje exponenciálnu výpočtovú zložitosť, čím sťažuje použitie tohto prístupu pre väčšie problémy.

V ideálnom prípade by mal systém na riadenie toku práce vytvárať nový harmonogram automaticky podľa znalostí o toku práce a s využitím informácií o pôvodnom harmonograme. Samotní agenti vytvárajú harmonogram manuálne len v prípade, že systém nedokáže zmeniť harmonogram automaticky. Ak sú však znalosti o toku práce uložené v systéme správne a úplné, potom by manuálne zmeny nemali byť vo väčšine prípadov potrebné.

Tento článok sa venuje algoritmu pre zmenu harmonogramu, ktorý je určený pre systém riadenia toku práce pri procese prípravy vojenských cvičení v Centre simulačných technológií Národnej akadémie obrany v Liptovskom Mikuláši. Tento algoritmus nájde optimálnu zmenu harmonogramu s ohľadom na nejakú metriku (napríklad minimálny počet ovplyvnených aktivít). V najhoršom prípade má algoritmus kubickú výpočtovú zložitosť vzhľadom na počet aktivít a závislostí. Algoritmus sa nevenuje preplánovaniu, preto je použiteľný len pre dobre definované domény, kde sú potrebné aktivity známe a dopredu naplánované.

V 2. kapitole je stručné predstavenie projektu RAPORT, ktoré poskytuje základný kontext použitia predstavovaného algoritmu. Samotný algoritmus pre zmenu harmonogramu je predstavený v kapitole 3. Kapitola 4 uvádza výsledky časovej zložitosti algoritmu.

2 Systém RAPORT

Systém RAPORT [11] je navrhnutý pre pilotnú aplikáciu: príprava vojenských cvičení v Centre simulačných technológií (CST) Národnej akadémie obrany (NAO) v Liptovskom Mikuláši. CST organizuje tréning a výučbu veliteľov s podporou informačných a komunikačných technológií.

Momentálne sú cvičenia organizované manuálne s použitím kancelárskeho softvéru a papierových dokumentov. Aktivity realizujú pracovníci CST, medzi ktorých sú rozdeľované jednotlivé úlohy spojené s prípravou. Koordinácia je realizovaná prostredníctvom kontrolných stretnutí. Príprava jednotlivých cvičení sa môže časovo prekrývať.

Systém pre podporu manažmentu znalostí RAPORT je navrhnutý tak, aby mohol byť použitý pre ľubovoľný riadiaci proces. Je navrhnutý pre splnenie nasledovných požiadaviek:

- Poskytovať informácie vzťahujúce sa k aktuálnemu pracovnému kontextu, roli v organizácii a roli v danej inštancii pracovného procesu (preddefinované e-maily, dokumenty, formuláre atď.)
- Podporovať výmenu skúseností medzi používateľmi tým, že umožní interaktívnu spoluprácu a výmenu poznámok a odporúčaní.
- Kontrolovať dôležité dátumy v harmonograme, vyhodnocovať tok práce pre všetkých agentov a všetky aktivity a v prípade nutnosti prispôsobiť harmonogram.
- Zbierať skúsenosti používateľov a poskytovať ich používateľom v podobnom pracovnom kontexte.

Systém RAPORT kombinuje procesne orientovanú a výstupovo orientovanú paradigmu. Aktivity toku práce sú dobre zadefinované a systém podporuje ich vykonávanie tým, že prideleným agentom poskytuje potrebné dokumenty, opisy procesov a návody. Zároveň každá aktivita končí v momente, keď sú vytvorené všetky výstupné dokumenty tejto aktivity.

V procese toku práce plánovania vojenského cvičenia ktorú spravuje systém RAPORT je harmonogram tvorený dátumami, kedy musia byť ukončené jednotlivé aktivity. Celý harmonogram je relatívny vzhľadom na dátum cvičenia - Deň D. Všetky aktivity majú svoj dátum ukončenia relatívny ku Dňu D. Aby bola príprava cvičenia úspešná, musia byť v Deň D úspešne ukončené všetky aktivity.

Rovnako ako vo viacerých aktuálnych projektoch, súčasťou systému je ontológia použitá na reprezentáciu informácií a znalostí o procese toku práce a získaných skúsenostiach. Pre všeobecný riadiaci proces je vytvorená generická ontológia. Podľa analýzy procesu prípravy a organizácie vojenského cvičenia je potom vytvorená doménová ontológia. Pre algoritmus ktorému sa venuje tento článok však ontológia nehrá podstatnejšiu úlohu.

Ďalšie informácie o projekte RAPORT je možné nájsť v dokumentácii k projektu [11].

3 Zmena harmonogramu ako minimálny rez grafu

Táto kapitola obsahuje základnú štruktúru algoritmu pre zmenu harmonogramu.

Základnou myšlienkou algoritmu je vytvorenie grafu závislostí pre aktuálny harmonogram. V tomto grafe je každá aktivita a každá závislosť reprezentovaná hranou. Následne označkujeme jednotlivé hrany cenami skrátenia týchto hrán. Ak hrana nemôže byť skrátená, cena skrátenia je nekonečná. Toto nastane, ak aktivita reprezentovaná hranou dosiahla minimálnu dĺžku alebo ak hrana reprezentuje závislosť a závislá aktivita začína okamžite po aktivite na ktorej je závislá.

Ak nejaká aktivita (A_{fail}) nesplní dátum svojho ukončenia, algoritmus nájde zmenu harmonogramu s minimálnou cenou, a to pomocou nájdenia minimálneho rezu grafu závislostí. Nájdený minimálny rez, ktorý rozdeľuje aktivitu porušujúcu harmonogram a koncovú aktivitu, zodpovedá optimálnej zmene harmonogramu. Aktivity a závislosti ktoré pretína rez sú skrátené. Aktivity a závislosti v prvej časti grafu (ktorá obsahuje A_{fail}) sú posunuté. Aktivity a závislosti v druhej časti grafu ostanú nezmenené.

Pseudokód celého algoritmu je nasledujúci:

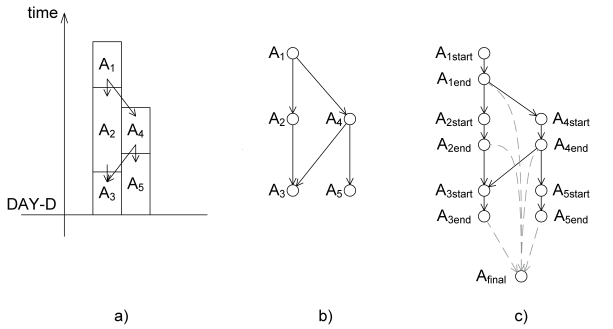
```
1. Vytvor graf závislostí G (kapitola 3.1)
2. Odstráň z G všetky aktivity, ktoré nezávisia
(priamo alebo nepriamo) na A_fail.
3. Označkuj hrany G cenami (kapitola 3.2)
4. Nájdi minimálny rez (kapitola 3.3)
5. Zmeň harmonogram podľa nájdeného minimálneho
rezu (kapitola 3.4)
6. Informuj agentov zodpovedných za zmenené
aktivity.
```

Kontrola plnenia harmonogramu sa robí každodenne (alebo v inom jednotkovom intervale). Aktivity sa preto posúvajú alebo skracujú vždy o 1 deň. Ak potrebujeme presunúť aktivity o viac ako 1 deň, je možné vykonať celý proces niekoľko krát.

Ďalšie podkapitoly obsahujú podrobnejšie informácie o jednotlivých krokoch algoritmu.

3.1 Transformácia na graf

Aktivity sú zviazané prostredníctvom dokumentov. Nejaká aktivita A_{use} je závislá na aktivite $A_{produce}$ ak aktivita A_{use} používa niektorý dokument (D) vytvorený aktivitou $A_{produce}$. A_{use} môže začať až po tom, ako aktivita $A_{produce}$ skončila a vytvorila dokument D. Ak sú nejaké aktivity navzájom závislé, ale neprenáša sa medzi nimi žiadny dokument, do systému sa pridá virtuálny prázdny dokument, ktorý túto závislosť vyjadruje.



Obrázok 1. Príklad konverzie aktivít a závislostí (a) na orientovaný graf závislostí (b), rozdelenie vrcholov aktivít a pridanie koncového vrcholu (c).

Každá aktivita má tiež zadefinované 3 časy: čas začiatku, čas ukončenia a minimálny čas potrebný na jej vykonanie. Na začiatku sa tieto časy inicializujú podľa štandardného harmonogramu toku práce. Čas začiatku a konca sa potom môže ďalej meniť v dôsledku zmeny harmonogramu.

Pre účely tohto algoritmu je teda aktivita vyjadrená ako $A = (id,\ D_{used},\ D_{produced},\ t_{start},\ t_{end},\ t_{min})$, kde id je identifikátor aktivity, D_{used} je množina dokumentov ktoré aktivita potrebuje, $D_{produced}$ je množina dokumentov vytváraných aktivitou, t_{start} a t_{end} sú časy začiatku a konca aktivity v aktuálnom harmonograme a t_{min} je minimálny čas potrebný pre aktivitu ($t_{end} - t_{start} \geq t_{min}$). Definícia aktivity v doménovej ontológii obsahuje aj ďalšie informácie (roly, zodpovedných agentov, šablóny dokumentov, aktívne poznámky a.i.), avšak pre potreby algoritmu pre zmenu harmonogramu ich môžeme zanedbať.

Z modelu toku práce (obrázok 1 a) uloženého v doménovej ontológii môžeme zostrojiť graf závislosti aktivít. V takomto grafe sú aktivity vrcholmi a závislosti hranami (obrázok 1 b).

Pre algoritmus minimálneho rezu potrebujeme, aby boli ako hrany vyjadrené okrem závislostí aj samotné aktivity. Preto musíme rozdeliť každý vrchol grafu závislostí reprezentujúci nejakú aktivitu A na dva vrcholy A_{start} a A_{end}, ktoré budú spojené hranou e_A. Všetky hrany, ktoré pôvodne vchádzali do vrcholu A budú smerovať do A_{start} a všetky vychádzajúce hrany budú smerovať z A_{end}.

Do grafu závislostí ešte pridáme koncovú aktivitu A_{final}. A_{final} reprezentuje konečný termín (Deň D) a je závislá od všetkých aktivít harmonogramu (obrázok 1 c). Hrany smerujúce do A_{final} môžeme zároveň použiť pre nastavenie ceny posunutia aktivít, pretože tieto hrany sa v reze nachádzajú práve vtedy, keď sa posúvajú aktivity z ktorých tieto hrany vychádzajú.

Pseudokód algoritmu pre transformáciu na graf závislostí je nasledujúci:

```
vytvor prázdny graf závislostí G = ∅
pre každú aktivitu
("A", D_used, D_produced, t_start, t_end, t_min)
    pridaj vrcholy A_start, A_end do G
    pridaj hranu e_A = (A_start, A_end) do G
pre každú aktivitu
("A", D_A_used, D_A_produced, t_A_start, t_A_end, t_A_min)
    pre každú aktivitu
    ("B", D_B_used, D_B_produced, t_B_start, t_B_end, t_B_min)
        ak D_A_produced ∩ D_B_used ≠ ∅
            pridaj hranu e_AB = (A_end, B_start) do G
pridaj vrchol A_final pre koncovú aktivitu
pre každú aktivitu
("A", D_used, D_produced, t_start, t_end, t_min)
    pridaj hranu e_Afin = (A_end, A_final) do G
```

```
pre každú hranu e_A (reprezentujúcu aktivitu A)
    ak aktivita nemôže byť skrátená
    (t_A_end − t_A_start = t_A_min)
        throughput_e_A = ∞
    ak aktivita môže byť skrátená
    (t_A_end − t_A_start > t_A_min)
        throughput_e_A = 1
pre každú hranu e_AB (reprezentujúcu závislosť B
na A)
    ak závislosť nemôže byť skrátená
    (t_B_start − t_A_end = 0)
        throughput_e_AB = ∞
    ak závislosť môže byť skrátená
    (t_B_start − t_A_end > 0)
        ak B = A_final
            throughput_e_AB = 0.001
        inak
            throughput_e_AB = 0
```

3.2 Cena hrán

Každý rez grafu závislostí, ktorý oddeľuje aktivitu ktorá porušila harmonogram od koncovej aktivity, reprezentuje jednu možnú zmenu harmonogramu. Chceme nájsť zmenu, ktorá by skrátila a posunula aktivity za minimálnu cenu. Musíme preto hranám priradiť ceny, ktoré zodpovedajú skráteniu týchto hrán (a zodpovedajúcich aktivít a závislostí). Ceny potom budú predstavovať maximálnu priepustnosť hrán.

Ceny môžeme zadefinovať rôznymi spôsobmi. Najjednoduchší spôsob je zadefinovať cenu skrátenia aktivity ako 1 a cenu skrátenia závislosti ako 0. Takýmto spôsobom hovoríme algoritmu, že chceme skrátiť čo najmenej aktivít a nevadí nám skracovanie závislostí (skrátenie závislosti znamená skrátenie prestávky medzi dvomi aktivitami). Ak chceme obmedziť aj posúvanie aktivít, ale zároveň tým neovplyvniť skracovanie aktivít, môžeme nastaviť cenu za posunutie aktivity na dostatočne malé číslo, povedzme 0,001. Cena za posunutie aktivity A je cenou hrany $e_{Afin} = (A_{end}, A_{final})$, ktorá reprezentuje závislosť koncovej aktivity na ukončení aktivity A.

Je možný aj iný spôsob oceňovania. Aktivity môžu mať zadefinované rôzne ceny alebo môžeme uprednostňovať skrátenie aktivít s väčšou časovou rezervou.

Musíme tiež zabezpečiť, aby sa aktivita alebo závislosť neskrátila pod svoju minimálnu dobu trvania. Ak má aktivita minimálnu dĺžku, nemôže byť ďalej skracovaná a cena jej skrátenia je preto nekonečná. Podobne, ak má závislosť nulovú dĺžku (závislá aktivita začína okamžite po skončení aktivity na ktorej je závislá), nemôže byť skrátená a jej cena skrátenia je nekonečná.

V systéme RAPROT je použitý nasledujúci spôsob prideľovania ceny/priepustnosti:

3.3 Minimálny rez

Na nájdenie minimálneho rezu používame Ford-Fulkersonov algoritmus pre hľadanie maximálneho toku [3]. Pre tento účel však môže byť použitý ľubovoľný algoritmus ktorý hľadá minimálny rez grafu. Jedinou požiadavkou je schopnosť pracovať s nekonečnou priepustnosťou hrán.

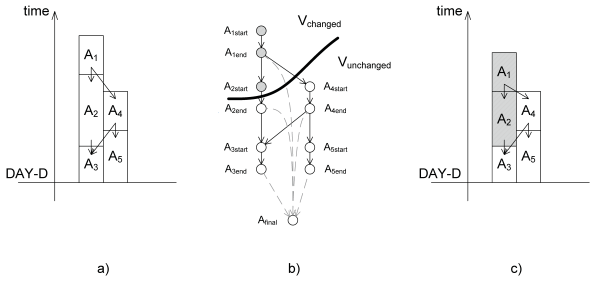
Algoritmus rozdeľuje graf závislostí na dve časti tým, že pridelí vrcholom v rôznych častiach grafu rôzne značky. Zároveň tiež pre graf závislostí vypočíta maximálny tok, ktorý je rovný kapacite minimálneho rezu. Kapacita minimálneho rezu je zároveň cenou zmeny harmonogramu vykonanej podľa tohto rezu.

Konverzia harmonogramu na graf a späť je veľmi priamočiara (s časovou zložitosťou $O(|A|+|D|)$). Najväčší dopad na časovú zložitosť má samotný použitý algoritmus pre nájdenie minimálneho rezu. Časová zložitosť Ford-Fulkersonovho algoritmu je $O((|A| + |D|)^2 * |A|)$, kde $|A|$ je počet aktivít a $|D|$ je počet závislostí. Toto je zároveň zložitosť celého algoritmu na zmenu harmonogramu. (V literatúre je možné nájsť algoritmus pre nájdenie minimálneho rezu s časovou zložitosťou $O(|A|^3)$ [8].)

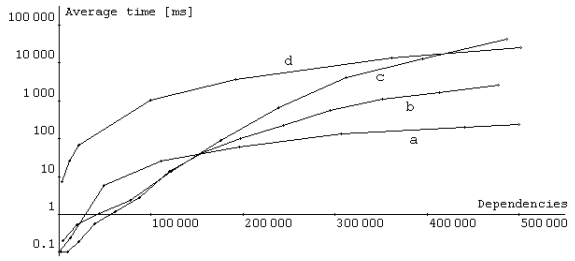
3.4 Zmena harmonogramu

Nájdením minimálneho rezu (rezu s minimálnou kapacitou) grafu závislostí zároveň nájdeme zmenu harmonogramu s najnižšou cenou. Cena zmeny harmonogramu je rovná kapacite minimálneho rezu.

Algoritmus pre minimálny rez rozdelil vrcholy grafu do dvoch množín: $V_{changed}$ a $V_{unchanged}$ (obrázok 2 b). Vrchol prislúchajúci aktivite ktorá nedodržala harmonogram patrí do prvej množiny ($A_{fail} \in V_{changed}$), koncový vrchol patrí do druhej množiny ($A_{final} \in V_{unchanged}$).



Obrázok 2. Príklad minimálneho rezu pre zmenu harmonogramu. Z pôvodného harmonogramu (a) sa vytvorí graf závislostí a podľa jeho minimálneho rezu (b) sa aktivita A_1 presunie a aktivita A_2 skráti (c).



Obrázok 3. Závislosť časovej zložitosti (v logaritmickej mierke) na počte závislostí pre náhodný graf s pravdepodobnosťou pridania závislosti medzi aktivitami 0.1 (a), 0.5 (b) a 1.0 (c) a graf s bariérovou synchronizáciou (d).

Každý vrchol grafu predstavuje začiatok alebo koniec nejakej aktivity (kapitola 3.1). Vrcholy patriace do $V_{changed}$ budú posunuté dopredu v čase o 1 deň. Vrcholy patriace do $V_{unchanged}$ ostanú nezmenené.

Hrany vnútri $V_{changed}$ alebo $V_{unchanged}$ predstavujú aktivity alebo závislosti, ktorých dĺžka ostane nezmenená, pretože sa posunie buď aj ich počiatočný aj koncový vrchol alebo ani jeden z nich. Aktivity a závislosti reprezentované hranami prechádzajúcimi z $V_{changed}$ do $V_{unchanged}$ sa skrátia o 1 deň, pretože sa posunú len ich počiatočné vrcholy, zatiaľ čo ich koncové vrcholy sa nezmenia (obrázok 2 c). Nový harmonogram potom ešte podlieha schváleniu zodpovedného pracovníka. Pracovníci zodpovední za presunuté alebo skrátené aktivity sú následne informovaní.

Ak je kapacita minimálneho rezu (maximálny prietok grafu, resp. minimálna cena skrátenia harmonogramu) nekonečná, potom nie je možné zmeniť harmonogram bez porušenia obmedzení vyplývajúcich zo závislostí medzi aktivitami alebo minimálneho trvania aktivít. V takomto prípade systém informuje zodpovednú osobu o kritickom stave, kedy nie je možné ukončiť tok práce v požadovanom čase.

Samotná zmena harmonogramu sa vykoná podľa nasledujúceho pseudokódu:

```
ak maximálny tok = ∞
   skonči s chybou
inak
   pre každý vrchol A_start ∈ V_changed
      zväčši t_{A_start} o 1
   pre každý vrchol A_end ∈ V_changed
      zväčši t_{A_end} o 1
```

4 Výsledky časovej zložitosti

Získať štatistické dáta o tokoch práce je zložité, keďže je dostupných len málo definícií tokov práce. Testovali sme preto algoritmus na dvoch typoch grafov: náhodných a bariérových.

Použili sme náhodne generované grafy, s aktivitami náhodnej dĺžky začínajúcimi v náhodných časoch. Závislosti medzi aktivitami sú pridávané s určitou pravdepodobnosťou. V takomto grafe sme potom náhodne vybrali jednu aktivitu, zvýšili čas jej vykonávania a vykonali pre ňu optimálne posunutie harmonogramu. Náhodný graf poskytuje dobrú aproximáciu reálneho toku práce, keďže reálne toky práce zvyčajne nemajú jednotnú štruktúru. Analýza reálnych tokov práce by samozrejme poskytla lepší model.

Aby sme mohli analyzovať vplyv paralelných aktivít na výpočtovú zložitosť, použili sme aj grafy s paralelnými aktivitami synchronizovanými bariérovou synchronizáciou. V každom čase sa paralelne vykonáva sada aktivít. Ďalšia sada sa začne vykonávať až po tom, ako sa skončilo vykonávanie predchádzajúcej sady (aktivity v každej sade závisia na všetkých aktivitách predchádzajúcej sady). Týmto simulujeme tok práce s niekoľkými synchronizačnými bodmi.

Časy vykonávania pre rôzne veľkosti grafu a pravdepodobnosti pridania hrany sú na obrázku 3. V menej prepojených grafoch rastie časová zložitosť pomalšie. Rovnako aj variancia časov (na grafe nie je zobrazená) je pre menej prepojené grafy podstatne nižšia. Je to spôsobené tým, že v toku práce s menším počtom obmedzení je riešením zvyčajne skrátenie alebo posunutie jednej alebo niekoľkých aktivít, zatiaľ čo v toku práce v ktorom sú aktivity plne prepojené závislosťami môže byť riešenie veľmi jednoduché (skrátenie jednej aktivity), ale aj veľmi zložité (napríklad presunutie polovice aktivít), čo vyžaduje vykonanie viac krokov Ford-Fulkersonovho algoritmu.

Merania a numerické vyhodnotenie potvrdili predpovedanú kubickú výpočtovú zložitosť (kapitola 3.3.), ktorá závisí hlavne na počte závislostí (počet závislostí väčšinou rastie oveľa rýchlejšie než počet aktivít). Prístup sa dobre vyrovnáva s veľkými problémami.

5 Ďalšia práca

Bude vhodné spojiť navrhnutú metódu s existujúcimi metódami na tvorbu harmonogramu, predovšetkým s metódami založenými na CPM a PERT [10]. Pri semiautomatickej zmene harmonogramu tieto metódy poskytnú používateľovi dodatočné informácie o harmonograme, ako napríklad zoznam kritických aktivít a celková časová rezerva pri optimistickom/pesimistickom scenári vykonávania. Pri automatickej zmene harmonogramu je možné použiť tieto dodatočné informácie na zmenu cenovej funkcie, napríklad zvýšením ceny skrátenia aktivít ktoré sa môžu stať kritickými.

Výzvou do budúcnosti je pridanie obmedzení vyplývajúcich z využívania zdrojov. Toto by sa dalo dosiahnuť napríklad pridaním zdrojových závislostí, podobne ako v algoritme pre tvorbu harmonogramov RCPM [7]. Zdrojové závislosti by tvorili ďalšie závislosti v grafe závislostí.

6 Zhodnotenie

Predstavený algoritmus určený pre dynamickú zmenu harmonogramu dokáže ošetriť porušenie harmonogramu tým, že presunie a skráti závislé aktivity. Každá zmena harmonogramu má svoju cenu. Táto cena je výsledkom nepredvídanej udalosti, ktorá túto zmenu spôsobila. Predstavený algoritmus používa informáciu o tejto cene na to, aby našiel najlacnejšiu zmenu harmonogramu v polynomickom (kubickom) čase.

Predstavený algoritmus pre zmenu harmonogramu je vhodný pre toky práce, ktoré spĺňajú nasledujúce podmienky:

1. Tok práce je dobre zadefinovaný a naplánovaný. Poznáme dokumenty používané a vytvárané každou aktivitou a závislosti medzi aktivitami. Všetky aktivity majú naplánovaný čas vykonávania.
2. Aktivity môžu byť skrátené, ale iba po minimálnu, kritickú, dĺžku. Aktivita má nominálnu dĺžku, ktorá môže byť v prípadne nutnosti ďalej skracovaná. Každá aktivita má však zadefinovaný kritický čas, ktorý je bezpodmienečne nutný pre jej úspešné ukončenie.
3. Musí byť splnený koncový termín. Termíny pre jednotlivé aktivity môžu byť zmenené, ale všetky aktivity musia skončiť pred konečným termínom (Deň D). Posunutiu termínov za Deň D je potrebné za každú cenu zabrániť.
4. Účastníci toku práce (agenti) sú ochotní prijať zmenený harmonogram za predpokladu, že neporušuje podmienky 2 a 3.

Na agentov zúčastňujúcich sa toku práce nie sú kladené žiadne ďalšie obmedzenia. Systém RAPORT, pre ktorý bol algoritmus primárne navrhnutý, sa týka len ľudských agentov. Algoritmus samotný však môže byť rovnako úspešne použitý aj pre umelých agentov alebo hybridné systémy.

Referencie

1. Bonnal P., De Jonghe J., and Ferguson J., A Deliverable-Oriented Evm System Suited to a Large-Scale Project. Project Management Journal, 2006
2. Corkill D.D., Rubinstein Z.B., Lander S.E., and Lesser V.R., Live-Representation Process Managment. Proc. 5th International Conference on Enterprise Information Systems, Angers, France, 2003
3. Ford L.R. and Fulkerson D.R., Maximal Flow Through a Network. Canadian Journal of Mathematics, 8, 1956
4. Fischer L., The Workflow Handbook 2001. Workflow Management Coalition (WfMC), 2001
5. Forgac R., Budinska I., Gatial E., Nguyen G., Laclavik M., Balogh Z., Mokris I., Hluchy L., Ciglan M., and Babik M., Ontology Based Knowledge Management for Organizational Learning. Proc. of 9-th Intl. Conf. ISIM'06 Information Systems Implementation and Modelling, Brno, April, MARQ Ostrava, 2006
6. Fox M.S., Constraint-Directed Search: A Case Study of Job-Shop Scheduling. Morgan Kaufmann Publishers, Inc., 1987
7. Kim K.: A Resource-Constrained CPM (RCPM) Scheduling and Control Technique with Multiple Calendars. Dissertation, Faculty of Virginia Polytechnic Institute and State University, USA, 2003
8. Kučera L., Kombinatorické algoritmy. Praha, SNTL, 1993
9. Lin X., Janak S.L., Floudas C., A New Robust Optimization Approach for Scheduling under Uncertainty: I. Bounded Uncertainty. Computers and Chemical Engineering, 28, 2004, 1069–1085
10. Moder J.J., Phillips C.R, and Davis E.W., Project Management with CPM, PERT, and Precedence Diagramming, 3rd Edition. Van Nostrand Reinhold Company, New York, NY., 1983
11. Research and Development of a Knowledge Based System to Support Workflow Management in Organizations with Administrative Processes - RAPORT (APVT-51-024604) http://raport.ui.sav.sk/
12. Sadeh N., Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling, Ph.D. Thesis. School of Computer Science, Carnegie Mellon University, 1991
13. Vin J.P. and Ierapetritou M.G., A New Approach for Efficient Rescheduling of Multiproduct Batch Plants. Industrial Engineering and Chemical Research, 39, 2000, 4228–4238
14. Williams T., Workflow Management within the ARIS Framework. http://www.pera.net/Methodologies/ARIS/ARIS_Paper_by_Ted_Williams.html

Non-monotonic reasoning with various kinds of preferences in the relational data model framework*

Radim Nedbal

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
radned@seznam.cz

Abstract. *The paper gives an overview of recent advances in the field of logic of preference and discusses their applicability in the frame of the relational data model. Namely, non-monotonic reasoning mechanisms with various kinds of preferences are reviewed in detail, and a way of suiting them to practical database applications is presented. These mechanisms enable to reason simultaneously about sixteen strict and non-strict kinds of preferences, including ceteris paribus preferences. To make the mechanisms useful for practical applications, the assumption of preference specification consistency has to be loosened. This is achieved in two steps: firstly, all the preference specifications are generalized to permit uncertainty, and secondly, not a total pre-order on worlds but a partial pre-order on worlds is used in the semantics, which enables to indicate some kind of conflict among worlds by their incomparability. Most importantly, the semantics of set of preferences is related to that of a disjunctive logic program.*

1 Introduction

All to often no reasonable answer is returned by an SQL-based search engine though one has tried hard writing query to match one's personal preferences closely. The case of repeatedly receiving *empty query result* is extremely disappointing to the user. On the other hand, leaving out some conditions in the query often leads to another unpleasant extreme: an *overloading* with lots of mostly *irrelevant information*.

This observation stems from the fact that traditional database query languages treat all the requirements on the data as mandatory, hard ones. However, it is natural to express queries in terms of both hard as well as soft requirements, i.e., preferences, in many applications. In the "real world", preferences are understood in the sense of wishes: in case they are not satisfied, database users are usually prepared to accept worse alternatives. Thus preferences require a paradigm shift from exact matches towards a best possible matchmaking.

The paper presents a work in progress aiming at **simultaneous usage of various preferences (including set preferences) with general, preference logic based semantics in the context of database queries**. The objective is to provide database users with a language that is declarative, can be used to define such database queries that not necessarily all answers but rather the best, the most preferred ones are returned, includes various kinds of preferences, and has an intuitive, well defined semantics allowing for conflicting preferences.

In section 2, the basic concepts of logic of preference and non-monotonic logic of preference are briefly summarized. In section 3, basic concepts and key features of the proposed approach are presented; section 4 gives a brief overview of related work, and the 5th section concludes the paper.

2 Preliminaries

The logic of preference has been studied since the sixties as a branch of philosophical logic: Logicians and philosophers have been attempting to define the one well-formed logic that people should follow when expressing preferences.

2.1 Logic of preference

It is Von Wright's essay [10] that tries to give the first axiomatization of a logic of preference. The general idea is that the expression "a is preferred to b" should be understood as the preference of a state (a world) where a occurs over a state where b occurs. Von Wright expressed a theory based on five axioms. The problem is that empirical observation of human behavior provides counterexamples of this axiomatization.

Later, Von Wright [11] introduced a more general frame to define preferences, updating also the notion of ceteris paribus preferences. In this approach, he considers a set S of n logically independent states

* This work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization" and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

of affairs and the set $W = 2^S$ of 2^n combinations of the elements of S. An s-world is called any element of W that holds when s holds. In the same way is defined a C_i-world, where C_i is a combination of elements of S. Now, von Wright gives two definitions (strong and weak) of "s is preferred to t *under the circumstances* C_i":

1. (strong): s is preferred to t under the circumstances C_i iff every C_i-world that is also an s-world and not a t-world is preferred to every C_i-world that is also a t-world and not an s-world.
2. (weak): s is preferred to t under the circumstances C_i iff some C_i-world that is also an s-world is preferred to some C_i-world that is also a t-world, and no C_i-world which is a t-world is preferred to any C_i-world which is an s-world.

Finally, if s is preferred to t under all circumstances C_i, according to either definition, then s is said to be preferred to t ceteris paribus.

It can be concluded that the philosophical discussion about preferences failed the objective to give a unifying frame of generalized preference relations that could hold for any kind of states, based on well-defined axiomatization.

More recently, Von Wright's ideas and the discussion about "logical representation of preferences" attracted attention again. For instance Doyle and Wellman [4] give a modern treatment of preferences ceteris paribus.

2.2 Logic of preferences

A drawback of the present state of the art in the logic of preference is that proposed logics typically formalize only preference of one kind. Consequently, when formalizing preferences, one has to choose which kind of preference statements are used for all preferences under consideration.

To study the interaction among kinds of preferences, a non-monotonic preference logic for various kinds of preferences, *logic of preferences* – in contrast to the usual reference to the *logic of preference*, has been recently developed by Kaci and Torre [5]. They have developed algorithms for a non-monotonic preference logic for sixteen kinds of preferences: four basic types, each of them strict or non-strict, with or without ceteris paribus proviso.

To describe ceteris paribus preference, a general construction proposed by Doyle and Wellman [4] is employed. Their language for preference built over a set of propositions is defined inductively from propositional variables. They mean by *proposition* a set of individual objects, elements of a set W. These individual objects can be understood as worlds, i.e., truth assignments for propositional variables. In other words, a propositional formula is identified with worlds – fulfilling truth assignments, and the powerset 2^W is taken to be the set of all propositional formulas.

Their ceteris paribus preferences are based on a notion of contextual equivalence:

Definition 1. ***(Contextual equivalence)*** *[4, Def.4] Let W be a set of worlds and $\xi(W)$ be the set of equivalence relations on W. A contextual equivalence on W is a function $\eta : 2^{2^W} \to \xi(W)$ assigning to each set of propositional formulas $\{\varphi, \psi, \ldots\}$ equivalence relation $\eta(\varphi, \psi, \ldots)$.*

If $w\ \eta(\varphi, \psi, \ldots)\ w'$, we usually write

$$w \equiv w' mod_\eta(\varphi, \psi, \ldots) \ .$$

Definition 2. ***(Preference model)*** *A preference model $\mathcal{M} = \langle W, \succeq, \eta \rangle$ is a triplet in which W is a set of worlds, $\succeq$ is a total pre-order, i.e., a relation which is complete, reflexive, and transitive, over W, and η is a contextual equivalence function on W.*

Definition 3. ***(Comparative greatness)*** *[4, Def.5] We say that "φ is weekly greater than ψ," written $\varphi \trianglerighteq \psi$, is satisfied in the model $\mathcal{M}$, written $\mathcal{M} \models \varphi \trianglerighteq \psi$, iff $w_1 \succeq w_2$ whenever*

1. $w_1 \models \varphi \wedge \neg\psi$,
2. $w_2 \models \neg\varphi \wedge \psi$, *and*
3. $w_1 \equiv w_2 mod_\eta(\varphi \wedge \neg\psi, \neg\varphi \wedge \psi)$.

This definition of ceteris paribus preferences seems very close to the intended semantics behind von Wright's principles. Preferences of φ over ψ are defined as preferences of $\varphi \wedge \neg\psi$ over $\neg\varphi \wedge \psi$, which is standard and known as von Wright's expansion principle [10]. Also, note that if the equivalence relation $\eta(\varphi \wedge \neg\psi, \neg\varphi \wedge \psi)$ is the universal relation, i.e., an equivalence relation with only one equivalence class, then the ceteris paribus preference reduces to strong condition (φ is preferred to ψ when each $\varphi \wedge \neg\psi$ is preferred to all $\neg\varphi \wedge \psi$).

The following proposition [1] shows that Def.3 reduces a preference with ceteris paribus proviso to a set of preferences for each equivalence class of the equivalence relation.

Proposition 1. *[5, Prop.2] Assume a finite set of propositional variables, and let $\epsilon(\eta, \varphi, \psi)$ be the set of propositional formulas which are true in all worlds of an equivalence class of $\eta(\varphi, \psi)$, but false in all others: $\{\chi | \exists w \forall w' : w \equiv w' mod_\eta(\varphi, \psi) \iff w' \models \chi\}$. We have that "$\varphi$ is weakly greater than ψ" is satisfied in the model $\mathcal{M} = \langle W, \succeq, \eta \rangle$ iff for all propositions $c \in \epsilon(\eta, \varphi \wedge \neg\psi, \neg\varphi \wedge \psi)$, we have that $w_1 \succeq w_2$ whenever*

1. $w_1 \models \varphi \wedge \neg\psi \wedge c$,
2. $w_2 \models \neg\varphi \wedge \psi \wedge c$.

The logical language introduced in [5] extends propositional logic with sixteen kinds of preferences:

Definition 4. ***(Language)*** *[5, Def.3] Given a finite set of propositional variables $p, q, \ldots$, the set L_0 of propositional formulas and the set L of preference formulas is defined as follows:*

$$L_0 \ni \varphi, \psi\text{: } p|(\varphi \wedge \psi)|\neg\varphi$$
$$L \ni \Phi, \Psi\text{: } \varphi\ {}^{x}{>}^{y}\ \psi|\varphi\ {}^{x}{\geq}^{y}\ \psi|\varphi\ {}^{x}{>}_{c}^{y}\ \psi|\varphi\ {}^{x}{\geq}_{c}^{y}\ \psi| \neg\Phi|(\Phi \wedge \Psi) \quad \text{for } x, y \in \{m, M\}$$

Definition 5. ***(Monotonic semantics)*** *[5, Def.4] Let $\mathcal{M}$ be a preference model. When $x = M$ we write $x(\varphi, \mathcal{M})$ for*

$$\max(\varphi, \mathcal{M}) = \{w \in W | w \models \varphi \wedge \forall w' \in W : w' \models \varphi \Rightarrow w \succeq w'\} \ ,$$

and analogously when $x = m$ we write $x(\varphi, \mathcal{M})$ for

$$\min(\varphi, \mathcal{M}) = \{w \in W | w \models \varphi \wedge \forall w' \in W : w' \models \varphi \Rightarrow w' \succeq w\} \ .$$

$$\mathcal{M} \models \varphi\ {}^{x}{>}^{y}\ \psi \text{ iff } \forall w \in x(\varphi \wedge \neg\psi, \mathcal{M}), \forall w' \in y(\neg\varphi \wedge \psi, \mathcal{M}) : w \succ w'$$
$$\mathcal{M} \models \varphi\ {}^{x}{\geq}^{y}\ \psi \text{ iff } \forall w \in x(\varphi \wedge \neg\psi, \mathcal{M}), \forall w' \in y(\neg\varphi \wedge \psi, \mathcal{M}) : w \succeq w'$$
$$\mathcal{M} \models \varphi\ {}^{x}{>}_{c}^{y}\ \psi \text{ iff } \forall c \in \epsilon(\eta, \varphi \wedge \neg\psi, \neg\varphi \wedge \psi), \forall w \in x(\varphi \wedge \neg\psi \wedge c, \mathcal{M}), \forall w' \in y(\neg\varphi \wedge \psi \wedge c, \mathcal{M}) : w \succ w'$$
$$\mathcal{M} \models \varphi\ {}^{x}{\geq}_{c}^{y}\ \psi \text{ iff } \forall c \in \epsilon(\eta, \varphi \wedge \neg\psi, \neg\varphi \wedge \psi), \forall w \in x(\varphi \wedge \neg\psi \wedge c, \mathcal{M}), \forall w' \in y(\neg\varphi \wedge \psi \wedge c, \mathcal{M}) : w \succeq w'$$

Moreover, logical notions are defined as usual:

$$S \models \Phi \iff \forall \mathcal{M} : \mathcal{M} \models S \Rightarrow \mathcal{M} \models \Phi \ .$$

Note that $\varphi\ {}^{m}{\geq}_{c}^{M}\ \psi$ is the Doyle and Wellmans's comparative greatness (Def.3).

In this paper, we are interested in a special kind of theories, namely preference specifications:

Definition 6. ***(Preference specification)*** *[5, Def.5] Let $\mathcal{P}_{\rhd}$ be a set of preferences of the form $\{\varphi_i \rhd \psi_i : i = 1, \ldots, n\}$. A preference specification $\mathcal{P}$ is a tuple $\langle \mathcal{P}_{\rhd} | \rhd \in \{\ {}^{x}{>}^{y},\ {}^{x}{\geq}^{y},\ {}^{x}{>}_{c}^{y},\ {}^{x}{\geq}_{c}^{y}\ | x, y \in \{m, M\}\}\rangle$, and $\mathcal{M}$ is its model iff it models all $\mathcal{P}_{\rhd}$:*

$$\mathcal{M} \models \mathcal{P}_{\rhd} \iff \forall(\varphi_i \rhd \psi_i) \in \mathcal{P}_{\rhd} : \mathcal{M} \models \varphi_i \rhd \psi_i \ .$$

Corollary 1. *Observe that by Prop.1, we can replace ceteris paribus preferences, written ${}^{x}{>}_{c}^{y}$ or ${}^{x}{\geq}_{c}^{y}$, by sets of ordinary preferences without a ceteris paribus proviso. Consequently, we can restrict ourselves to the eight types of preferences without ceteris paribus clauses.*

2.3 Non-monotonic logic of preferences

Non-monotonic reasoning has been characterized by Shoham [9] as a mechanism that selects a subset of the models of a set of formulas, which we call distinguished models. Thus non-monotonic consequences of a logical theory are defined as all formulas which are true in the distinguished models of the theory.

An attractive property occurs when there is only one distinguished model, as then all non-monotonic consequences can be found by calculating the unique distinguished model and characterizing all formulas satisfied by this model. It has been proved in the literature that a unique distinguished model can be defined for the following sets of preferences: $\mathcal{P}_{\ {}_{m>M}}$, $\mathcal{P}_{\ {}_{m>m}}$, and $\mathcal{P}_{\ {}_{M>M}}$.

Moreover, Kaci and Torre [5] have defined a distinguished model and proved ist uniqueness for

$$\langle \mathcal{P}_{\rhd} | \rhd \in \{\ {}^{x}{>}^{y},\ {}^{x}{\geq}^{y},\ {}^{x}{>}_{c}^{y},\ {}^{x}{\geq}_{c}^{y}\ | x \in \{m, M\}, y{=}M\}\rangle$$

and also for

$$\langle \mathcal{P}_{\rhd} | \rhd \in \{\ {}^{x}{>}^{y},\ {}^{x}{\geq}^{y},\ {}^{x}{>}_{c}^{y},\ {}^{x}{\geq}_{c}^{y}\ | x = m, y \in \{m, M\}\}\rangle$$

They have also provided algorithms to calculate these two unique models and presented a way to combine these models to find a distinguished model of all the types of preferences given together. Their algorithms also capture all the algorithms for handling all the kinds of preferences separately.

It should be pointed out, that the consistency of preference specification, i.e., existence of its preference model, has been assumed by now. This assumption, however, is hard to fulfil in practical applications. In order not to restrict the use of the logic of preference, Boella and Torre [1] have proposed a minimal logic of preference in which *any* preference specification is consistent. They achieve the consistency by means of:

- formalizing a preference φ over ψ as the absence of a ψ world that is preferred over a φ worlds;
- amending the preference model definition by using partial pre-order instead of total pre-order on worlds, which enables to indicate some kind of conflict among worlds (by their incomparability).

Their non-monotonic reasoning is based on distinguished models called *most connected models.*

Definition 7. ***Most connected model*** *[1, Def.4] A model $\mathcal{M} = \langle W, \succeq, \eta\rangle$ is at least as connected as another model $\mathcal{M}' = \langle W, \succeq', \eta\rangle$, written as $\mathcal{M} \sqsubseteq \mathcal{M}'$, if $\succeq' \subseteq \succeq$, i.e.,*

$$\forall w_1, w_2 \in W : w_1 \succeq' w_2 \Rightarrow w_1 \succeq w_2 \ .$$

A model $\mathcal{M}$ is most connected if there is no other model $\mathcal{M}'$ s.t. $\mathcal{M}' \sqsubset \mathcal{M}$, i.e., s.t. $\mathcal{M}' \sqsubseteq \mathcal{M}$ without $\mathcal{M} \sqsubseteq \mathcal{M}'$.

In comparison with Kaci and Torre's language of logic of preferences, their language is by far less expressive, having only one kind of preference.

3 Preferences in database queries

To improve the readability, $x \succeq y \wedge \neg(y \succeq x)$, $\succeq (x,y) \wedge \neg \succeq (y,x)$, and $\succeq (x,y) \wedge \succeq (y,x)$ is substituted by $x \succ y$, $\succ (x,y)$, and $= (x,y)$, resp., henceforth.

3.1 Basic concepts and key features

To reach the target, we need to accommodate an expressive language with various kinds of preferences in the RDM framework so that any set of (possibly conflicting) preferences has a well defined semantics. We propose to base its model-theoretic semantics on those of preference logic languages.

In the following list of basic concepts of our approach, the key features are boldfaced.

- User preferences are expressed in a **preference logic language**.
- Semantics of a set of (possibly conflicting) preferences is related to that of a **disjunctive logic program** (DLP).
- **Non-monotonic reasoning mechanisms** about preferences has to be employed to reason about preferences that are defined in such a way that consistency is ensured under all circumstances.
- A *preference operator* (PO) returning only the best tuples in the sense of user preferences is used to embed preferences into relational query languages (RQL).

We identify propositional variables with tuples, i.e., facts over relations. A subset of a relation instance, i.e., a set of facts, creates a world, an element of a set W, and propositions are logically implied by worlds in which they hold true.

3.2 User preferences

Our starting point is the language (Def. 4) introduced by Kaci and Tore [5] who extend propositional language with sixteen kinds of preferences.

To define the semantics without the consistency assumption, the definition (Def.2) of the preference model has to be extended. It, however, is not necessary to extend it as much as Boella and Torre [1] have done, who have replaced total pre-order with partial pre-order on worlds in the preference model definition. By contrast, it shows that *partial pre-order*, i.e., a binary relation which is reflexive and transitive, provides a sufficient space of models.

Definition 8. ***(Preference model)*** *A preference model $\mathcal{M}(R) = \langle W, \succeq \rangle$ over a relation schema R is a couple in which W is a set of worlds, relation instances of R, and $\succeq$ is a partial pre-order over W, the preference relation.*

Observe that as preferences with ceteris paribus provisos can be reduced in accordance with Cor.1 to sets of preferences without such provisos, we have neglected the contextual equivalence.

Definition 9. ***(Models of preferences)*** *Let $\mathcal{M}$ be a preference model and w, w' elements of W s.t. $w \models \neg\varphi \wedge \psi$ and $w' \models \varphi \wedge \neg\psi$. Then:*

$\mathcal{M} \models \varphi \ {}^{M}\!>^{M} \psi$ *iff* $\exists w'$ *s.t.*[1] $\forall w$: *if* $\varphi \wedge \neg\psi \not\models_W$ *false, we have* $\neg(w \succeq w')$.
$\mathcal{M} \models \varphi \ {}^{M}\!\geq^{M} \psi$ *iff* $\exists w'$ *s.t.* $\forall w$: *if* $\varphi \wedge \neg\psi \not\models_W$ *false, we have* $\neg(w \succ w')$.
$\mathcal{M} \models \varphi \ {}^{m}\!>^{M} \psi$ *iff* $\forall w \forall w'$, *we have* $\neg(w \succeq w')$.
$\mathcal{M} \models \varphi \ {}^{m}\!\geq^{M} \psi$ *iff* $\forall w \forall w'$, *we have* $\neg(w \succ w')$.
$\mathcal{M} \models \varphi \ {}^{M}\!>^{m} \psi$ *iff* $\exists w \exists w'$: *if* $\neg\varphi \wedge \psi \not\models_W$ *false and* $\varphi \wedge \neg\psi \not\models_W$ *false, we have* $\neg(w \succeq w')$.
$\mathcal{M} \models \varphi \ {}^{M}\!\geq^{m} \psi$ *iff* $\exists w \exists w'$: *if* $\neg\varphi \wedge \psi \not\models_W$ *false and* $\varphi \wedge \neg\psi \not\models_W$ *false, we have* $\neg(w \succ w')$.
$\mathcal{M} \models \varphi \ {}^{m}\!>^{m} \psi$ *iff* $\exists w \forall w'$: *if* $\neg\varphi \wedge \psi \not\models_W$ *false, we have* $\neg(w \succeq w')$.
$\mathcal{M} \models \varphi \ {}^{m}\!\geq^{m} \psi$ *iff* $\exists w \forall w'$: *if* $\neg\varphi \wedge \psi \not\models_W$ *false, we have* $\neg(w \succ w')$.

3.3 Preference specification semantics

Definition 10. ***(Preference specification)*** *Let R be a relation schema. Given the set $L_0(R)$ from the definition (Def.4) of the language in which propositional variables are identified with facts over the relation R, $\mathcal{P}_\rhd(R)$ is a set of preferences over the relation schema R of the form $\{\varphi_i \rhd \psi_i : i = 1, \ldots, n\}$ for $\varphi_i, \psi_i \in L_0(R)$. A preference specification over the relation schema R is a tuple $\langle \mathcal{P}_\rhd(R) | \rhd \in \{ {}^{x}\!>^{y}, {}^{x}\!\geq^{y} | x, y \in \{m, M\}\}\rangle$, and $\mathcal{M}(R)$ is its model, i.e., a preference specification model, iff it models all $\mathcal{P}_\rhd(R)$:*

$$\mathcal{M}(R) \models \mathcal{P}_\rhd(R) \iff \forall(\varphi_i \rhd \psi_i) \in \mathcal{P}_\rhd(R) : \mathcal{M}(R) \models \varphi_i \rhd \psi_i$$

To calculate a preference specification model, we associate the preference specification model $\mathcal{P}$ with a DLP in three steps:

First step: Create a partition $(E_1, \ldots, E_n)$ of W so that $w, w' \in E_i$ iff any of the following conditions is fulfilled for every preference $\varphi \rhd \psi$:

[1] $\varphi \wedge \neg\psi \not\models_W$ *false* denotes that there is a model in W for $\varphi \wedge \neg\psi$.

1. $w \models \varphi \wedge \neg\psi$ and $w' \models \varphi \wedge \neg\psi$,
2. $w \models \neg\varphi \wedge \psi$ and $w' \models \neg\varphi \wedge \psi$,
3. $w \models (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$ and $w' \models (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$.

Second step: Substitute each preference type by a logical formula[2]:

$\varphi \ {}^{M}{>}^{M} \ \psi$**:** $\exists E_i \forall E_j : \not\succeq (E_j, E_i)$ if $\varphi \wedge \neg\psi \not\models_W$ *false*.
$\varphi \ {}^{M}{\geq}^{M} \ \psi$**:** $\exists E_i \forall E_j : \not\succ (E_j, E_i)$ if $\varphi \wedge \neg\psi \not\models_W$ *false*.
$\varphi \ {}^{m}{>}^{M} \ \psi$**:** $\forall E_j \forall E_i : \not\succeq (E_j, E_i)$.
$\varphi \ {}^{m}{\geq}^{M} \ \psi$**:** $\forall E_j \forall E_i : \not\succ (E_j, E_i)$.
$\varphi \ {}^{M}{>}^{m} \ \psi$**:** $\exists E_j \exists E_i : \not\succeq (E_j, E_i)$ if $\varphi \wedge \neg\psi \not\models_W$ *false*.
$\varphi \ {}^{M}{\geq}^{m} \ \psi$**:** $\exists E_j \exists E_i : \not\succ (E_j, E_i)$ if $\varphi \wedge \neg\psi \not\models_W$ *false*.
$\varphi \ {}^{m}{>}^{m} \ \psi$**:** $\exists E_j \forall E_i : \not\succeq (E_j, E_i)$.
$\varphi \ {}^{m}{\geq}^{m} \ \psi$**:** $\exists E_j \forall E_i : \not\succ (E_j, E_i)$.

The above formulae can be expressed as disjunctions.

Third step: Formulae expressing properties of the above predicates and their relations have to be added:

$$\not\succeq (A,B) \vee \succeq (A,B) \leftarrow \not\succeq (B,A),$$
$$\not\succeq (B,A) \vee [\succeq (A,B) \wedge \succeq (B,A)] \leftarrow \not\succ (B,A).$$

$$\succeq (A,C) \leftarrow \succeq (A,B) \wedge \succeq (B,C).$$
$$\| (A,B) \leftarrow \not\succeq (A,B) \wedge \not\succeq (B,A).$$
$$\succeq (A,B) \leftarrow \neg \not\succeq (A,B).$$
$$\mathit{false} \leftarrow \not\succeq (A,B) \wedge \succeq (A,B).$$
$$\succeq (A,A) \leftarrow .$$

3.4 Non-monotonic reasoning

To define the meaning of the program, we employ *optimal model semantics* [8].

Definition 11. ***(Atomic weight assignment)*** *[8, Def.2] An atomic weight assignment, $\wp$, for a program P, is a map from the Herbrand Base B_P of P to $\mathbb{R}_0^+$, where $\mathbb{R}_0^+$ denotes the set of nonnegative real numbers.*

Definition 12. ***(Aggregation strategy)*** *[8, Def.3] An aggregation strategy $\mathcal{A}$ is a map from*[3] $M^{\mathbb{R}_0^+}$ *to* $\mathbb{R}$.

Definition 13. ***(Herbrand Objective function)*** *[8, Def.4] The Herbrand Objective Function, HOF$(\wp, \mathcal{A})$ is a map from 2^{B_P} to $\mathbb{R}_0^+$ defined as follows:*

$$\mathrm{HOF}(\wp, \mathcal{A})(M) = \mathcal{A}(\{\wp(A) | A \in M\}) \ .$$

[2] Elements of E_i and E_j fulfill $\varphi \wedge \neg\psi$ and $\neg\varphi \wedge \psi$, resp., in the following list.

[3] Given a set X, M^X denotes the set of all multisets whose elements are in X.

Definition 14. ***(Optimal model)*** *[8, Def.5] Let P be a logic program, $\wp$ an atomic weight assignment, and $\mathcal{A}$ an aggregation strategy. Suppose that $\mathcal{F}$ is a family of models of P. We say that M is an optimal $\mathcal{F}$-model of P with regard to $(\wp, \mathcal{A})$ if:*

1. $M \in \mathcal{F}$;
2. $\nexists M' : M' \in \mathcal{F} \wedge \mathrm{HOF}(\wp, \mathcal{A})(M') < \mathrm{HOF}(\wp, \mathcal{A})(M)$.

We use the notation Opt$(P, \mathcal{F}, \wp, \mathcal{A})$ to denote the set of all optimal $\mathcal{F}$-models of P with regard to $(\wp, \mathcal{A})$.

Applying a variant of the connectivity principle (c.f. Def.7), distinguished models, defining the meaning of the program P, can be selected from stable models $\mathrm{ST}(P)$ of P so that the intensional relation, $\|$, of incomparable elements is minimal in the sense of set inclusion. Accordingly, we get the intended optimal model semantics of our program when we extend the notions of aggregation strategy and Herbrand objective function so that the relation of set inclusion can be captured.[4]

3.5 Preference operator

To embed preferences into RQL, a PO $\omega_{\mathcal{P}}$ returning only the best tuples in the sense of user preferences $\mathcal{P}$ is defined.

Ordering the partition of W according to the intensional relation $\succeq$ that is subsumed in an optimal model $M_P \in \mathrm{Opt}(P, \mathrm{ST}(P), \wp_0, \mathcal{A}_0)$, the most preferred worlds ultimately are located in maximal elements of the partition. To find the maximal elements, the ordered partition is associated with a positive datalog program consisting of one rule:

$$M(A) \leftarrow M(B) \wedge \succeq (A,B).$$

and facts: $\succeq (E_i, E_j) \in M_P$.

The least nonempty models of the above positive datalog program yield the interpretations of the predicate M identifying the maximal elements.

3.6 Preferences and relational algebra

The following theorem identifies a sufficient condition under which the PO and relational algebra (RA) selection commute. It provides a basic rule for rewriting preference queries using the standard strategies like *pushing selection down.*

Theorem 1 (Commuting with selection). *Given a relation schema R, a preference model $\mathcal{M}(R)$ over R, a partition $(E_1, \ldots, E_n)$ of W ordered by $\succeq$,*

[4] Note that Opt$(P, \mathrm{ST}(P), \wp_0, \mathrm{sum})$, in general, contains more than one optimal model.

i.e., $\forall w, w' \in W$ *with* $w \in E_i, w' \in E_j : i \leq j \iff w \succeq w'$*, and a selection condition* φ *over* R*, if the formula*

$$\forall w_1, w_2 : w_1 \in E_i \wedge w_2 \in E_j \wedge \succ (E_j, E_i) \\ \wedge \omega_\varphi(w_1) = w_1 \Rightarrow \omega_\varphi(w_2) = w_2$$

is valid, then for all instances $I(R)$*:*

$$\sigma_\varphi(\omega_\mathcal{P}(I(R))) = \omega_\mathcal{P}(\sigma_\varphi(I(R))) \ .$$

To check the validity of the above sufficient condition, we need to assign a meaning to the program that defines the selection condition $\varphi(x)$ and contains the following two rules defining $\phi(x)$:

$$\phi(x) \leftarrow \varphi(x).$$
$$\phi(x) \leftarrow y \in B \wedge x \in A \wedge \succeq (A, B) \wedge \neg \succeq (A, B) \wedge \phi(y).$$

and EDB consisting of the biggest possible instance $I(R)$ of R and facts: $\succeq (E_i, E_j) \in M_P$.[5] The validity of the sufficient condition, then, corresponds to that of the following equality: $\forall t \in I(R) : \varphi(t) = \phi(t)$.

4 Related work

The study of preferences in the context of database queries has been originated by Lacroix and Lavency [7]. Nevertheless, only at the turn of the millennium this area attracted broader interest again: Kießling et al. [6] and Chomicki et al. [3] have pursued independently a similar, *qualitative* approach within which preferences between tuples are specified directly, using binary *preference relations*. The embedding into RQL they have used is identical to the presented approach: an operator returning only the best preference matches. However, they haven't considered preferences between *sets* of elements. A special case of this embedding represents *skyline operator* introduced by Börzsönyi et al. [2].

5 Conclusions

Pursuing the goal of embedding preference queries in the RDM, it was shown that **user preferences can be captured in a logical language containing sixteen kinds of preferences**, and the semantics of the language can be defined with respect to the recent advances in logical representation of preferences allowing for **conflicting preferences**.

Embedding preferences into RQL was implemented through a **PO returning the most preferred sets of tuples**. This operator has a simple formal semantics defined by means of optimal models of a DLP.

A sufficient condition under which the PO and RA selection commute was identified, establishing thus a key rule for rewriting the preference queries using the standard algebraic optimization strategies.

Future work directions include developing algorithms for evaluating the PO and identification of its algebraic properties, in order to lay the foundation for the optimization of preference queries.

[5] Observe that the above program is *stratifiable*. Thus its stable model semantics can be computed in polynomial time.

References

1. Boella G. and van der Torre L.W.N., A Non-Monotonic Logic for Specifying and Querying Preferences. In L.P. Kaelbling and A. Saffiotti, (eds), IJCAI, Professional Book Center, 2005, 1549–1550
2. Börzsönyi S., Kossmann D., and Stocker K., The Skyline Operator. In Proceedings of the 17th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society, 2001, 421–430
3. Chomicki J., Preference Formulas in Relational Queries. ACM Trans. Database Syst., 28, 4, 2003, 427–466
4. Doyle J. and Wellman M.P., Representing Preferences as Ceteris Paribus Comparatives. In Decision-Theoretic Planning: Papers from the 1994 Spring AAAI Symposium, AAAI Press, Menlo Park, California, 1994, 69–75
5. Kaci S. and van der Torre L.W.N., Non-Monotonic Reasoning with Various Kinds of Preferences. In Ronen I.Brafman and U. Junker, (eds), IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling, August 2005, 112–117
6. Kießling W., Foundations of Preferences in Database Systems. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002, 311–322
7. Lacroix M. and Lavency P., Preferences; Putting More Knowledge into Queries. In P. M. Stocker, W. Kent, and P. Hammersley, (eds), VLDB, Morgan Kaufmann, 1987, 217–225
8. Leone N., Scarcello F., and Subrahmanian V., Optimal Models of Disjunctive Logic Programs: Semantics, Complexity, and Computation. IEEE Transactions on Knowledge and Data Engineering, 16, 4, April 2004, 487–503
9. Shoham Y., Nonmonotonic Logics: Meaning and Utility. In Proc. of the 10th IJCAI, Milan, Italy, 1987, 388–393
10. von Wright G., The Logic of Preference. Edinburgh University Press, Edinburgh, 1963
11. von Wright G., The Logic of Preference Reconsidered. Theory and Decision, 3, 1972, 140–169

Automatizácia dokazovania bezpečnostných vlastností kryptografických protokolov

Marián Novotný

Ústav informatiky Prírodovedeckej fakulty UPJŠ v Košiciach
Jesenná 5, 040 01 Košice, Slovensko
marian.novotny@upjs.sk

Abstrakt *Táto práca sa zaoberá analýzou kryptografických protokolov pomocou logík vier. Definujeme rozhodovaciu procedúru pre takéto logiky. Navrhneme a implementujeme program pre analýzu s možnosťou editácie odvodzovacích pravidiel. V programe implementujeme dve známe logiky: BAN,AUTLOG. Pomocou programu zanalyzujeme niektoré protokoly.*

1 Úvod

1.1 Kryptografické protokoly

Kryptografické protokoly sú špeciálnym druhom protokolov. Pod pojmom protokol myslíme konečnú postupnosť posielania správ medzi účastníkmi protokolu. Úlohou kryptografických protokolov je napĺňať komunikačné ciele účastníkov a zároveň zabezpečovať ich bezpečnostné potreby.

Ciele kryptografických protokolov môžu byť rôzne - môžu zabezpečovať autentifikáciu účastníkov, manažment kľúčov, elektronické obchodovanie, voľby atď.

Najdôležitejšu triedu kryptografických protokolov tvoria protokoly pre manažment kľúčov a autentifikáciu. Cieľom autentifikácie je zaručenie identity účastníka. Cieľom protokolov pre manažment kľúčov je dohodnúť medzi účastníkmi kľúč, ktorý bude používaný v ďalšej komunikácii.

Protokoly využívaju a vytvárajú rámec použitia základných kryptografických primitív ako sú napríklad symetrické a asymetrické šifrovanie, hašovacie funkcie, digitálne podpisy, certifikáty, časové pečiatky, príležitostné slová.

Predpoklady aj ciele protokolov na autentifikáciu a manažment kľúčov možno ľahko sformulovať. Tieto protokoly obvykle pozostávajú z výmeny niekoľkých, nie veľmi komplikovaných správ. Napriek tomu nie je jednoduché navrhnúť bezpečnostný protokol, ktorý je odolný voči útokom.

1.2 Analýza protokolov pomocou logík vier

U mnohých kryptografických protokolov sa ukázala ich zraniteľnosť. Vzniká prirodzená požiadavka analyzovať a formálne dokazovať vlastnosti protokolov.

Základné otázky, na ktoré hľadáme odpovede sú:

1. Čo protokolom dosiahneme? Robí to, čo chceme?
2. Aké predpoklady vyžaduje protokol?
3. Robí protokol niečo nadbytočné?

Prvou formálnou logikou pre analýzu protokolov bola *BAN logika*, publikovaná v [1]. BAN logika sa stala úspešnou metódou, pomocou ktorej sa odhalili viaceré bezpečnostné slabiny v kryptografických protokoloch. Existuje mnoho následovníkov BAN logiky – AUTLOG, GNY, Svo atď.

Pri analýze protokolu je konvenčná metóda popisu protokolu (definovanie postupnosti posielania a obsahu správ) v tejto metóde nahradená definovaním správ pomocou logických formúl. Takto popísaný protokol nazývame *idealizovaný protokol*. Pomocou odvodzovacích pravidiel sa z predpokladov protokolu a idealizovaného protokolu odvádzajú nové formuly. Toto odvádzanie je monotónne(nerúší sa platnosť žiadnych faktov). Úlohou odvádzania je odvodiť ciele protokolu.

Analýza protokolu má tieto fázy:

1. Idealizácia protokolu - prepis do formalizmu logiky
2. Sformulovanie predpokladov - definovanie počiatočných vier účastníkov
3. Sformulovanie cieľov protokolu
4. Rozhodnutie, či možno odvodiť ciele protokolu z predpokladov

V tejto práci sa budeme zaoberať touto metódou analýzy protokolov. Cieľom tejto práce je navrhnúť a implementovať algoritmus, ktorý automatizuje fázu č.4 pri analýze pomocou logík vier. Implementujeme v tomto programe najznámejšie formálne logiky ako BAN, AUTLOG a taktiež zanalyzujeme niektoré protokoly.

2 Rozhodovacia procedúra pre analýzu pomocou logík vier

Pri analýze pomocou logík vier potrebujeme rozhodnúť, či z množiny predpokladov (ľubovoľná množina konštantných termov) možno odvodiť pomocou aplikácie pravidiel logiky ciele protokolu. Pričom pravidlá

sú v tvare $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$, kde $\mathcal{H}_1,\ldots,\mathcal{H}_n,\mathcal{C}$ sú termy bez konštánt a cieľom protokolu je nejaký konštantný term. Pod *aplikáciou pravidla* $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ na predpoklady $H_1,\ldots,H_n$ myslíme nájsť substitúciu σ, ktorá je unifikátor termov $(H_1,\mathcal{H}_1),\ldots,(H_n,\mathcal{H}_n)$ a aplikovať substitúciu σ na term $\mathcal{C}$. Pod pojmom *logika* myslíme usporiadanú dvojicu - množinu funkčných symbolov FS a množinu odvodzovacích pravidiel $\vdash$ (označujeme $\langle FS,\vdash\rangle$). To, že je konštantný term t *odvoditeľný* z predpokladov Γ pomocou pravidiel $\vdash$ ozn. $\Gamma \vdash t$. Medzi základné požiadavky na rozhodovaciu procedúru patrí požiadavka, aby algoritmus skončil po konečnom počte krokov a dal správnu odpoveď (pozitívnu, alebo negatívnu).

Pri výbere metódy sme sa nechali inšpirovať článkom D. Monniauxa [6], v ktorom opisuje algoritmus pre analýzu protokolov pomocou GNY logiky. Podobnú metódu použil Darred Kindred vo svojej dizertačnej práci [4], avšak na rozdiel od tejto práce, sa pre odvádzanie termov využíva spätné odvádzanie.

2.1 Základné stratégie odvádzania

Poznáme dve základné metódy testovania odvoditeľnosti termu t z množiny predpokladov Γ pomocou pravidiel $\vdash$:

1. *Dopredné odvádzanie* štartuje z množiny predpokladov Γ . Aplikujeme odvodzovacie pravidlá na odvodenie nových formúl, zjednotíme množinu nových formúl s množinou predpokladov. Zistíme, či sme už odvodili t. Ak nie, tak pokračujeme s rozšírenou množinou.
2. *Spätné odvádzanie* štartuje z formuly t. Nájdeme všetky pravidlá, ktorých závery by sme mohli unifikovať s t. Ohodnotíme premenné pravidiel a rekurzívne pokračujeme s odvádzaním predpokladov ohodnotených pravidiel.

Nanešťastie pre obidve metódy existujú pravidlá, ktoré nie sú *vhodné* pre odvádzanie. Napr. v BAN logike je pravidlo $\frac{P \models \sharp(X)}{P \models \sharp((X,Y))}$, ktoré nie je vhodné pre dopredné odvádzanie. Rovnako v tejto logike existuje pravidlo $\frac{P \lhd \{X\}_K, P \models P \overset{K}{\longleftrightarrow} Q}{P \lhd X}$, s ktorým by sme zrejme mali problém pri spätnom odvádzaní.

Pravidlo považujeme za vhodné pre dopredné odvádzanie, ak platí $Var(\mathcal{C}) \subseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$.

Podobne ak $Var(\mathcal{C}) \supseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$, potom je pravidlo vhodné pre spätné odvádzanie.

Odvodzovacie pravidlá logík preto rozdelíme do disjunktných skupín *kompozičných* a *dekompozičných* pravidiel, pričom kompozičné pravidlá sú vhodné pre spätné a dekompozičné pre dopredné odvádzanie.

V dekomp. pravidle $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$, $m > 0$, $k \geq 0$ rozlišujeme *hlavné* ($\mathcal{H}_i$) a *pomocné* ($\mathcal{P}_i$) predpoklady, pre ktoré platí $\bigcup_{i=1}^{m} Var(\mathcal{H}_i) \supseteq \bigcup_{i=1}^{k} Var(\mathcal{P}_i)$. Pre pravidlo $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$

- ak $Var(\mathcal{C}) \supsetneq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$, potom je pravidlo kompozičné,
- ak $Var(\mathcal{C}) \subsetneq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$, potom je pravidlo dekompozičné,
- ak $Var(\mathcal{C}) = \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$, potom určíme, či je pravidlo kompozičné, alebo dekompozičné.

Toto rozdelenie vychádza z nasledujúceho pozorovania. Ak by sme pri doprednom odv. použili pravidlo, v ktorom $|Var(\mathcal{C})| > |\bigcup_{i=1}^{n} Var(\mathcal{H}_i)|$, tak aspoň jedna premmenná v závere by nebola inicializovaná. Podobné problémy by sme mali použitím pravidla, v ktorom $|Var(\mathcal{C})| < |\bigcup_{i=1}^{n} Var(\mathcal{H}_i)|$ pri spätnom odvádzaní. Tu by sme nemali ohodnotenú niektorú premennú v predpokladoch pravidla.

2.2 Kritéria pre logiky

V tejto kapitole definovaná rozhodovacia procedúra si nekladie nárok byť univerzálnou procedúrou pre logiky. V tejto časti definujeme požiadavky pre logiky, s ktorými budeme pracovať.

Pre odvodzovacie pravidlo $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ musí platiť:

- $Var(\mathcal{C}) \subseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$, alebo
- $Var(\mathcal{C}) \supseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i)$.

Ďalej budeme požadovať, aby bolo možné odvádzať bez toho, aby sme použitím kompozičného pravidla „zložili“ formulu, ktorú by sme následne použitím dekompozičného pravidla „rozložili“. Hovoríme, že odvodenie formuly $\Gamma \vdash t$ je *normálnym odvodením*, ak v strome odvodenia $\Gamma \vdash t$ nevznikli aplikáciou kompozičného pravidla hlavné predpoklady dekompozičných pravidiel. Definícia narmálneho odvodenia formuly $\Gamma \vdash t$ hovorí, že ak sa v strome odvodenia nachádza vrchol C, ktorý vznikol aplikáciou dekompozičného pravidla $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$ na predpoklady $H_1,\ldots,H_m,P_1,\ldots,P_k$, potom vrcholy $H_1,\ldots,H_m$ nemohli vzniknúť aplikáciou kompozičného pravidla. Ak $H_1,\ldots,H_m$ nie sú predpokladmi logiky, tak pravidlá $p_1,\ldots,p_m$ musia byť dekompozičné.

$$\frac{\frac{\vdots}{H_1}p_1 \;\cdots\; \frac{\vdots}{H_m}p_m \quad \frac{\vdots}{P_1} \;\cdots\; \frac{\vdots}{P_k}}{C}$$

Logika $\langle FS,\vdash\rangle$ spĺňa *normálne odvodzovacie kritérium*, ak pre ľubovolnú množinu predpokladov Γ, formulu t platí, že ak existuje odvodenie $\Gamma \vdash t$, potom existuje normálne odvodenie $\Gamma \vdash t$.

2.3 Transformácia logiky

Hlavnou myšlienkou transformácie logiky $\langle FS, \vdash \rangle$ je zostrojiť inú logiku $\langle FS', \vdash' \rangle$, v ktorej budeme vedieť po konečnom počte krokov odpovedať na otázku, či platí $\Gamma' \vdash' t$. V tejto časti definujeme transformáciu pravidiel logiky a ukážeme dôležitú vlastnosť $\Gamma' \vdash' t \equiv \Gamma \vdash t$.

Definícia transformácie logiky Zavedieme symbol *cieľ* '$\boxdot$', ktorý neoznačuje žiadnu funkciu. $\boxdot t$ znamená, že „chceme zložiť formulu t". Tento symbol použijeme na realizáciu stratégie odvádzania, ktorej hlavnou ideou je použiť kompozičné pravidlo, len ak záver tohto pravidla budeme chcieť „zložiť". Pre množinu funkčných symbolov logiky FS definujeme množinu funkčných symbolov transformovanej logiky takto:

$$FS' = FS \cup \{\langle \boxdot f, i\rangle;\ \langle f, i\rangle \in FS \wedge i > 0\}.$$

Pre množinu odvodzovacích pravidiel $\vdash$ definujeme množinu transformovaných pravidiel $\vdash'$ takto:

1. pre kompozičné pravidlo $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$
   (a) $\frac{\boxdot\mathcal{C},\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$,
   (b) $\frac{\boxdot\mathcal{C}}{\boxdot\mathcal{H}_i}$ pre $i \in \{1,\ldots,n\}$,
2. pre dekompozičné pravidlo $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$, $m > 0$, $k \geq 0$
   (a) $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$,
   (b) ak $k \geq 1$ potom aj $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m}{\boxdot\mathcal{P}_i}$ pre $i \in \{1,\ldots,k\}$.

Vlastnosti transformácie

Lemma 1. *Systém pravidiel $\vdash'$, ktorý vznikol transformáciou z množiny pravidiel $\vdash$, je vhodný pre dopredné odvádzanie.*

Dôkaz. Chceme ukázať, že pre každé pravidlo $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ existuje množina $M \subseteq \{\mathcal{H}_1,\ldots,\mathcal{H}_n\}$ taká, že platí

$$\bigcup_{\mathcal{H}\in M} Var(\mathcal{H}) = \bigcup_{i=1}^{n} Var(\mathcal{H}_i) \cup Var(\mathcal{C}).$$

Dôkaz urobíme rozborom prípadov podľa typu pravidla. Ak je pravidlo tvaru:

1.a $\frac{\boxdot\mathcal{C},\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}} \Longrightarrow M = \{\boxdot\mathcal{C}\}$,
1.b $\frac{\boxdot\mathcal{C}}{\boxdot\mathcal{H}} \Longrightarrow M = \{\boxdot\mathcal{C}\}$,
2.a $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}} \Longrightarrow M = \{\mathcal{H}_1,\ldots,\mathcal{H}_m\}$,
2.b $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m}{\boxdot\mathcal{P}} \Longrightarrow M = \{\mathcal{H}_1,\ldots,\mathcal{H}_m\}$.

Všetky tvrdenia vyplývajú priamo z definície kompozičných, dekompozičných, transformovaných pravidiel. □

Veta 1. *Pre ľubovolnú logiku $\langle FS, \vdash \rangle$, ktorá spĺňa normálne odvodzovacie kritérium, pre ľubovolnú množinu predpokladov Γ a všetky formule t platí*

$$\Gamma \cup \{\boxdot t\} \vdash' t \equiv \Gamma \vdash t.$$

Dôkaz.

1. $\Gamma \cup \{\boxdot t\} \vdash' t \Longrightarrow \Gamma \vdash t$
   Nech $t'_1,\ldots,t'_m = t$ je vytvárajúca postupnosť odvodenia $\Gamma \cup \{\boxdot t\} \vdash' t$, postupnosť $t_1,\ldots,t_n = t$ vznikne z $t'_1,\ldots,t'_m = t$ vynechaním termov so špeciálnym symbolom $\boxdot$.
   Matematickou indukciou vzhľadom na dĺžku vytvárajúcej postupnosti chceme ukázať, že pre každé $i \leq n$ platí, že postupnosť $t_1,\ldots,t_i$ je vytvárajúca postupnosť odvodenia $\Gamma \vdash t_i$. Predpokladajme, že tvrdenie platí pre $j < i$.
   - Ak $t_i \in \Gamma$, tak to platí triviálne.
   - Ak $t_i \notin \Gamma$, potom term t_i vznikol vo vytvárajúcej postupnosti $t'_1,\ldots,t'_n$ aplikáciou pravidla tvaru:
     (a) $\frac{\boxdot\mathcal{C},\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ a z IP platí tvredenie pre $H_1,\ldots,H_n$, aplikáciou pravidla $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ dostávame $\Gamma \vdash t_i$,
     (b) $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$ a tvrdenie vyplýva priamo z indukčného predpokladu.

2. $\Gamma \vdash t \Longrightarrow \Gamma \cup \{\boxdot t\} \vdash' t$
   Keďže logika spĺňa normálne odvodzovacie kritérium, tak ak existuje odvodenie $\Gamma \vdash t$, potom musí existovať normálne odvodenie. Dôkaz urobíme pomocou indukcie cez strom normálneho odvodenia $\Gamma \vdash t$, pričom predpokladáme, že tvrdenie platí pre podstromy vrcholu t v strome odvodenia.
   - Ak $t \in \Gamma$, tak to platí triviálne.
   - Ak sa formula t odvodila aplikáciou kompozičného pravidla $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ z predpokladov $\{H_1,\ldots,H_n\}$, potom
     (a) podľa definície transformácie logiky $\frac{\boxdot\mathcal{C}}{\boxdot\mathcal{H}_1,\ldots,\boxdot\mathcal{H}_n} \subseteq \vdash'$ a aplikáciou týchto pravidiel na $\boxdot t$ dostávame $\Gamma \cup \{\boxdot t\} \vdash' \boxdot H_i$,
     (b) keďže $\Gamma \vdash H_i$ sú odvodenia v podstromoch vrcholu t, môžme použiť indukčný predpoklad . Odtiaľ máme $\Gamma \cup \{\boxdot H_i\} \vdash' H_i$.

     Spojením (a), (b) máme $\Gamma \cup \{\boxdot t\} \vdash' H_i$. Aplikáciou $\frac{\boxdot\mathcal{C},\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ na $\{\boxdot t, H_1,\ldots,H_n\}$ dostávame odvodenie $\Gamma \cup \{\boxdot t\} \vdash' t$.
   - Ak sa formula t odvodila aplikáciou dekompozičného pravidla $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$ z predpokladov $\{H_1,\ldots,H_m,P_1,\ldots,P_k\}$, potom
     (a) Z IP vyplýva $\Gamma \cup \{\boxdot H_i\} \vdash' H_i$. Podľa definície normálneho odvodenia $H_i \in \Gamma$, alebo H_i vznikol aplikáciou dekompozičného pravidla a nepotrebujeme $\boxdot H_i$, teda máme $\Gamma \vdash' H_i$.
     (b) Aplikikáciou pravidliel $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m}{\boxdot\mathcal{P}_1,\ldots,\boxdot\mathcal{P}_k}$ na predpoklady $\{H_1,\ldots,H_m\}$ (podľa bodu (a) platí $\Gamma \vdash' H_i$) dostávame $\Gamma \vdash' \boxdot P_i$.

(c) Z IP vyplýva $\Gamma \cup \{\boxdot P_i\} \vdash' P_i$. Spojením (b), (c) dostávame $\Gamma \vdash' P_i$. Aplikáciou pravidla $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_m,\mathcal{P}_1,\ldots,\mathcal{P}_k}{\mathcal{C}}$ na $\{H_1,\ldots,H_m,P_1,\ldots,P_k\}$ dostávame odvodenie $\Gamma \vdash' t$, teda aj $\Gamma \cup \{\boxdot t\} \vdash' t$. □

2.4 Definícia rozhodovacej procedúry

Funkcia $odvodDopr(\langle FS,\vdash\rangle,\Gamma,t) : boolean$ dá dopredným odvádzaním odpoveď na otázku $? - \Gamma \vdash t$.
Funkcia $rozhodni(\langle FS,\vdash\rangle,\Gamma,t) : boolean$ dá odpoveď na otázku $? - \Gamma \vdash t$ pre logiky $\langle FS,\vdash\rangle$, ktoré spĺňajú normálne odvodzovacie kritérium.

```
function odvodDopr(⟨FS,⊢⟩, Γ, t) : boolean;
begin
    if t ∈ Γ then return := true;
    else begin
      M := {f ; f ∉ Γ ∧ f vznikla aplikáciou pravidla
z ⊢ na predpoklady z Γ};
      if M = ∅ then return := false;
      else return := odvodDopr(⟨FS,⊢⟩, Γ ∪ M, t);
    end;
end;

function rozhodni(⟨FS,⊢⟩, Γ, t) : boolean;
begin
    ⟨FS',⊢'⟩ := transformuj(⟨FS,⊢⟩);
    return := odvodDopr(⟨FS',⊢'⟩, Γ ∪ {⊡t}, t);
end;
```

3 Implementácia logík pomocou rozhodovacej procedúry

V tejto časti definujeme funkčné symboly a odvodzovacie pravidlá formálnych logík BAN a AUTLOG s úpravami, aby sme ich mohli implementovať v rozhodovacej procedúre.

3.1 BAN logika

Množinu funkčných symbolov, odvodzovacie pravidlá definujeme podľa článku [1], pričom budeme dodržiavať pôvodnú notáciu BAN logiky.

Množina funkčných symbolov Kvôli presnejšiemu vymedzeniu vzťahu medzi súkromným a verejným kľúčom, rozšírime množinu funkčných symbolov o binárnu funkciu *inv* ($inv(V,S)$ znamená $S = V^{-1}$). Zreťazenie formúl reprezentujeme pomocou binárnej funkcie *pair*. Napríklad zreťazenie formúl X,Y,Z zapíšeme ako $pair(X,pair(Y,Z))$.

funkčný symbol	BAN	význam
$believes(P,X)$	$P \models X$	P verí v X
$sees(P,X)$	$P \triangleleft X$	P vidí správu X
$said(P,X)$	$P \mid\sim X$	P povedal X
$jurisdiction(P,X)$	$P \Rightarrow X$	P má rozhodovacie právo ohľadne X
$fresh(X)$	$\sharp(X)$	formula X je čerstvá
$sharedkey(P,Q,K)$	$P \overset{K}{\leftrightarrow} Q$	P a Q môžu používať zdielaný kľúč K
$publickey(K,P)$	$\overset{+K}{\mapsto} P$	P má verejný kľúč K
$secret(P,Q,Y)$	$P \overset{Y}{=} Q$	P a Q zdieľajú spoločné tajomstvo Y
$encrypt(X,K)$	$\{X\}_K$	formula X je zašifrovaná pomocou kľúča K
$pair(X,Y)$	(X,Y)	zreťazenie formúl X,Y
$inv(V,S)$	$(S = V^{-1})$	V,S je ver., súkr. kľúč
$combine(X,Y)$	$\langle X\rangle_Y$	X je skombinovaná s Y

Odvodzovacie pravidlá Implementovali sme pravidlá BAN logiky ako boli publikované v [1]. Nové pravidlá sme pridávali najmä kvôli symetrii zreťazenia. Keďže sme nepotrebovali využiť vlastnosť asociatívnosti zreťazenia formúl, nemuseli sme pridať pravidlá realizujúce túto vlastnosť.

Pridávanie pravidiel a rozdelenie do disjunktných množín kompozičných a dekompozičných pravidiel sme robili tak, aby platilo normálne odvodzovacie kritérium. V práci [3] sme vlastnosť normálneho odvodzovacieho kritéria pre takto definovanú logiku dokázali.

Pre analýzu napr. protokolu Needham-Schroeder s verejnými kľúčmi [1, s.34] je potrebné pravidlo 1, ktoré autori článku[1] implicitne používajú pre odvádzanie, avšak explicitne ho neformulovali.

$$\frac{P \triangleleft \langle X\rangle_Y , P \models P \overset{Y}{=} Q, P \models \sharp(Y)}{P \models \sharp(X)} \qquad (1)$$

Toto pravidlo hovorí o tom, že ak účastník P vidí formulu X skombinovanú s Y, verí že Y je zdieľané tajomstvo. Navyše ešte verí, že Y je čerstvé, potom P verí, že je čerstvá aj formula X.

V tejto časti vymenujeme kompozičné a dekompozičné pravidlá BAN logiky so stručným popisom významu jednotlivých pravidiel.

Kompozičné pravidlá BAN logiky

$\frac{P \models X, P \models Y}{P \models (X,Y)}$ ak P verí formulám, tak verí aj ich zreťazeniu

$\frac{P \models Q \models X, P \models Q \models Y}{P \models Q \models (X,Y)}$ podobne, ale pre „druhý stupeň“

$\frac{P \models \sharp(X)}{P \models \sharp((X,Y))}$ ak je časť správy čerstvá (ak P verí, že je čerstvá), potom je čerstvá celá správa

$\frac{P \models \sharp(X)}{P \models \sharp((Y,X))}$ kvôli symetrii zreťazenia

$\frac{P \models \sharp(X)}{P \models \sharp(\{X\}_K)}$ ak je správa čerstvá, potom je čerstvá aj zašifrovaná správa

$\frac{P \models \sharp(X)}{P \models \sharp(\langle X \rangle_Y)}$ ak je správa čerstvá, potom je čerstvá aj skombinovaná správa

Dekompozičné pravidlá BAN logiky V zozname dekompozičných pravidiel názorne odlíšime pomocné od hlavných predp. pomocou čiary nad pomocnými.

$\frac{P \triangleleft \{X\}_K, P \models P \overset{K}{\longleftrightarrow} Q}{P \models Q \mid\sim X}$ ak K je zd. kľúč medzi P a Q a P vidí správu zašifr. s K, tak verí, že ju poslal Q

$\frac{P \triangleleft \langle X \rangle_Y, P \models P \overset{Y}{\rightleftharpoons} Q}{P \models Q \mid\sim X}$ ak Y je tajomstvo medzi P, Q a P vidí správu s Y, tak P verí, že ju poslal Q

$\frac{P \models Q \mid\sim X, \overline{P \models \sharp(X)}}{P \models Q \models X}$ kontrola čerstvosti „nonce“

$\frac{P \models Q \Rightarrow X, \overline{P \models Q \models X}}{P \models X}$ ak P verí, že Q má právo rozhodovať ohľadne X a verí, že Q verí X, potom P verí X

$\frac{P \triangleleft \{X\}_K, P \models P \overset{K}{\longleftrightarrow} Q}{P \triangleleft X}$ účastník môže pomocou sym. kľúča dešifrovat správu

$\frac{P \triangleleft \{X\}_K, \overline{P \models \overset{+K}{\mapsto} P}}{P \triangleleft X}$ podobne

$\frac{P \triangleleft \langle X \rangle_Y}{P \triangleleft X}$ P môže rozložiť skomb. správu

Pre úplnosť ešte uvedieme zvyšné dekompozičné pravidlá, ktorých význam je zrejmý.

$\frac{P \models (X,Y)}{P \models X}$ $\frac{P \models (X,Y)}{P \models Y}$ $\frac{P \models Q \models (X,Y)}{P \models Q \models X}$ $\frac{P \models R \overset{Y}{\rightleftharpoons} Q}{P \models Q \overset{Y}{\rightleftharpoons} R}$

$\frac{P \triangleleft (X,Y)}{P \triangleleft X}$ $\frac{P \triangleleft (Y,X)}{P \triangleleft (X,Y)}$ $\frac{P \models Q \mid\sim (X,Y)}{P \models Q \mid\sim X}$ $\frac{P \models R \overset{K}{\longleftrightarrow} Q}{P \models Q \overset{K}{\longleftrightarrow} R}$

$\frac{P \triangleleft \langle X \rangle_Y, P \models P \overset{Y}{\rightleftharpoons} Q, \overline{P \models \sharp(Y)}}{P \models \sharp(X)}$ $\frac{P \models Q \models S \overset{K}{\longleftrightarrow} R}{P \models Q \models R \overset{K}{\longleftrightarrow} S}$

$\frac{P \triangleleft \{X\}_S, P \models \overset{+V}{\mapsto} Q, \overline{(S = V^{-1})}}{P \models Q \mid\sim X}$ $\frac{P \models Q \models S \overset{Y}{\rightleftharpoons} R}{P \models Q \models R \overset{Y}{\rightleftharpoons} S}$

$\frac{P \triangleleft \{X\}_S, P \models \overset{+V}{\mapsto} Q, \overline{(S = V^{-1})}}{P \triangleleft X}$ $\frac{P \models Q \models (X,Y)}{P \models Q \models X}$

3.2 Autlog

Logika Autlog je rozšírenie BAN logiky, prvýkrát bola publikovaná v článku[5]. Autormi tejto logiky sú Volker Kessler a Gabriele Wedel.

Definícia množiny funkčných symbolov Množina funkčných symbolov logiky Autlog pozostáva z funkčných symbolov BAN logiky a funkcií, ktoré uvedieme v tejto časti.

funkčný symbol	notácia	význam
$recognizable(X)$	$\phi(X)$	X je rozpoznateľná
$says(P,X)$	$P \thickapprox X$	P nedávno povedal X
$hash(X)$	$hash(X)$	použitie „hašovacej“ funkcie na formulu$X$
$mac(K,X)$	$mac(K,X)$	použitie MAC funkcie s kľúčom K na X

Množinu funkčných symbolov rozšírime o pomocné symboly $\alpha, \beta, \gamma, \delta, \epsilon, \eta, \theta$ s árnosťou 2, ktoré poslúžia pri implementácií pravidiel v rozhodovacej procedúre.

Odvodzovacie pravidlá Podľa definície odvodzovacieho pravidla, v pravidle $\frac{\mathcal{H}_1,\ldots,\mathcal{H}_n}{\mathcal{C}}$ musí platiť

$$Var(\mathcal{C}) \subseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i) \text{ alebo}$$
$$Var(\mathcal{C}) \supseteq \bigcup_{i=1}^{n} Var(\mathcal{H}_i).$$

Takže nemôžme priamo pridať pravidlo ako je napríklad

$$\frac{P \triangleleft P \overset{K}{\longleftrightarrow} Q, P \models \sharp(P \overset{K}{\longleftrightarrow} Q)}{P \models \sharp(\{X\}_K)} \tag{2}$$

Namiesto pravidla 2 pridáme jedno dekompozičné a jedno kompozičné pravidlo

$$\frac{P \triangleleft P \overset{K}{\longleftrightarrow} Q, \overline{P \models \sharp(P \overset{K}{\longleftrightarrow} Q)}}{\alpha(P,K)}, \quad \frac{\alpha(P,K)}{P \models \sharp(\{X\}_K)}.$$

Pri takomto pridávaní pravidiel si musíme dávať pozor, aby sme nepokazili vlastnosť normálneho odvodzovacieho kritéria.

Vzhľadom na to, že logika AUTLOG pozostáva zo 43 dekompozičných a 19 kompozičných pravidiel, nebudeme uvádzať úplný zoznam odvodzovacích pravidiel, ktoré si možno pozrieť pomocou programu. Podrobnejšie informácie o logike Autlog a jednotlivých pravidlách môže čitateľ nájsť v článku [5].

4 Program ABLOB

Program ABLOB sme implementovali v objektovo-orientovanom programovací jazyku DELPHI7. Podrobný popis návrhu a implementácie aplikácie môže čitateľ nájsť v [3].

Program ABLOB

- umožňuje pomocou uživateľského rozhrania editovať logiku t.j. definovať funkčné symboly, kompozičné a dekompozičné pravidlá. Taktiež dovoľuje editovať protokol t.j. konštantné funkčné symboly (napr. mená účastnikov), predpoklady protokolu a ciele protokolu , pričom vykonáva syntaktickú kontrolu, kontroluje správne určenie pravidiel. Kvôli lepšej čitateľnosti editor farebne odlišuje jednotlivé znaky.
- automaticky transformuje logiku, spustí rozhodovaciu procedúru v transformovanej logike. Výsledok analýzy však prevedie do pôvodnej logiky. Ak sa podarí odvodiť cieľ, tak vypíše čitateľne naformátovanú vytvárajúcu postupnosť odvodenia. Ak sa podarí odvodiť niektorý predpoklad protokolu, vo výsledku sa zobrazí upozornenie, že ten predpoklad je rendundantný. Ak sa nepodarí odvodiť ciel, tak sa program pokúsi dodefinovať niektoré predpoklady (medzi predpoklady pridá term t, ak sa podarilo odvodiť $\boxdot t$ a pokusi sa nájsť minimálne dodatočne pridané predpoklady). Ak sa podarí odvodiť protokol s dodatočnými predpokladmi, tak vypíše vytvárajúcu postupnosť, v ktorej odlíši dodatočne pridané predpoklady.
- umožňuje zobrazovať výsledky analýzy v systéme LaTeX. Užívateľ zadefinuje LaTeX-ovský balíček s rovnakým názvom ako sa volá logika, pre každý funkčný symbol zadefinuje nový príkaz. V programe taktiež možno nastaviť externý kompilátor LaTeX-u a dvi prehliadač.

5 Analýza protokolov pomocou programu ABLOB

Cieľom analýz protokolov, ktoré sme realizovali pomocou programu nebolo objavovanie nových chýb protokolov, alebo pôvodná analýza nových protokolov. Našim cieľom bolo v programe vyskúšať analýzy protokolov zo známych článkov, pomocou nich otestovať program a implementácie jednotlivých logík. Zistiť, či sa dopracujeme k rovnakým výsledkom a vytvoriť v programe databázu protokolov, ktorá môže poslúžiť pre študijné účely záujemcom o kryptografické protokoly.

5.1 Analýza protokolov pomocou BAN logiky

Pomocou programu sme zanalyzovali všetky protokoly z článku [1] t.j. protokoly Otway-Rees, Needham Shroeder-SK, Kerberos, Wide-mouthed-frog, Andrew RPC1, Andrew RPC2, Andrew RPC3, Yahalom, Needham Shroeder PK, X509. Vzhľadom na to, že čitateľ si môže pomocou programu prezrieť podrobné analýzy jednotlivých protokolov, nebudeme uvádzať výsledky analýz.

Pri analýze protokolu Kerberos[1, s.22] sme pridali prirodzený predpoklad $A \models \sharp(T_A)$, bez ktorého by sme neodvodili cieľ $A \models B \models A \xleftrightarrow{K_{AB}} B$.

Pre analýzu protokolu Andrew RPC3[1, s.29] bolo potrebné pridať pravidlo 3, ktoré autori neuviedli, aj napriek tomu, že ho pri analýze použili.

$$\frac{P \lhd \{X\}_K, P \models P \xleftrightarrow{K} Q, \overline{P \models \sharp(K)}}{P \models \sharp(X)} \tag{3}$$

Pre analýzu protokolu Yahalom[1, s.30] bolo potrebné pridať pravidlo 4, ktoré však autori uviedli v poznámke ku analýze[1, s.33].

$$\frac{P \models R \mid\sim P \xleftrightarrow{K} Q, P \lhd \{X\}_K}{P \lhd X} \tag{4}$$

5.2 Analýza protokolov pomocou logiky AUTLOG

Pomocou logiky AUTLOG sme zanalyzovali jednoduché protokoly „výziev a odpovedí“ z článku[5]. Kôli otestovaniu programu logikou s veľkým počtom pravidiel sme pomocou logiky AUTLOG zanalyzovali protokol Kerberos podobne ako v BAN logike, pričom sme prišli k rovnakým záverom ako pomocou BAN logiky, ale s podrobnejšími odvodeniami.

6 Záverečné zhrnutie

V tejto práci sme formalizovali pojem logiky pre analýzu bezpečnosti kryptografických protokolov, definovali rozhodovaciu procedúru, ktorá rozhodne, či je formula odvoditeľná z množiny predpokladov pomocou odvodzovacích pravidiel logiky.

Navrhli a implementovali sme program ABLOB. Ide o program s užívateľsky príjemným prostredím, možnosťou editácie logiky, protokolu, možnosťou formátovaného výstupu analýzy protokolu do systému LaTeX. V tomto programe sme implementovali známe logiky vier – BAN a AUTLOG . Pomocou programu ABLOB sme zanalyzovali niektoré protokoly.

Podobný program s názvom REVERE vytvoril Darrell Kindred pod vedením Jeannette Wing ako súčasť dizertačnej práce[4]. Žiaľ, nepodarilo sa nám získať tento program, takže porovnávať programy môžme len z tabuľky časov[4, s.110] a výstupov analýz, ktoré sú prílohou práce [4].

Výstupy programu REVERE sú v textovej forme, vo výstupoch sú uvedené ciele protokolu, odvoditeľnosť jednotlivých cieľov a zoznam všetkých odvedených formúl. V programe ABLOB sú výstupy analýzy

úplné, okomentované odvodenia cieľov a zoznamy nevyužitých predpokladoch pri odvádzaní. Tieto výstupy je tiež možné zobrazovať v systéme LaTeX. Program REVERE využíva spätné odvádzanie termov. Na rozdiel od programu ABLOB podporuje odvádzanie pomocou „prepisovacích“ pravidiel.

Uvedieme tabuľku porovnávajúcu časy analýz protokolov. Program REVERE bol testovaný na počítači Digital AlphaStation 500 s procesorom Alpha 500 MHz [4, s.110].

Program ABLOB bol testovaný na počítači s procesorom Athlon 2000+ XP.

logika	protokol	REVERE	ABLOB
BAN	Kerberos	4.7s	0.062s
	Needham-Schr(SK)	1.5s	0,078s
	CCITT X.509	23.8s	0.047s
	Wide-Mouth Frog	19.3s	0.031s
AUTLOG	Kerberos	11.3s	0.25s

Referencie

1. Burrows M., Abadi M., and Needham R., A Logic of Authentication. Technical Report 39, Digital Equipment Corporation, Systems Research Centre, 1989
2. Jenčušová E. and Jirásek J., Formal Methods of Analysis of Security Protocols. Tatra Mt. Math. Publ. 25, 2002, 1–10
3. Novotný M., Automatizácia dokazovania odolnosti kryptografických protokolov voči útokom. Diplomová práca, Prírodovedecká fakulta UPJŠ, 2005
4. Kindred D., Theory Generation for Security Protocols. PhD thesis, Carnegie Mellon University, 1999
5. Kessler V. and Wedel G., Autlog Ś an Advanced Logic of Authentication. Proceedings of the Computer Security Foundations Workshop VII, 1994
6. Monniaux D., Decision Procedures for the Analysis of Cryptographic Protocols by Logics of Belief. 12th Computer Security Foundations Workshop, 1999
7. Monniaux D., Analysis of Cryptographic Protocols Using Logics of Belief: an Overview. Journal of Telecomunications and Information Technology, 2002

ITAT'07

Information Technology – Applications and Theory

EXTENDED ABSTRACT

Complexity of training data in neural-network learning*

Věra Kůrková

Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague
vera@cs.cas.cz, http://www.cs.cas.cz/∼vera

Abstrakt *A measure characterizing complexity of data for training neural networks is proposed. The measure depends on the type of computational units in the network.*

Learning of neural networks can be formally described as minimization of error functionals over parameterized sets of input-output functions computable by a given class of networks. The error functionals are determined by training data described either by a discrete sample of input-output pairs or a probability measure with respect to which the training sample is chosen.

The *expected error functional* $\mathcal{E}_\rho$ determined by a non degenerate (no nonempty open set has measure zero) *probability measure* ρ on $Z = X \times Y$ (where X is a *compact* subset of $\mathbb{R}^d$ and Y a *bounded* subset of $\mathbb{R}$) is defined as

$$\mathcal{E}_\rho(f) = \int_Z (f(x) - y)^2 \, d\rho.$$

The *empirical error functional* $\mathcal{E}_z$ determined by a *sample of data* $z = \{(u_i, v_i) \in X \times Y \,|\, i = 1, \ldots, m\}$ is defined as

$$\mathcal{E}_z(f) = \frac{1}{m} \sum_{i=1}^{m} (f(u_i) - v_i)^2.$$

The simplest model of a neural network is a *network with n hidden units and one linear output unit*. The input-output functions of networks of this type are of the form

$$\mathrm{span}_n \, G = \left\{ \sum_{i=1}^{n} w_i g_i \,|\, w_i \in \mathbb{R},\, g_i \in G \right\},$$

where G is the set of functions that can be computed by *computational units* of a given type (such as perceptrons or radial-basis functions). The number n of hidden units plays the role of a measure of *model complexity of the network*. Its size is critical for a feasibility of an implementation.

We propose to measure complexity of training data with respect to a type of computational units G by the *speed of decrease* of infima of error functionals over $\mathrm{span}_n G$ with n increasing. The faster such a rate of decrease, the better approximation of global minima of error functionals can be achieved using networks with a reasonably moderate number of computational units.

The speed of decrease of infima of error functionals can be estimated in terms of a certain norm tailored to the set G of the functions at which these functionals achieve global minima.

Let $(\mathcal{L}^2_{\rho_X}(X), \|.\|_{\mathcal{L}^2_{\rho_X}})$ denote the space of functions satisfying $\int_X f^2 d\rho_X < \infty$, where ρ_X denotes the *marginal probability measure* on X defined for every $S \subseteq X$ as $\rho_X(S) = \rho(\pi_X^{-1}(S))$ with $\pi_X : X \times Y \to X$ denoting the projection.

The global minimum of the expected error $\mathcal{E}_\rho$ over the whole space $\mathcal{L}^2_{\rho_X}(X)$ is achieved at the *regression function* f_ρ defined for all $x \in X$ as

$$f_\rho(x) = \int_Y y \, d\rho(y|x),$$

where $\rho(y|x)$ is the *conditional (w.r.t. x) probability measure* on Y (see, e.g., [1]). The global minimum of the empirical error $\mathcal{E}_z$ is achieved at any function interpolating the sample z.

The next theorem shows that the speed of decrease of infima of error functionals over sets $\mathrm{span}_n G$ can be estimated in terms of a norm, called G-variation. This norm is defined for any bounded nonempty subset G of $\mathcal{L}^2_{\rho_X}(X)$ as the *Minkowski functional of the closed convex symmetric hull of G*, i.e.,

$$\|f\|_G = \inf \left\{ c > 0 \,:\, c^{-1} f \in \mathrm{cl}\,\mathrm{conv}\,(G \cup -G) \right\}, \quad (1)$$

where the closure cl is taken with respect to the topology generated by the norm $\|.\|_{\mathcal{L}^2_{\rho_X}}$ and conv denotes the convex hull.

Theorem 1. *Let d, m, n be positive integers, both $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$ be compact, $z = \{(u_i, v_i) \in X \times Y \,|\, i = 1, \ldots, m\}$ with all u_i distinct, ρ be a non degenerate probability measure on $X \times Y$, and G be a bounded subset of $\mathcal{L}^2_{\rho_X}(X)$ with $s_G = \sup_{g \in G} \|g\|_{\mathcal{L}^2_{\rho_X}}$. Then*

$$\inf_{f \in span_n G} \mathcal{E}_\rho(f) - \mathcal{E}_\rho(f_\rho) \leq \frac{s_G^2 \|f_\rho\|_G^2}{n}$$

and for every $h \in \mathcal{L}^2_{\rho_X}(X)$ interpolating the sample z,

* This work was partially supported by the project 1ET100300517 of the program Information Society of the National Research Program of the Czech Republic and the Institutional Research Plan AV0Z10300504.

$$\inf_{f\in span_n G} \mathcal{E}_z(f) \leq \frac{s_G^2 \|h\|_G^2}{n}.$$

For the proof see [4].

So the infima of error functionals achievable over networks with n hidden units computing functions from a set G decrease at least as fast as $\frac{1}{n}$ times the square of the G-variational norm of the regression function or some interpolating function. When these norms are small, good approximations of the two global minima, $\min_{f\in\mathcal{L}^2_{\rho_X}(X)} \mathcal{E}_\rho(f) = \mathcal{E}_\rho(f_\rho)$ and $\min_{f\in\mathcal{L}^2_{\rho_X}(X)} \mathcal{E}_z(f) = 0$, can be obtained using networks with a moderate number of units.

We propose to use the magnitude of the G-variation of the regression function f_ρ and the infimum of G-variations of all functions interpolating the sample z as measures of *complexity of data* given by the probability measure ρ and the sample z, resp.

For some sets G, there is an interesting relationship between G-variation, smoothness, and dimensionality.

Variation with respect to sigmoidal perceptrons is related to smoothness described by restriction on partial derivatives [2] and variation with respect to Gaussian radial-basis functions to smoothness described in terms of smoothing operators [3].

It was shown in [2] that variation with respect to perceptrons is related to the Sobolev seminorm $\|.\|_{d,1,\infty}$ defined in terms of maxima of partial derivatives as

$$\|f\|_{d,1,\infty} = \max_{|\alpha|=d} \|D^\alpha f\|_{\mathcal{L}^1_\lambda(\mathbb{R}^d)},$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index with nonnegative integer components, $D^\alpha = (\partial/\partial x_1)^{\alpha_1} \ldots (\partial/\partial x_d)^{\alpha_d}$ and $|\alpha| = \alpha_1 + \cdots + \alpha_d$.

Variation with respect to perceptrons is bounded from above by the seminorm $\|.\|_{d,1,\infty}$ times the function $k(d)$ of the number of variables d satisfying $k(d) \sim \left(\frac{4\pi}{d}\right)^{1/2} \left(\frac{e}{2\pi}\right)^{d/2}$.

Thus by Theorem 1, for any sample of data z, which can be interpolated by a function h such that the squares of the maxima of the $\mathcal{L}^1_\lambda$-norms of partial derivatives of the order $|\alpha| = d$ do not exceed an exponentially increasing upper bound $\frac{d}{4\pi} 2^d$, more precisely

$$\|h\|^2_{d,1,\infty} = \max_{|\alpha|=d} \|D^\alpha f\|^2_{\mathcal{L}^1_\lambda(\mathbb{R}^d)} \preceq \frac{d}{4\pi} \left(\frac{2\pi}{e}\right)^d < \frac{d}{4\pi} 2^d ,$$

the minima of the empirical error $\mathcal{E}_z$ over networks with n sigmoidal perceptrons decrease to zero rather quickly – at least as fast as $\frac{1}{n}$.

Variation with respect to Gaussian radial units is related to Bessel norms, which define smoothness using smoothing operators. The operators are of the form of convolutions with certain special functions called Bessel potentials, which are defined by means of their Fourier transforms. For $r > 0$, the *Bessel potential* of order r, denoted by β_r, is the function on $\mathbb{R}^d$ with the Fourier transform

$$\hat{\beta}_r(s) = (1 + \|s\|^2)^{-r/2} .$$

For d a positive integer, $r > d/2$, and $q \in [1, \infty]$, the *Bessel potential space* denoted by $(L^{q,r}, \|.\|_{L^{q,r}})$ is defined as

$$L^{q,r} := \{f \mid f = w * \beta_r,\ w \in \mathcal{L}^q\}$$

$$\|f\|_{L^{q,r}} := \|w\|_{\mathcal{L}^q} \quad \text{for} \quad f = w * \beta_r.$$

It was shown in [3] that for any integer $r > d/2$, variation with respect to Gaussian radial units is bounded from above by the Bessel norm $\|.\|_{L^{1,r}}$ times a function $k(r, d)$ of the number of variables d and the degree r of the Bessel potential, which is defined as $k(r, d) = \frac{(\pi/2)^{d/4} \Gamma(r/2 - d/4)}{\Gamma(r/2)}$, where Γ denotes the Gamma function.

So by Theorem 1, for any sample of data z, which can be interpolated by a function h such that the square of its Bessel norm of some order $r > d/2$ does not exceed $1/k(r, d) = \frac{\Gamma(r/2)}{(\pi/2)^{d/4} \Gamma(r/2 - d/4)}$, the minima of the empirical error $\mathcal{E}_z$ over networks with n sigmoidal perceptrons decrease to zero rather quickly – at least as fast as $\frac{1}{n}$.

So for both perceptron and Gaussian radial units, the tolerance on data complexity guaranteeing fast convergence of error functionals is increasing exponentially fast with dimensionality of input data.

Reference

1. Cucker F. and Smale S., On the Mathematical Foundations of Learning. Bulletin of the AMS 39, 2002, 1–49
2. Kainen P.C., Kůrková V., and Vogt A., A Sobolev-Type upper Bound for Rates of Approximation by Linear Combinations of Plane Waves. Journal of Approximation Theory (to appear)
3. Kainen P.C., Kůrková V., and Sanguineti M., Estimates of Approximation Rates by Gaussian Radial-Basis Functions. In Adaptive and Natural Computing Algorithms - ICANNGA'07, B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro (eds), Part II, LNCS 4432, Berlin, Heidelberg, Springer-Verlag, 2007, 11–18
4. Kůrková V., Estimates of Data Complexity in Neural-Network Learning. In SOFSEM 2007, LNCS 4362, J. Leeuwen, G.F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plil (eds), Berlin, Heidelberg: Springer-Verlag, 2007, 377–387
5. Kůrková, V., Minimization of Error Functionals over Perceptron Networks. Neural Computation (to appear)

ITAT'07

Information Technology – Applications and Theory

WORK IN PROGRESS

Dynamická inovácia zabezpečenej komunikácie medzi komponentami distribuovaných informačných systémov*

Miroslav Beličák

Technická univerzita v Košiciach, Letná 9, Košice 041 20, Slovensko
Miroslav.Belicak@gmail.com
http://www.fei.tuke.sk/

Abstrakt *Príspevok popisuje základné princípy mechanizmu pre dynamické modifikovanie zabezpečenia komunikácie medzi uzlami distribuovaných informačných systémov (DIS). Implementácia navrhovaného mechanizmu umožní počas vykonávania programu inovovať resp. upravovať aplikovanie kryptovacích algoritmov v DIS. Popritom nebude nutné pozastaviť vykonávanie DIS, resp. rekompilovať zdrojový kód jeho určitých súčastí. Túto vlastnosť je možné využiť v prípade, že je potrebné zabezpečiť komunikáciu využitím viacerých kryptovacích algoritmov, resp. niekoľkonásobného zakryptovania odosielaných údajov, resp. v prípade, že máme k dispozícií implementáciu kryptovacieho algoritmu ktorý ešte v DIS nebol aplikovaný.*

1 Úvod

Pri distribuovaných informačných systémoch (DIS) resp. distribuovaných komponentoch informačných systémoch (DCIS) zohráva zabezpečenie vzdialenej komunikácie jednu z najdôležitejších úloh, nakoľko prenášané údaje môžu obsahovať citlivé informácie rôzneho druhu [9]. Množstvo útokov je namierených práve voči prenášaným údajom po sieti medzi týmito komponentami, či už z cieľom impersonovať jednu z komunikujúcich strán, resp. prezerať ich alebo modifikovať a následne preposlať ďalej. Zabezpečená komunikácia (*secured communication*) nám poskytuje dve vlastnosti [6]:

- *súkromie (privacy)* - údaje nebude možné počas prenosu prezerať
- *integritu (integrity)* - údaje ostanú počas prenosu nezmodifikované

Na tomto mieste nebudeme podrobne uvádzať jednotlivé typy kryptovacích algoritmov, ako aj základné definície z oblasti informačnej bezpečnosti, ale sa obmedíme len na tie, na ktoré sa budeme odvolávať v nasledujúcich častiach článku.

Kryptografia (*Cryptography*) [1, 3, 8] - vedná disciplína, ktorá sa venuje ochrane prenosu údajov po nezabezpečenom komunikačnom kanáli[1] alebo ich uskladneniu v nezabezpečenom údajovom repozitári. Svojími prostriedkami zaisťuje, aby prenášané údaje boli zrozumiteľné iba korektne participujúcim stranám, a aby po svojom prenose ostali konzistentné. Jej základnými prostriedkami sú **kryptovacie algoritmy** (*cryptographic algorithms* - CA) [1, 2, 3, 11, 12], ktoré modifikujú (kryptujú/dekryptujú) vstupný prúd údajov na základe určitého **kľúča** resp. kľúčov, ktorým najčastejšie býva krátka množina údajov (rádovo v bitoch). CA delíme na dve základné typy:

- *symetrické (symmetric)* - kryptujú aj dekryptujú vstupné údaje na základe toho istého kľúča (napr. Rijndael, RS2, Mars, RC2, RC6, Serpent, Twofish, Blowfish, a pod.).
- *asymetrické (asymmetric)* - kryptujú aj dekryptujú vstupné údaje pomocou rôznych kľúčov[2] (napr. RSA).

Vprípade, že sa tretej strane podarí odchytiť prenášané údaje, ich dekryptovanie do zrozumiteľného stavu môže zabrať neúmerné množstvo času[3]. Príspevok sa venuje vylepšeniu kryptovania údajov, nakoľko ochrana prenášaných údajov závisí vo veľkej miere od kvality kryptovacích algoritmov a metodík.

2 Súčasný stav a motivácia

V súčasnosti existuje množstvo hotových softvérových implementácií rôznych kryptovacích algoritmov pre rozličné platformy (napr. .NET). Problémom je však zvládnuť potrebné zmeny v DIS, resp. DCIS týkajúce

* VEGA 1/2176/05 - Technológie podporujúce životný cyklus distribuovaných systémov na báze agentov a komponentov. VEGA 1/3135/06 - Metódy a prostriedky pre tvorbu integrovaných distribuovaných aplikácií na báze ambientov - agentov vyššej úrovne.

[1] Prenos zakryptovaných údajov medzi dvoma komunikujúcimi uzlami pozostáva z troch základných častí: *zakryptovanie* (použitím vhodného kryptovacieho algoritmu) → *prenos po nezabezpečenom kanáli* → *dekryptovanie*.

[2] Obyčajne sa pre ne generuje pár kľúčov - v ktorom jeden kľúč je určený pre kryptovanie (tzv. *verejný kľúč - public key*) a jeden pre dekryptovanie (tzv. *súkromný kľúč - private key*).

[3] Väčšinou to závisí najmä od typu CA, dĺžky jeho kľúča v neposlednom rade výpočtového výkonu počítačového systému, ktorým útočník disponuje.

sa používaných algoritmov. Ak sú totiž - pre zabezpečenie komunikácie - kódované takpovediac "napevno ", potrebné je prerábať značne rozsiahle časti kódu (a na mnohých miestach), príp. upravovať tabuľky bázy údajov obsahujúcej súkromné a verejné kľúče. Nasleduje opätovná rekompilácia, testovanie, nasadenie do prevádzky a pod. žiadúce je, aby potrebné zmeny bolo možné previesť plynule a na všetkých potrebných súčastiach DIS resp. DCIS a aby činnosť samotného systému nebolo potrebné pozastaviť (čo by ináč mohli nepríjemne pocítiť jeho používatelia).

V súčasnosti existuje určité množstvo rôznych architektúr a frameworkov pre zabezpečenie softvérových systémov resp. vzdialenej komunikácie. Niektoré sú realizované hardvérovo, iné softvérovo. Spomeňme aspoň *bezpečnostný framework pre P2P Grid-systémy* [4], alebo *Adaptovateľný bezpečnostný framework pre Service-based systémy* [13].

Ako už bolo spomenuté, pri DIS alebo DCIS je potrebné zabezpečiť komunikáciu medzi všetkými vzdialenými uzlami. Pri väčšom počte uzlov berieme do úvahy nasledujúce činnosti spojené so zabezpečením komumunikácie:

• **ujednotenie** - medzi viac ako dvoma vzdialenými uzlami DIS môže byť v niektorých situáciách potrebné ujednotiť zabezpečenie komunikácie, t.j. aby používali rovnaké kryptovacie/dekryptovacie mechanizmy. To je potrebné napr. v prípade, že sa pri škálovaní systému rozšíri počet komunikujúcich uzlov a ak jednotlivé dvojice resp. určité skupiny kryptujú údaje jedným CA a iné dvojice resp. skupiny iným. Tým dochádza ku strate výkonu a k možnosti vzniku bezpečnostných "dier "v dôsledku nesúrodých zabezpečovacích mechanizmov.

• **aktualizovanie** implementácií kryptovacích algoritmov - DIS resp. DCIS musí byť schopný aktualizácie dielčích implementácií umožňujúcich použitie CA. Napr. v prípade novej verzie CA, ktorá má dlhší kľúč, resp. upravený kryptovací algoritmus, musí byť k dispozícií aktualizačný mechanizmus, ktorý uzlu aplikácie umožní plynulý prechod na používanie vyššej verzie CA.

• **vynútenie** používania určitého typu CA pre niektoré spojenia. Občas sa môže vyskytnúť prípad, kedy potrebujeme určitej skupine uzlov "vnútiť "používanie určitého CA pre zabezpečenie komunikácie[4].

[4] Ku tomu môže dôjsť napr. v prípade, že máme napr. medzi dvoma uzlami k dispozícií použitie dvoch CA. Použitie bezpečnejšieho CA s dlhším kľúčom je časovo náročné a keďže sa po komunikačnom kanáli posielajú obvykle informácie štandardného typu, používa sa rýchlejší, avšak menej bezpečný algoritmus. V prípade, že sa budú odosielať citlivé údaje, je vhodné komunikujúce strany "prinútiť"použiť bezpečnejší CA.

• **schopnosť použitia nového CA** - otvorenosť pre použitie implementácií nových CA. V podstate sa jedná o škálovateľnosť, kedy sa implementácie nových CA (napr. vo forme určitých komponentov[5]) dajú jednoducho použiť v komunikujúcich uzloch DIS resp. DCIS. Bolo by vhodné, keby sa správali ako určité zasúvateľné *plug-in moduly*.

• možnosť **flexibilného** používania implementácií CA

• **využívať preddefinované mechanizmy** - bolo by žiadúce mať k dispozícií napr. vysokoúrovňový mechanizmus pre výmenu kľúča symetrického CA pomocou asymetrického CA[6].

3 Návrhované riešenie

Náš návrh sa pokúša vyriešiť vyššie uvedené problémy. Riešenie spočíva vo forme špeciálneho servra vytvoreného ako štandardná windows-služba (*windows-service*). Jeho názov je *Implementations of Cryptographic Algorithms server* (*ICA server*) a jeho konceptuálny návrh a použitie je zobrazené na obrázku 1. Jeho použitie a správanie by sa malo v budúcnosti čo najviac približovať štandardným SQL servrom, teda aby príjimal dotazy od svojich klientov (napr. údaje ktoré treba zakryptovať + ako ich treba zakryptovať/dekryptovať) a vracal požadované údaje (napr. zakryptované/dekryptované údaje). Predpokladáme, že takýmto spôsobom by bolo možné zjednodušiť zabezpečený návrh DIS a adaptovať zmeny v zabezpečenej komunikácií bez nutného pozastavenia DIS a dočasného odpojenia jeho používateľov.

Server bude umožňovať:

• **štandardné** kryptovanie/dekryptovanie použitím určitého CA.

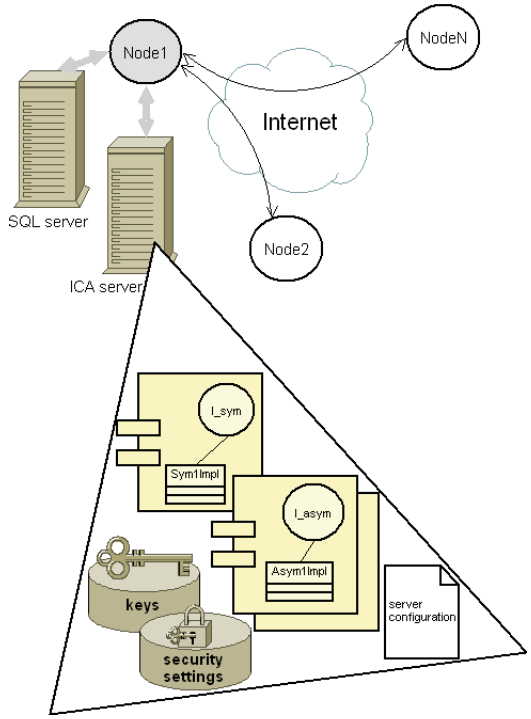
• **kombinované** kryptovanie použitím viecerých symetrických resp. asymetrických CA.

• **viacnásobné** prekryptovanie použitím toho istého CA.

Ďalej bude možné vykonávať kombinované kryptovanie využitím symetrických CA spolu s asymetrickými – **zreťazené kryptovanie**, **kombinované viacnásobné** kryptovanie a iné. V porovnaní s SQL servrom prvotná verzia ICA servra neobsahuje databázy ale komponenty implementujúce jednotlivé symetrické/asymetrické CA, ktoré vyhovujú presne definovanému rozhraniu, ktorému zasa "rozumejú"objekty klientskej aplikácie na konkrétnom uzly DIS resp.

[5] To však predpokladá podporu vopred stanoveného komunikačného rozhrania.

[6] Jedná sa o štandardný spôsob, kedy sa kľúč symetrického CA prenáša po komunikačnom kanále v zakryptovanom stave pomocou verejného kľúča jeho príjemcu používajúceho asymetrický CA.



Obrázok 1. Použitie ICA servra v distribuovaných prostrediach.

DCIS. Samozrejme, okrem toho si udržiava údajový sklad obsahujúci kľúče používané jeho používateľmi, bezpečnostné nastavenia (napr. spôsob autentifikácie a autorizácie jeho klientov), a celkovú konfiguráciu. Nad tým všetkým je *Query Processing* (QP), čo je analógia DBMS (*DataBase Management System*), ktorý spracuváva jednotlivé dotazy od používateľa.

3.1 Dotazovací jazyk pre ICA server

V tejto sekcii budú uvedené príklady použitia dotazovacieho jazyka pre ICA server (*Ica Query Language* - IcaQL) pre platformu Microsoft .NET. Tento jazyk slúži na komunikáciu medzi ICA servrom a uzlom DIS resp. DCIS, ktorý je voči nemu v roli klienta (analógia dotazovacieho jazyka SQL). Nasledujúci blok kódu[7] nám zobrazuje pripojenie sa ku servru, vytvorenie a odoslanie dotazu, prijatie odpovede a ukončenie spojenia.

```
string connectionString ="address:localhost;
port:54722; login:'myLogin';
passwd='myPasswd'";
Ica.IIcaConnection con =
new Ica.IcaConnector(connectionString);
string query = "myquery @par1";
Ica.IcaQLCommand cmd =
new IcaQLCommand(con, query);
Ica.IIcaParameter ipar
= new Ica.IcaParameter();
ipar.ParameterName = "@par1";
ipar.Value = formalParameter1;
ipar.DbType = System.Data.DbType.String;
cmd.Parameters.Add(ipar);
con.OpenConnection();
System.Data.IDataReader reader =
cmd.ExecuteCommand();
// returns standard .NET data reader
...
cmd.Close();
```

Priklady hodnoty *myQuery* môžu byť podľa požadovanej činnosti nasledujúce:

• **jednoduché** kryptovanie/dekryptovanie použitím symetrického algoritmu:

```
ENCRYPT DATA=@data USING _SYM=
('algName', KEY=(AUTOGENERATE, PATH=@path));
DECRYPT DATA=@data USING _SYM=
('algName', KEY=(PATH=@path));
```

• **jednoduché kombinované** kryptovanie využitím dvoch symetrických algoritmov používajúcich rovnaký kľúč:

```
ENCRYPT DATA=@data USING CHAIN=
{_SYM=(' algName', KEY=(PATH=@path));
_SYM=(' algName2', KEY=(PATH=@path))};
```

• **viacnásobné** kryptovanie použitím symetrického algoritmu:

```
ENCRYPT DATA=@data USING LOOP=
{COUNTER=3;_SYM=('algName',
KEY=(PATH= @path))};
```

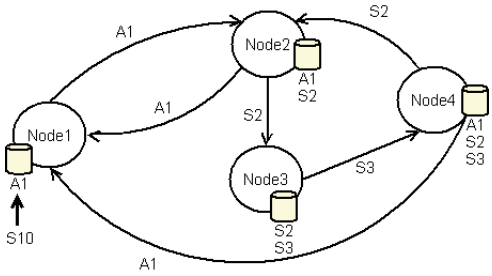
Element DATA predstavuje vstupné údaje[8] určené na kryptovanie/dekryptovanie, _SYM indikuje, že sa jedná o symetrický CA. KEY sa týka kľúča CA, a CHAIN definuje, že sa jedná o zreťazenie.

3.2 Prípadová štúdia

Predpokladajme, že máme štyri vzdialené uzly, ktoré medzi sebou komunikujú po nazabezpečenom kanáli použitím symetrických a asymetrických CA (obr. 2). Symetrické CA budeme označovať skratkou *Sx* a asymetrické *Ax*, kde $x = \{1,2,\ldots\}$. Jednotlivé čísla označujú jednotlivé typy CA (napr. *S1* bude odpovedať

[7] Nápadne pripomína odoslanie dotazu pre určitý SQL server. Aj tu je možné vidieť podobnosť medzi ICA servrom a SQL servrom.

[8] Vstupným údajom (*@data*) bude môcť byť aj *BLOB* (*Binary Long OBject*).



Obrázok 2. Komunikujúce uzly DIS resp. DCIS (prípadová štúdia).

RC5, *S2* Rijndael a pod.). Šípky zobrazujú smer odosielania informácie a nad nimi je uvedený aj druh CA použitý pre zabezpečenie. Pod údajovými skladmi pri jednotlivých uzloch sú uvedené názvy implementácií algoritmov, ktoré sú v nich uložené a ktorými tenktorý uzol disponuje.

Pri použití ICA servra jednotlivými uzlami DIS musí byť medzi nimi (minimálne medzi dvoma) určitá hierarchia, popisujúca dominantné postavenie jedného uzla a musí byť akceptovaná všetkými zúčastnenými uzlami. Táto hierarchia určuje, ktorý z uzlov bude mať vedúce postavenie a teda bude určovať spôsob zabezpečenia komunikácie medzi ostatnými uzlami[9] (a môže byť definovaná napr. jedinečnou celočíselnou hodnotou pre každý uzol). Predpokladajme, že uzol *Node1* má voči uzlom *Node2,3,4* dominantné postavenie a chce zaviesť používanie symetrického CA (*S10*) medzi všetkými uzlami (čo je jedna z možností využitia ICA servra). V pilotnej verzii ICA servra je postup jednoduchý a pojednáva o čom nasledujúca podkapitola.

Aplikovanie implementácie nového CA Keďže ICA server väčšinu svojej funkcionality ponúka vo forme služieb (od prijatia dotazu cez jeho analýzu, vykonanie požadovanej činnosti až po zaslanie odpovede) pre implementáciu jeho funkcionality sme sa rozhodli použiť technológiu prichádzajúcu s Windows Vista a .NET Framework 3.0 – *Windows Workflow Foundation* (WWF). Jedná sa o architektúru podporujúcu tvorbu systémov na báze pracovného toku (*workflow*). Detailnejšie informácie o WWF možno nájsť v [5, 7, 10].

Koncept riešenia pre aplikovanie implementácie nového CA pre všetky uzly DIS resp. všetky DCIS bude nasledujúci:

Odoslanie informatívneho tokenu[10] *ostatným uzlom v skupine → serializácia implementácie CA → odoslanie implementácie ostatným uzlom → čakanie na potvrdzujúce tokeny → čakanie na príchod tokenu potvrdzujúceho, že je už možné začať komunikovať použitím nového CA.*

Uvedený koncept riešenia vytvoreného pomocou technológie WWF je zobrazený na obrázku (obr. 3.), kde vyplnené elipsy predstavujú tzv. *aktivity* (*activities*) za ktorými sa ukrýva kód, nevyplnené elipsy predstavujú aktivity ktoré sa vykonajú až potom, čo nastanú určité *udalosti* (*events*) (napr. po prijatí všetkých informačných tokenov zaslaných zúčastnenými uzlami). Kosoštvorec predstavuje *vetvenie*. Za povšimnutie stojí skutočnosť, že hoci jeden uzol má vedúce postavenie, riešenie je tu rozšírené tak, že komunikujúce uzly môžu na určitý čas odmietnuť prijatie implementácie nového algoritmu napr. z dôvodu veľkej vyťaženosti, alebo nestáleho spojenia počas určitej periodicky sa opakujúcej doby (o ktorej nadradený uzol nemusí mať znalosti[11], najmä ak sa DIS resp. DCIS nedávno rozšíril o nové komunikujúce uzly a nadradený uzol je jedným z nich).

Algoritmus pokrývajúci činnosť prvých dvoch aktivít je zobrazený na obrázku 4., kde informačný token je vo forme tabuľky do ktorej zapisujú všetky zúčastnené uzly či príjimajú/neprijímajú novú implementáciu. Ich odpovede sa šíria vše-smerovo, až doputujú naspäť do nadradeného-dominantného uzla (iniciátora komunikácie). Obrázok 4. zobrazuje celý proces od odoslania požiadavky iniciátorom, až po prijatie výsledkov komunikácie.

4 Záver a nasledujúce etapy práce

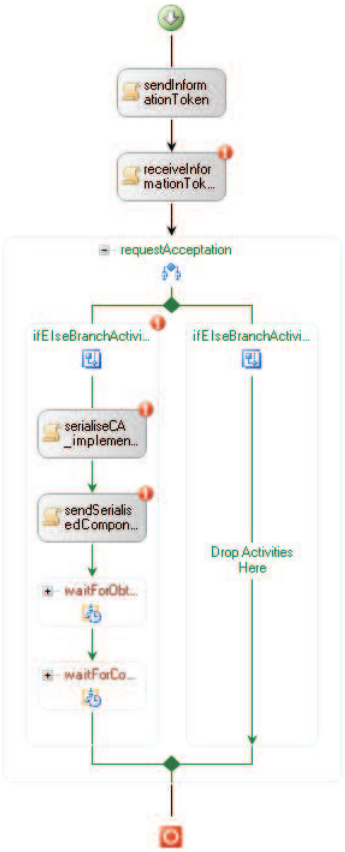
Flexibilný prístup pre zabezbečenie komunikácie môže výrazne uľahčiť prácu návrhárom DIS. Navyše, v prípade, že za kryptovanie prenášaných údajov zodpovedá samostatný systém, škálovateľnosť a modifikovateľnosť použitia CA sa stáva výrazne jednoduchšou.

V súčasnosti už máme vytvorené komunikačné kanály medzi servrom a jeho klientami. Pre úspešné ukončenie práce na ICA servri bude potrebné ešte zabezpečiť komunikáciu medzi klientom servra a samotným ICA servrom (tento zabezpečovací mechanizmus je momentálne v polohe návrhu, podobne je tomu aj pri samotných službách servra). Ďalej bude potrebné

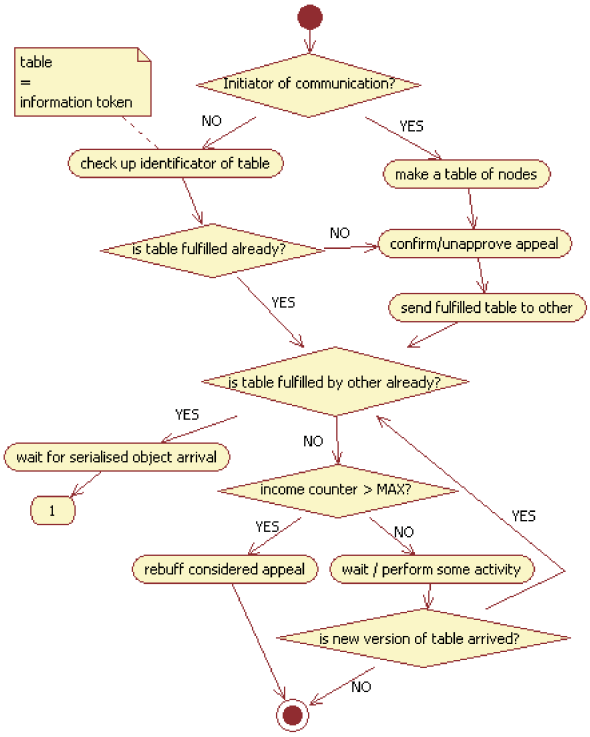
[9] V prípade, že napr. zlyhá sieťové spojenie medzi vedúcim uzlom a ostatnými uzlami, vedúce postavenie sa prenesie na nasledujúci uzol v poradí podľa hierarchie.

[10] Objekt, ktorý je odoslaný zúčastneným uzlom a obsahuje rôzne informácie, napr. že dôjde ku určitej zmene v zabezpečení komunikácie. Uzly môžu z neho čítať aj zapisovať (napr. že ju príjimajú/neprijímajú).

[11] Tu sa črtá možnosť nasadenia určitého znalostného podsystému ako súčasť ICA servra, ktorý by udržiaval rôzne profily všetkých komunikujúcich uzlov.



Obrázok 3. Aplikovanie nového CA pomocou technológie WWF.



Obrázok 4. Postup pri odosielaní a prijatí informačného tokenu pre zúčastnené uzly.

došpecifikovať dotazovací jazyk IcaQL. Na záver by sme ešte chceli do servra a jeho klientov zapracovať meranie času medzi jednotlivými odozvami (medzi jednotlivými uzlami a medzi klientami servra a serverom samotným).

Momentálne ešte nedoriešenou otázkou ostáva, či pre uschovávanie kľúčov klientov servra a iných citlivých informácií bude ICA server spolupracovať s SQL servrom, resp. bude používať vlastné udaj. sklady.

Referencie

1. Breveglieri L. and Maistri P., An Operation-Centered Approach to Fault Detection in Symmetric Cryptography Ciphers. IEEE Transactions on Computers, 56, 5, 2007
2. Dongara P. and Vijaykumar T.N., Accelerating Private-Key Cryptography via Multithreading on Symmetric Multiprocessors. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'03), IEEE, 2003, 58–69
3. Elbirt A. and Paar Ch., Instruction-Level Distributed Processing for Symmetric-Key Cryptography. In Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), IEEE, 2003, 78b
4. Ellahi T.N., Hudzia B., Mcdermott L., and Kechadi T., Security Framework for P2P Based Grid Systems, In Proc. of the Fifth International Symposium on Parallel and Distributed Computing (ISPDC'06), IEEE, 2006, 230–237
5. Esposito D., Getting Started with Microsoft Windows Workflow Foudation: A Developer Walktrough, Solid Quality Learning, `http://msdn.microsoft.com/windowsvista/building/workflow`, 2005
6. Microsoft: Building Secure ASP.NET Applications, Authentication, Authorization, Secure Communication, Microsoft Corporation, version 1.0, 2002
7. Myers B.R., Foundation of WF, ISBN 1590597184, 2006
8. Nam H.Ch., Kim J.Y., Hong S. J.Y., and Lee S., A Secure Checkpointing System. In Proc. of the 2001 Pacific Rim International Symposium on Dependable Computing (PRDC.01), IEEE, 2001, 49

9. Salomão S.L.C., de Alcântara J.M.S., Alves V.C., and Vieira A.C.C., SCOB, a Soft-Core for the Blowfish Cryptographic Algorithm, XII Brazilian Symposium on Integrated Circuits and Systems Design, IEEE, 1999, 0220
10. Shukla D. and Schmidt B., Essential Windows Workflow Foundation, Addison-Wesley Proffesional, ISBN 0321399838, 2006
11. Schneier B., Applied Cryptography, John Wiley and Sons, New York, 1996
12. Wu L., Weaver Ch., and Austin T., CryptoManiac: A Fast Flexible Architecture for Secure Communication. In proc. of the 28th Annual International Symposium on Computer Architecture (ISCA'01), IEEE, 2001, 0110
13. Yau S.S., Yao Y., Chen Z., and Zhu L., An Adaptable Security Framework for Service-based Systems. In proc. of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05), IEEE, 2005, 28–35

Ochranné proxy-webové služby*

Miroslav Beličák

Technická univerzita v Košiciach, Letná 9, Košice 041 20, Slovensko
Miroslav.Belicak@gmail.com
http://www.fei.tuke.sk/

Abstrakt *Príspevok popisuje autentifikačný a autorizačný mechanizmus založený na štandardných webových službách. Jeho princíp spočíva v dvoch a viacerých štandardných webových službách, kde jedna z nich poskytuje používateľovi autentifikáciu a autorizáciu, ostatné sú tzv. proxy-webové služby, využívané ako "brány"pre prístup ku webovým službám poskytujúcim požadovanú funkcionalitu. Proxy-webové služby môžu obsahovať okrem zapezpečovacích mechanizmov, aj mechanizmy napr. na profilovanie výkonu webových služieb (resp. architektúry orientovanej na služby). Hlavná výhoda spočíva v jednotnom zabezpečenom prístupe ku všetkým webovým službám organizácie, vývoji webových služieb bez nutnosti integrácie bezpečnosti do procesu vývoja a schopnosti vyhnúť sa rekompilácií proxy-objektov klientských aplikácií v prípade istých zmien sa strane poskytovateľa služieb.*

1 Úvod a súčasný stav

Zabezpečenie prístupu ku firemným údajom je jedna z najdôležitejších výziev ako pre návrhárov, tak aj pre vývojárov a administrátorov. Takáto bezpečnosť sa týka sprístupnenia údajov ako vo vnútri podniku (pre úzky okruh používateľov), tak aj zvonku (pre širší okruh autorizovaných používateľov). Navrhované riešenie sa zaoberá tým druhým problémom - poskytnutia údajov naprieč hraníc podniku resp. organizácie pre určitú skupinu obchodných partnerov (vo všeobecnosti používateľov). V úvode spomenieme webové služby nakoľko sú predmetom záujmu, ako aj architektúru orientovanú na služby, nakoľko uvedené riešenie sa v určitom prípade môže s ňou prekrývať z architektonického aspektu.

1.1 Webové služby

Webové služby sa stali fenoménom súčasneho sveta informačných technológií a používajú sa v značne širokej oblasti, najmä v B2B aplikáciách. Vo všeobecnosti je môžné webové služby *(web services)* [1, 4, 6, 7] zadefinovať ako sebestačné *(self-contained)*, modulárne aplikácie dostupné prostredníctvom web-u, ktoré poskytujú množinu funkcionalít pre podniky (resp. firmy) alebo jednotlivcov.

Organizácia World Wide Web Consorcium (W3C) Web Services Architecture Working Group definuje webovú službu ako " *softvérový systém navrhnutý pre podporu spolupráce machine-to-machine interakcie prostredníctvom siete. Rozhranie má popísané vo formáte spracovateľnom strojom (špecificky WSDL - Web Service Description Language). Ostatné systémy interagujú s webovou službou spôsobom predpísaným jej popisom použitím SOAP-správ. Tie sú typicky dopravované použitím HTTP*[1] *protokolu spolu s XML serializáciou a v konjunkcii s ostatnými webovo súvisiacimi štandardami.*" [12].

Webové služby kombinujú silu dvoch rozšírených technológií: XML - univerzálny popisovací jazyk pre údaje a prenosový protokol HTTP, široko-podporovaný webovými prehliadačmi a webovými službami. Na webové služby sa môžeme dívať nasledovne:

Webové služby = XML + prenosový protokol

Kľúčovými vlastnosťami webových služieb sú: *sebestačnosť, sebapopísanie, modularita, možnosť publikovania, lokalizovania a vyvolania cez sieť.*

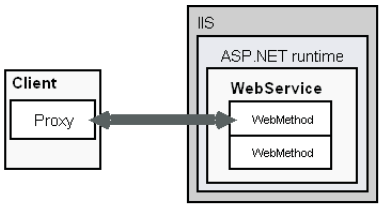
Klientské aplikácie môžu komunikovať s webovou službou napr. prostredníctvom tzv. proxy-objektu[2]. Situácia je zobrazená na obrázku obr. 1. Kedže sa jedná o .NET platformu, webová služba bude vykonávaná vo vykonávacom prostredí ASP .NET[3] v kontexte internetovej informačnej služby (*Internet Information Service - IIS*) [13]. Klient pristupuje k webovej metóde pomocou proxy-objektu, ktorý volá webovú metódu na strane webovej služby. Obidve metódy majú rovnaký názov, návratovú hodnotu a formálne parametre. Jej volanie môže byť *synchrónne* resp. *asynchrónne.*

* VEGA 1/2176/05 – Technológie podporujúce životný cyklus distribuovaných systémov na báze agentov a komponentov. VEGA 1/3135/06 - Metódy a prostriedky pre tvorbu integrovaných distribuovaných aplikácií na báze ambientov - agentov vyššej úrovne.

[1] HTTP - HyperText Transfer Protocol, prenosový protokol pre prenos údajov v internete.

[2] Napr. na platforme Microsoft .NET

[3] ASP - *Active Server Pages*, dynamické webové stránky, na pozadí ktorých sa na strane servra vykonáva určitý kód. Technológia spoločnosti Microsoft.



Obrázok 1. Komunikácia medzi klientom a webovou službou na platforme .NET.

1.2 Architektúra orientovaná služby

Architektúra orientovaná na služby (*Service-Oriented Architecture* – SOA) [1, 4, 5, 9] je novou výpočtovou paradigmou a paradigmou vývoja softvéru, v ktorej softvéroví vývojári sú zoskupení do troch skupín vzhľadom na ich zodpovednosti [10]:

• **tvorcovia aplikácie** (*application builders*) resp. **žiadatelia služby** (*service requesters*),

• **makléri služby** (*service brokers*),

• **poskytovatelia služby** (*service providers*).

Poskytovatelia služieb vyvýjajú služby nezávislo od potenciálnych aplikácií podľa otvorených protokolov a štandardov. Makléri služby publikujú dostupné služby verejnosti. Budovatelia aplikácie vyhľádávajú požadované služby a komponujú cieľovú aplikáciu použitím dostupných služieb. Takto je cieľová aplikácia vybudovaná prostredníctvom objavenia služieb a ich komponovania, namiesto tradičného procesu návrhu a kódovania softvéru.

Za služby - stavebné kamene pre SOA -môžeme použiť práve webové služby.

2 Motivácia

Motiváciu za touto prácou možno zhrnúť do nasledujúcich bodov:

• **potreba mať predpripravené, zabezpečené časti informačného systému (IS)**. Týmto sa zjednodušuje návrh a urýchľuje vývoj systému. Takto návrhár nemusí zohľadnovať možné bezpečnostné riziká, nakoľko za nich nesie zodpovednosť určitý komponent resp. framework. Teda *zjednodušuje* sa *návrh, implementácia a šetria sa finančné a časové prostriedky.*

• vzhľadom na skutočnosť, že webové služby sú v súčasnosti široko využívané, jedným z účelov je aj **zjednodušiť návrh a tvorbu IS na báze web. služieb**, najmä pre menšie organizácie resp. podniky.

• **možnosť dodatočného zabezpečenia** hotového IS na báze webových služieb (napr. pre malé organizácie) bez nutnosti prerábania väčšieho objemu návrhu a implementácie (s čím úzko súvisia napr. finančné prostriedky).

• **vytvorenie flexibilného a škálovateľného bezpečnostného mechanizmu** pre identifikáciu, autentifikáciu a autorizáciu klientov webových služieb. *V prípade neskorších dodatočných úprav spôsobu autentifikácie a autorizácie je potrebné vykonať zmeny len v bezpečnostnom mechanizme a nie na mnohých miestach rozsiahlého zdrojového kódu aplikácie.*

• vyskytli sa prípady útokov proti používateľom bankových IS spôsobom falzifikácie - vytvorili sa **kópie oficiálnych stránok** tej-ktorej banky. Používateľ, ktorý si ich ľahko pomýlil[4] s oficiálnymi stránkami do nich vyplnil svoje údaje (potrebné napr. pre manipuláciu so svojím kontom). Falošné webové stránky mu odovzdajú odpoveď napr. o úspešnom bankovom prevode zatiaľ čo zadané údaje sú postúpené útočníkovi, ktorý ich môže použiť pre manipuláciu s kontom nič-netušiaceho používateľa na skutočnom bankovom servri. Takýto typ útoku je možný nie len pri takýchto typoch IS, ale aj pri iných, najmä na báze webových aplikácií resp. aplikáciách podporujúcich aj webové používateľské rozhrania, kde *falošný webový IS predstiera správanie sa skutočného IS a to na podobnej webovej adrese.*

Vlastnosť vyššie spomenutého útoku by sa však mohla využiť aj ako obranný mechanizmus, kde voči skutočnej funkcionalite IS by bol "predsunutý" určitý "obranný štít", ktorý by napr. monitoroval používateľov, detekoval by útoky, profiloval výkon a pod. To všetko bez toho, aby používateľ vedel, že nepracuje so skutočným IS resp. jeho komponentom, ale s takým, ktorý má len jeho výzor resp. rovnaké rozhranie a kolaboruje so skutočným IS resp. komponentom a tým aj poskytuje jeho funkcionalitu[5].

Predpokladáme, že takýmto spôsobom ochrany nebude potrebné prerábať resp. zdokonaľovať bezpečnosť v už vytvorených častiach a komponentách IS, ale postačí pred každú z nich predsunúž "štíty". V nich by bolo možné zabezpečiť reálny komponent a príp. uplatniť iné podporné mechanizmy, ktoré by bolo možné škálovať a vylepšovať.

[4] Napr. ak používa e-banking zriedkavo, resp. prvý krát a oficiálne stránky si vyhľadá napr. na webe. Kežže aj názov webovej adresy môže byť veľmi podobný oficiálnemu (a ten presne nepozná) ľahko dôjde k pomýleniu.

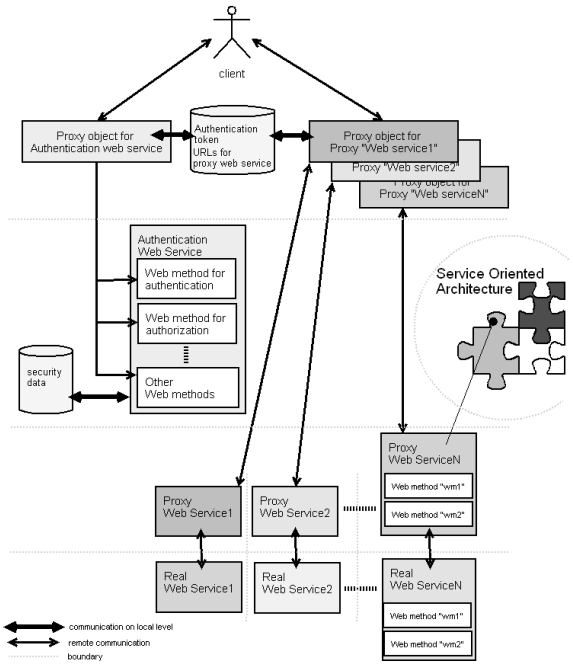
[5] Tento prístup môže mať vhodné využitie napr. v oblasti webových služieb, kde môže dôjsť k útoku na webovú službu jej zahltením rôznymi žiadosťami za účelom redukcie výkonu IS a teda spomalenia odozvy pre jeho používateľov.

3 Návrh

Základný koncept nášho riešenia, ktoré by vyhovovalo vyššie uvedeným požiadavkám je zobrazený na obrázku 2. Hlavnými stavebnými kameňmi v tomto riešení sú:

- *Autentifikačná webová služba* (AuWS)
- *Skupina proxy-webových služieb* (PxWS)

Ako už bolo naznačené, potrebná je identifikácia, autentifikácia a autorizácia klienta tej-ktorej webovej služby organizácie. V našom riešení sa o všetky tieto tri činnosti stará práve AuWS. AuWS je súčasne mechanizmom pre prístup ku tzv. *proxy-webovým službám*. Používateľ pristupuje k AuWS službe pomocou *proxy objektu pre AuWS*. Proxy webová služba je "ochranným štítom" prednastaveným pred webovou službou oganizácie a pre používateľa je prístupná cez *proxy objekt pre proxy-webovú službu*. Taktiež môže byť použitá aj pri kompozícií webových služieb pri tvorbe systémov na báze SOA. AuWS a PxWS so sebou veľmi úzko súvisia. Totiž - prístup ku PxWS bez predchádzajúcej autentifikácie u AuWS nie je možný. Pozrime sa teraz detailnejšie na jednotlivé typy webových služieb riešenia.



Obrázok 2. Základný koncept navrhovaného riešenia

3.1 Autentifikačná webová služba

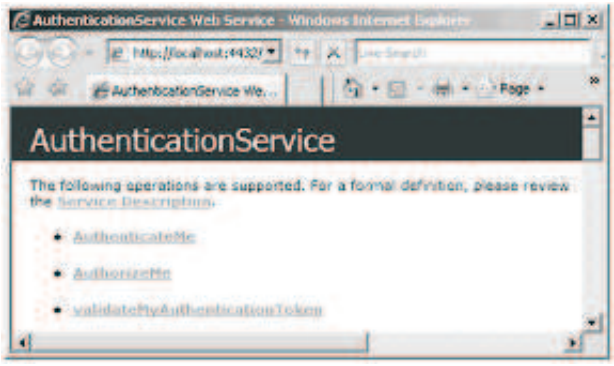
AuWS obsahuje (v súčasnej verzii) tri webové metódy (obr. 3.). Prvou je *AuthenticateMe*, ktorá autentifikuje používateľa na základe jeho *IP adresy*, *názvu stroja* a *mena používateľa*. AuWS využíva údajový sklad nazvaný jednoducho *security data*. V tomto sklade sa nachádza databáza autorizovaných používateľov a ich práv používať tú-ktorú webovú službu organizácie. Táto webová metóda nemá nijaké parametre. Totiž – všetky potrebné údaje o používateľovi, si dokáže extrahovať z HTTP kontextu používateľa (jeho meno, názov stroja spolu s IP adresou). Ak sa používateľ potrebuje autentifikovať, nezadáva nijaké parametre, len jednoducho zavolá túto metódu. V prípade, že sa jedná o korektného používateľa[6], webová metóda *AuthenticateMe* mu zašle tzv. autentifikačný token (AT) – reťazec, ktorý je jedinečný pre každého používateľa a je tvorený hodnotami (jeho meno a pod.), na základe ktorých ho webová metóda autentifikovala v zahešovanom[7] tvare. V opačnom prípade mu bude odoslané chybové hlásenie. Týmto autentifikačným tokenom sa bude autentifikovať voči proxy-webovým službám o ktorých pojednáva nasledujúca podkapitola. Ďalšou webovou metódou je *validateMyAuthenticationToken*. V rámci zvýšenia bezpečnosti je pre používateľa potrebné po určitom časovom intervale svoj AT obnovovať opätovnou autentifikáciou. V opačnom prípade (po uplynutí tohto intervalu) nebude môcť využívať webové služby organizácie. V prípade, že si nie je istý či je už potrebné jeho AT obnoviť, môže využiť túto metódu. Podobne ako *AuthenticateMe*, ani táto metóda nemá nijaké parametre. AT sa totiž po každej úspešnej autentifikácií používateľa zapisuje taktiež do bezpečnostnej databázy údajového skladu *security data*. Do tejto databázy má prístup aj táto metóda[8], ktorá si podobne ako *AuthenticateMe* dokáže všetky potrebné údaje zistiť z používateľovho HTTP kontextu. Po vyvolaní metódy táto vráti používateľovi reťazec s hodnotou "*valid*" v prípade, že jeho token ešte môže používať, v opačnom prípade vráti hodnotu "*invalid*". Posledná metóda nesie názov *AuthorizeMe*. Touto metódou používateľ získava URL adresu požadovanej webovej služby organizácie[9]. Táto metóda má dva parametre – AT a názov požadovanej služby. V prípade, že používateľ má platný AT a požaduje URL webovej služby na ktorú mu organizácia pridelila právo používať ju, metóda mu vráti jej aktuálne URL. V opačnom prípade mu vráti chybové hlásenie. Pred každým volaním požadovanej webovej služby organizácie je vhodné keď si používateľ takto overí

[6] Ktorý je zaevidovaný v bezpečnostnej databáze (v údajovom sklade *security data*.) do ktorej zapisuje prevádzkovateľ webovýsh služieb - organizácia, resp. firma.

[7] Použitá bola hešovacia funkcia MD5.

[8] Podobne ako všetky ostatné webové metódy služby *AuthenticationService*.

[9] V skutočnosti URL adresu proxy-služby.



Obrázok 3. Autentifikačná webová služba AuWS spolu s jej webovými metódami.

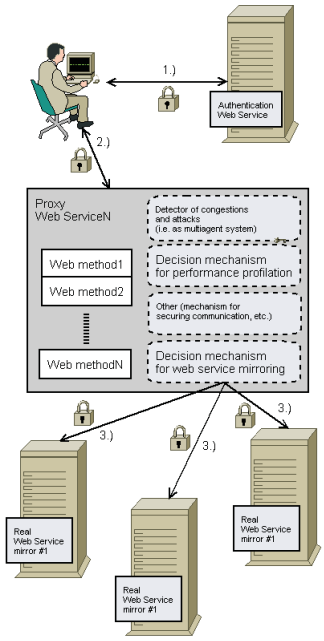
jej aktuálne URL, nakoľko jej prevádzkovateľ ju mohol medzičasom premiestniť na iný server. Postup pre používateľa môžeme teda zhrnúť nasledovne: *Autentifikácia web. metódou AuthenticateMe + odpamätanie si AT → zistenie URL požadovanej služby pomocou jej názvu a AT.*

Keďže vo všetkých prípadoch sa jedná o prenos veľmi citlivých a ľahko zneužiteľných údajov, prenos údajov medzi službou *AuthenticateMe* a jej klientami je potrebné zabezpečiť pomocou protokolu HTTPS (obr. 4. - komunikačný kanál "1.) ").

3.2 Proxy-webová služba

Ako už bolo spomenuté v úvode tejto podkapitoly, PxWS (obr. 4.) v kontexte nášho riešenia je služba, ktorá má rovnaké webové metódy (rovnaký názov, návratová hodnota a parametre) ako požadovaná služba organizácie. Názov PxWS je vhodné taktiež prispôsobiť názvu "skutočnej "webovej služby organizácie. Používateľ takto nevie, že v skutočnosti pracuje PxWS, ktorá chráni web. službu poskytujúcu požadovanú funkcionalitu. PxWS totiž môže byť "vyzbrojená" detektormi, ktoré dokážu rozpoznať, či sa nejedná napr. o *útok zahltením*, resp. nejaký iný. V takom prípade môže PxWS dočasne zablokovať takéto prichádzajúce žiadosti napr. z konkrétnej IP adresy, resp. od konkrétneho používateľa ktorý sa enormne veľa krát pokúša autentifikovať sa (*slovníkový útok*).

Ďalej sa v nej môžu nachádzať mechanizmy pre dynamické dolaďovanie výkonu web. služby organizácie, ktoré úzko súvisia s mechanizmom pre určitý *mirroring* - presmerovávanie žiadostí na iný server obsahujúci tú istú webovú službu za účelom lepšieho rozloženia záťaže na servre a tým aj lepšej odozvy pre používateľov. Totiž PxWS sa nemusí nachádzať spolu s požadovanými službami na tom istom servri. Podobne, nemusí sa ani nachádzať na rovankom servri spolu



Obrázok 4. Detailný pohľad na proxy-webovú službu.

s AuWS. PxWS jednotlivé žiadosti postupuje ďalej - "skutočným "službám organizácie a taktiež od nich príjima výsledky, ktoré potom vracia klientom. Medzi PxWS a skutočnými webovými službami je potrebné zabezpečiť komunikáciu cez protokol HTTPS (komunikačné kanály 3.) na obr. 4.), avšak medzi klientom a PxWS môžeme komunikovať aj cez HTTP (komunikačné kanály 2.), ktorý samozrejme môžeme zabezpečiť analogicky. Popritom však nevylučujeme, že v budúcnosti sa komunikácia môže zabezpečiť aj pomocou kombinácie asymetrických a symetrických kryptovacích algoritmov, nakoľko nie vždy bude možné nakonfigurovať server pre možnosť používania HTTPS.

Útočník, ktorý by sa chcel dostať ku web. službe firmy resp. organizácie musí takto najprv zistiť jej URL adresu (ktorú však nepozná ani korektný používateľ, nakoľko on komunikuje s PxWS). To sa mu môže podariť len po úspešnej autentifikácií AuWS. V prípade, že by sa mu aj podarilo zistiť jej URL, ocitne sa voči PxWS, ktorá v ďalších verziách bude opäť požadovať určitú autentifikáciu. V aktuálnej verzií PxWS tento mechanizmus ešte nie je zabudovaný. Adresa skutočnej webovej služby by však mala ostať známa len jej poskytovateľom a PxWS.

3.3 Prípadová štúdia

Implementáciu riešenia sme odskúšali na príklade jednoduchej kalkulačky vo forme webovej služby. Webovú službu sme pomenovali "*Experimental web service*"a zimplementovali sme jej jedinú webovú metódu "*calculate()*" (Figure 5.). Pri štandardnom volaní postupujeme môžeme postupovať nasledovne:

Pomocou utility *wsdl.exe* (dodávanej spolu s platformou .NET Framework) si vytvoríme proxy objekt pre túto službu, ktorý odkompilujeme do formy dynamicky spájanej knižnice (.dll súbor) pre platformu .NET. Využitím tejto knižnice bude možné webovú službu použiť nasledovne:

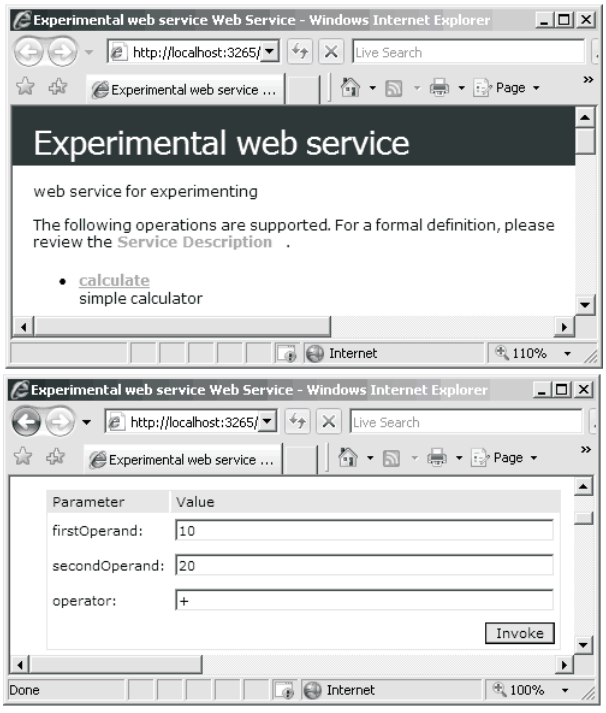
```
...
Experimentalwebservice ews = new
Experimentalwebservice();
...
int Result = ews.calculate(10, 50, '+');
...
```

Teda najprv si vytvoríme inštanciu webovej služby a neskôr môžeme volať jej metódy. V našom prípade sa jedná o volanie jedinej webovej metódy *calculate()*. Táto metóda má tri parametre, z ktorých prvé dva predstavujú celočíselné operandy a posledný predstavuje operátor (+, -, *, /). Návratovou hodnotou je výsledok výpočtu ako celé číslo. V prípade nášho návrhu musíme postupovať nasledovne:

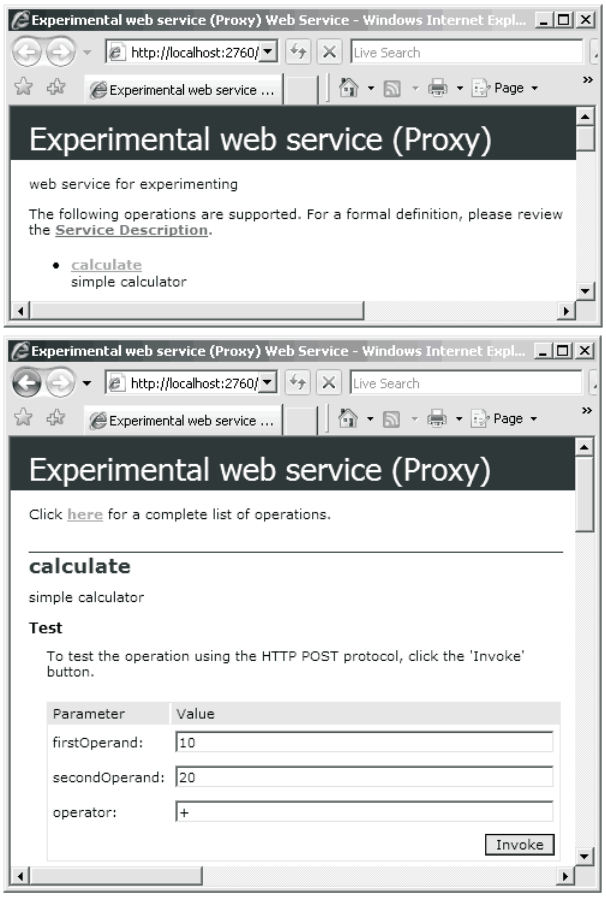
```
...
AuthenticationService asTemp = new
AuthenticationService();
...
string strToken =
(string)asTemp.AuthenticateMe();
...
string serviceUrl =
asTemp.AuthorizeMe(strToken,
"ExperimentalWebService");
...
ExperimentalwebserviceProxy ewsp = new
ExperimentalwebserviceProxy(serviceUrl);

int Result = ewsp.calculate(10, 50, '+');
...
```

Obyčajne je však názov PxWS zhodný s názvom požadovanej web. služby.



Obrázok 5. Web. služba *Experimental web service* (prípadova štúdia).



Obrázok 6. Proxy-webová služba *Experimental web service.*

4 Záver a nasledujuce etapy

Prvotná verzia riešenia je v podstate kompletne hotová. Testovali sme ju na lokálnom počítači bez použitia protokolu HTTPS. Náš experiment sme rozdelili do dvoch kategórií: V prvej sme volali metódu *calculate()* jednoduchej webovej služby (*ExperimetalWebService*). V prvom prípade sa jednalo o vytváranie jej proxy-objektu a volanie web. metódy spolu v jednom cykle, v druhom sa najprv vytvorila inštancia proxy objektu a v cykle sa volala len web. metóda. To iste sme previedli s použitím AxWS a PxWS. Výsledky zobrazuje tabuľka tab. 1[10].

Predpokladaný benefit uvedeného riešenia spočíva v možnosti dodatočného zabezpečenia hotových web. služieb (resp. časti SOA alebo iného distribuovaného systému), ďalej možnosti zjednodušeného návrhu a implementácie web. služby (zameriavame sa len na funkcionalitu) a v poskytnutí škálovateľného bezpečnostného mechanizmu (AT nemusí byť len reťazec). Výhodou je taktiež možnosť medzičasom rozmiestňovať "skutočné"web. služby aj na iné lokácie bez toho, aby o tom ich používateľ musel vedieť a prerábať časť implemetácie na svojej klientskej strane (viď. zdroj. kód v prípadovej štúdii).

Na tomto mieste môžeme stručne spomenúť produkt *AquaLogic Service Bus*, od fy. BEA, ktorého účel a využitie je veľmi podobné ako pri našom návrhu. Informácie o ňom je možné nájsť napr. v [2, 3, 8, 11].

V nasledujúcich etapách je potrebné ešte zabezpečiť prístup ku údajovému skladu security data (príp. umožniť prístup do neho aj pre PxWS) a implementovať automatické rušenie AT používateľa po určitom časovom intervale.

	Proxy-object created first, then method called in cycles		Instance created and method called in cycle	
Classic call	Number of calls	Time [ms]	Number of calls	Time [ms]
	1	**171,875**	1	171,875
	10	**234,375**	10	234,375
	100	812,5	100	781,25
	1000	6343,75	1000	6421,875
	Note: tested on localhost without HTTPS using			
Call via AuWS and PxWS	Proxy-object created first, then method called in cycles		Instance created and method called in cycle	
	Number of calls	Time [ms]	Number of calls	Time [ms]
	1	**15,625**	1	328,125
	10	**125**	10	578,125
	100	1203,125	100	2828,125
	1000	11671,875	500	13296,875
	Note: tested on localhost without HTTPS using			

Obrázok 7. Časy odoziev pri klasickom prístupe ku web. službe s využitím AuWS a PxWS.

Referencie

1. Alonso G. and Casati F., Web Services and Service-Oriented Architectures, ICDE (archive). In proc. of the 21st International Conference on Data Engineering (ICDE'05), 00, ISBN ~ISSN:1084-4627, IEEE Computer Society, 2005, 1147
2. What is BEA AquaLogic Service Bus?, BEA AquaLogic Service Bus 2.0 Documentation, Bea Systems, `http://e-docs.bea.com/alsb/docs20`, 2007
3. Done P., Securing Services Using the AquaLogic Service Bus, tutorial. BEA Systems, 2006
4. Endrei M., Ang J., Arsanjani A., Chua S., Comte P., Krokdahl P., Luo M., and Newling T.: Patterns: Service-Oriented Architecture and Web Services, WebSphere® Software. IBM, RedBooks, 2004
5. Erl T., Service-Oriented Architecture – Concepts, Technology, and Design. Prentice Hall Professional Technical Reference, ISBN 0-13-185858-0, 1, `www.serviceoriented.ws`, 2005
6. Imamura T., Tatsubori M., Nakamura Y., and Giblin Ch., Posters: Web Services Security Configuration in a Service-Oriented Archtecture, Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, ACM Press, 2005
7. Liu R., Chen F., Yang H., Chu W.C., and Lay Y.B., Agent-Based Web Services Evolution for Pervasive Computing. In Proc. of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), IEEE, 2004, 726–731
8. Mahajan R., SOA and the Enterprise – Lessons from the City. IEEE International Conference on Web Services (ICWS'06), 2006
9. Papazoglou M.P. and Georgakopoulos D., Guest Editors: Introduction to Service-Oriented Computing. Communications of the ACM, 46, 10, ACM Press, 2003, 24–28
10. Tsai W.T., Fan Ch., Chen Y., Paul R., and Chung J.-Y., Architecture Classification for SOA- Based Applications, Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06), 2006, 295–302
11. Umapathy K. and Purao S., Designing Enterprise Solutions with Web Services and Integration Patterns, IEEE International Conference on Services Computing (SCC'06), 2006
12. World Wide Web Consortium, `http://www.w3.org/`
13. Kačmář Dalibor: Programujeme .NET aplikace, Computer Press, 2001

[10] Padoxom je, že hoci sme ku web. metóde "skutočnej" web. služby prístupovali cez PxWS a ešte predtým sme použili AuWS, časy pri prvom a desiatich volaniach kedy sa v cykle volala len web. metóda, boli ešte kratšie ako pri štandardnom prístupe.

Simulácia firmy prostredníctvom autonómnych agentov

Branislav Bošanský and Cyril Brom

Matematicko-fyzikálna fakulta, Univerzita Karlova v Prahe,
Kabinet software a výuky informatiky, Malostranské nám. 25, Praha 1
bbosansky@zoznam.sk, brom@ksvi.mff.cuni.cz

Abstrakt *Popis spoločnosti pomocou procesov je bežnou praxou, ktorá poskytuje manažérom či vedúcim pracovníkom prehľad o fungovaní firmy, pričom jej výhodou je predovšetkým jednoduchosť a zrozumiteľnosť výsledného návrhu. Na druhú stranu je však uvedená špecifikácia menej vhodná pre účely simulácie takto definovaných organizácií, kde v súčasnosti používané programy sú založené na štatistických vyhodnoteniach bez možnosti vizualizácie jej vlastného priebehu. Vhodnejšou metódou pre vytvorenie vierohodnej simulácie je preto využitie autonómnych agentov, poskytujúcich reálnejší obraz modelovanej oblasti s možnosťou vizuálnej prezentácie či interaktívnych zásahov zo strany užívateľa. Problémom tohto prístupu je však vo všeobecnosti vyššia náročnosť tvorby návrhu, vzhľadom k nutnosti špecifikovať akcie pre jednotlivých agentov. V článku sa preto venujeme spojeniu tejto oblasti agentov a ich využitia pri simulácii firmy definovanej pomocou procesov, kde pri riešení vychádzame z procesov riadených udalosťami (EPC z angl. Event-Driven Process Chains) a ich formálneho popisu EPML (EPC Markup Language), ktorý rozširujeme o ďalšie prvky nutné pre zachytenie informácií umožňujúcich „preloženie" tohto jazyka do podoby pravidiel pre reaktívnych agentov, interagujúcich v multiagentovom systéme, ktorý reprezentuje virtuálnu firmu. Článok popisuje aktuálny stav prebiehajúcej práce, kde špecifikujeme potrebné rozšírenia EPML a popisujeme ich interpretáciu a využitie vo výslednom multiagentovom systéme.*

1 Úvod

Pre študentov ekonomických a manažerských oborov by bolo vhodným obohatením výuky mať možnosť vytvorenia návrhu firmy a následne s ním interaktívne pracovať v podobe simulácie, kedy by sa im vizuálne prezentoval príslušný virtuálny svet, v rámci ktorého by virtuálni zamestnanci na základe daného návrhu plnili svoje úlohy. Pre uskutočnenie práve uvedeného, ale aj iných podobných zámerov, je tak potrebné preskúmať nielen možnosti definovania návrhu firmy, kde sa v praxi využíva najmä modelovanie pomocou procesov, ale aj možnosti jeho transformácie do podoby výslednej simulácie.

1.1 Procesné modelovanie

Procesné modelovanie (BPM z angl. *Business Process Modelling*) sa v priebehu 90-tych rokov stalo používanou technikou pre zachytenie organizácie práce v spoločnostiach, ich oddeleniach až po znázornenie jednotlivých pracovných postupov a metodík. Významnými hľadiskami, ktoré sa zaslúžili o jeho popularitu, sú:

- *prirodzenosť* - uvažovanie o práci resp. pracovných postupoch vo forme navzájom nadväzujúcich procesov je prirodzené (pracovník A pracuje na úlohe U1, keď skončí, pracovník B bude s výsledkom úlohy U1 pracovať na úlohe U2)
- *jednoduchosť* - zachytenie pracovného postupu do formy procesov nie je príliš zložité[1]
- *prehľadnosť a zrozumiteľnosť* - na základe procesných špecifikácií je možné veľmi rýchlo získať prehľad a zorientovať sa vo fungovaní firmy či týmu ľudí

Existuje niekoľko jazykov, prostredníctvom ktorých môžeme procesy firmy zachytiť. Všetky majú istú formu grafickej reprezentácie a ich návrh prebieha väčšinou opäť grafickou cestou. Za najvýznamnejšie spomeňme najmä jazyky UML, ktorý okrem procesného návrhu poskytuje oveľa širšie možnosti využívané najmä v rámci softwareového inžinierstva, BPEL používaný v oblasti popisu práce webových služieb či EPC [1] určený na procesnú špecifikáciu firmy, zachytenie hierarchie procesov a organizačnej štruktúry. Dôležitosť procesnej špecifikácie je zdôraznená aj jej nevyhnutnosťou pre splnenie medzinárodných noriem, napríklad noriem kvality za účelom získania certifikátov ISO 9001 [2].

1.2 Simulácia firmy a využitie agentov

Špecifikácia firmy pomocou procesov prináša výhody vo forme pochopenia organizácie práce a dáva priestor na jej zlepšenie, no môže taktiež poslúžiť ako základ pre ďalšie využite. Medzi také môžeme napríklad zaradiť systémy pre kontrolu plnenia procesov, systémy pre podporu v rozhodovaní pre manažérov a najmä, už spomínané, možnosti simulácie takto navrhnutých postupov, ktoré, ako sme už na začiatku predznamenali, sú pre nás kľúčovou oblasťou, pričom jej využitie nie je obmedzené len na výuku študentov. Možnosť vyskúšať si niekoľko rôznych variant riešení zadanej

[1] myslené relatívne v porovnaní s inými formami popisu, ktoré budú spomenuté ďalej

úlohy a vidieť tak ich dopad na modelovú situáciu, je využiteľné i pre reálne spoločnosti, no je však vždy nutné dbať na správnu interpretáciu výsledkov simulácie a na vloženie korektných parametrov či zvolení vhodnej úrovne jej detailnosti. Súčasné možnosti simulácií firiem vieme rozdeliť na dva hlavné prúdy:

1. pravdepodobnostné vyhodnotenie jednotlivých variant pracovných postupov a z nich vyplývajúce štatistiky hodnotiace napr. odhadovaný zisk
2. simulácie využívajúce agentový prístup, kde je firma reprezentovaná multiagentovým systémom

Reprezentantom prvého smeru je napríklad nástroj ARIS Toolset [3] popisujúci procesy prostredníctvom jazyka EPC. Po doplnení údajov o dĺžke trvania jednotlivých činností a priradení pracovníkov (pochopiteľne, jeho možnosti sú ďaleko väčšie) vyhodnotí prechody daných procesov pričom ako výsledok poskytne údaje o odhadovaných ziskoch, trvaní jednotlivých procesov a podobne. Bohužiaľ, takáto simulácia má základné nedostatky práve v podobe čisto pravdepodobnostného vyhodnotenia, pomocou ktorého nie je napríklad možné riešiť pracovné činnosti súvisiace s komunikáciou medzi jednotlivými aktérmi [4] či iné komplexnejšie problémy a často, ako napríklad v uvedenom prípade programu ARIS, chýba vizuálne znázornenie samotného priebehu simulácie. Pochopiteľne, je mož-né rozšíriť túto formu o niektoré používané simulačné techniky,[2] avšak neodstránime tak všetky spomenuté nedostatky.

Smer, naznačený ako druhý, je zaujímavý najmä z pohľadu distribuovanej umelej inteligencie. Ako uvádzajú autori v [5], procesy v reálnych spoločnostiach je ťažké aproximovať jednoduchou funkciou, na základe ktorej prebehne výpočet spomenutý v predchádzajúcom odstavci. Je to najmä z dôvodu ich nepredikovateľnosti, kooperácie ľudí v rámci tímu či naopak distribúcie práce a využití zdrojov. Tým, že firmu reprezentujeme ako multiagentový systém a jednotlivé procesy ako problémové úlohy pre agentov predstavujúcich jednotlivých pracovníkov, môžeme využiť ich pozitívne vlastnosti, ako reaktívnosť, proaktívnosť či schopnosť sociálneho správania. Príkladom využitia agentov pri implementácii business procesov je, už odkazovaná práca [5] modelujúca spoločnosť British Telecom. Ako je vidieť aj na príkladoch uvedených v odkazovanom článku, v prípade popisu firmy, respektíve aj všeobecne multiagentového systému, autori špecifikujú akcie jednotlivo pre každého agenta, pričom túto metódu môžeme označiť za prirodzenú (vzhľadom k tomu, že definujeme chovanie autonómnych entít) a je využitá aj napríklad pri agentových simulačných nástrojoch SeSAm [6] či Brahms [7].

[2] napríklad diskrétnu simuláciu

Ak sa skúsime zamyslieť nad možnou aplikáciou tejto metódy (napríklad v podobe *if-then* pravidiel) na popis reálnych spoločností zistíme, že formalizácia komplexných firiem vyžaduje vyššie znalosti (hlavne z oblasti informatiky a matematiky), nie je dostatočne prehľadná a ťažšie sa v nej orientuje prípadne hľadajú nedostatky. Práve rozdielnosť špecifikácie práce agentov oproti zaužívanému procesnému návrhu preto identifikujeme ako hlavnú prekážku ich väčšieho využitia pre možnosti simulácie firiem. Cieľom výskumu je preto:

1. návrh takého formálneho popisu, ktorý by kombinoval výhody procesnej špecifikácie a zároveň by bol schopný niesť informácie potrebné pre definovanie pravidiel pre agentov
2. vytvorenie a implementácia modelu virtuálnej firmy navrhnutej procesným modelovaním.

V tomto texte sa zameriavame prvú časť tohoto cieľa, pričom predstavujeme základné myšlienky prezentujúce aktuálny stav pokračujúceho výskumu[3]. Zvyšok článku je organizovaný nasledujúcim spôsobom: v druhej časti definujeme problém na nižšej úrovni, naznačíme možné spôsoby riešenia a zadefinujeme EPC. V tretej časti sa budeme ďalej venovať jazyku EPML a jeho rozšíreniam, ktoré sú nutné pre umožnenie prekladu do výsledného multiagentového systému. V posledných sekciách popíšeme spôsoby implementácie a taktiež ďalšiu prácu, a to ako myšlienky riešenia druhej časti výskumu, tak aj možnosti využitia v iných oblastiach.

2 Analýza problému

Ako sme uviedli v predchádzajúcej sekcii, cieľom práce je definovať jazyk, založený na procesnom princípe, ktorý by sme vedeli transformovať do podoby používanej na popis správania sa agentov v multiagentovom systéme, čím dokážeme zlúčiť už spomínané výhody procesného návrhu (jednoduchosť, prehľadnosť) s výhodami a silou agentov v oblasti simulácií. Pri definovaní takéhoto jazyka máme v zásade dve možnosti: buď vytvoriť nový jazyk, ktorý bude spĺňovať očakávané predpoklady, alebo za týmto účelom upraviť niektorý z existujúcich jazykov. Na začiatku práce sme po úvahe prvú možnosť zamietli z týchto príčin:

kompatibilita - bolo by výrazne náročnejšie dosiahnuť kompatibilitu s existujúcimi programami pre tvorbu procesných návrhov firiem

jednoduchosť používania - užívatelia by boli nútení pracovať s neznámym formalizmom, ktorý by museli naštudovať a osvojiť si ho

[3] momentálne prebiehajúceho vo forme diplomovej práce na MFF UK

využiteľnosť už uložených dát - vzhľadom na možné problémy s kompatibilitou s existujúcimi jazykmi, bolo by oveľa ťažšie využiť množstvo už uložených znalostí

Z uvedených dôvodov sme sa primárne venovali variante rozšírenia niektorého zo v súčasnosti používaných jazykov (UML, BPEL, EPC), spomínaných už v úvode, pričom ako základ bol nakoniec zvolený EPC. Dôvodom pre výber tohoto jazyka bola najmä jeho rozšírenosť a popularita (vychádza z neho ARIS, ale používa ho aj Microsoft Visio), zameranie práve na popis procesov firiem (aj v spojení s organizačnou hierarchiou) a vyššia výpovedná hodnota oproti napr. jazyku UML. Na druhú stranu, možnou komplikáciou vyplývajúcou z tejto voľby je nejednoznačnosť v definíciách a formalizovaných podobách EPC.

2.1 Definícia EPC

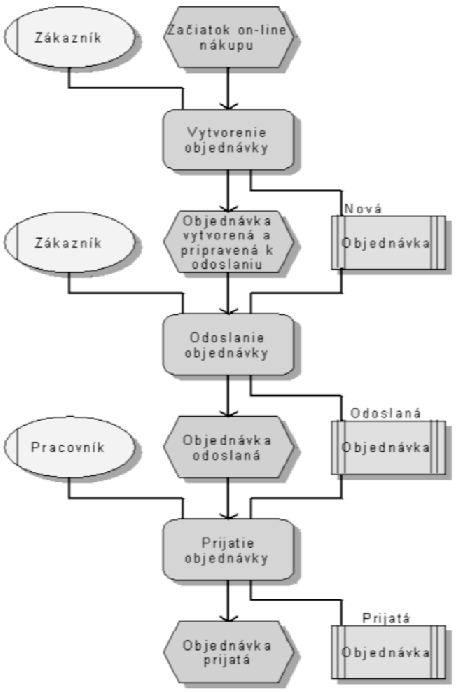
Neformálne[4] môžeme EPC charakterizovať ako postupnosť procesov (v rámci EPC sú nazývané *funkcie*) doplnených o udalosti, prostredníctvom ktorých dochádza buď k aktivácii bezprostredne nasledujúceho procesu, resp. dôjde k výskytu danej udalosti po úspešnom ukončení procesu, ktorý jej predchádzal. Príklad jednoduchého EPC diagramu znázorňuje obrázok 1, kde vidíme udalosti znázornené ako zaoblené obdĺžniky a funkcie ako šesťuholníky. Pre rozdelenie kontrolného toku (z angl. *control flow*) využíva EPC tri konektory (*and, or* a *xor*), pomocou ktorých je možné vytvoriť niekoľko nezávislých tokov či ich opäť spolu synchronizovať.

Pod označením EPC však rozumieme aj ďalšie varianty a rozšírenia uvedenej základnej definície. Bežne využívané (napr. v programe ARIS, prípadne popísané v [9]) je doplnenie o nasledujúce elementy:

- *dátové pole* (*data field*) reprezentujúce dátové elementy (zdroje potrebné pre vykonanie procesu)
- *aktér* (*participant*), prostredníctvom ktorého môžeme zachytiť organizačnú štruktúru spoločnosti resp. pri spojení s procesom tak určujeme jeho vykonávateľa

Pochopiteľne existuje ešte niekoľko ďalších možných rozšírení (napr. v ARIS-e sú to špeciálne typy dátových polí ako *počítačový systém, telefón, fax* apod.), avšak pre náš zámer nie sú tieto potrebné. V rámci simulácie firmy potrebujeme totiž reprezentovať všetky entity, ktoré sa v rámci procesov využívajú a nie je potrebná žiadna špecializácia pre vybrané typy.

[4] formálnu definíciu a úplný popis jazyka je možné nájsť v [1], prípadne [8]



Obrázok 1. Príklad procesnej špecifikácie nákupu v on-line obchode znázornená pomocou EPC.

2.2 Formálny zápis EPC

Nejednoznačnosť v definícii ovplyvnila aj nejednoznačnosť vo formálnych zápisoch EPC, kde každý program využíva svoj vlastný formát [9]. Keďže, ako sme naznačili v predchádzajúcich častiach, budeme rozširovať možnosti jazyka EPC o uloženie informácií pre vytvorenie agentov, rozumnou požiadavkou pre formalizmus, z ktorého budeme vychádzať, je jednoduchosť, prehľadnosť a v ideálnom prípade dostatočná abstrakcia, ktorá by zahrňovala všetky ostatné formalizmy. Týmto požiadavkám takmer presne odpovedá jazyk EPML (z angl. *EPC Markup Language*) [9], ktorý vznikol práve za účelom premostenia jednotlivých formátov EPC.

3 Jazyk EPML a jeho rozšírenie

Jazyk EPML je založený na báze XML a zjednodušene teraz popíšeme jeho najdôležitejšie elementy (pre úplnú špecifikáciu viď [9] prípadne [10], príklad jazyka EPML, už doplneného o rozširujúce prvky, je zobrazený na obrázku 2). Základným prvkom EPML je stromová štruktúra jednotlivých EPC diagramov, kde jej listom (teda samotným diagramom) je element ⟨*epc*⟩

a vnútorné uzly sú adresáre ⟨*directory*⟩. V EPC diagrame je reťazec procesov uložený pomocou tzv. *EPC elementov* reprezentujúcich udalosti ⟨*event*⟩, procesy ⟨*function*⟩, aktérov ⟨*participant*⟩, dátové položky ⟨*dataField*⟩ a konektory (⟨*and*⟩, ⟨*or*⟩ a ⟨*xor*⟩). Tie sú navzájom prepojené dvoma typmi orientovaných hrán – jedná sa o hrany kontrolného toku ⟨*arc*⟩ (spájajúce udalosti, procesy a konektory) a hrany relácií ⟨*relation*⟩ (spájajúce aktérov a dátové položky s procesmi).

Okrem vlastných EPC diagramov obsahuje štruktúra súboru EPML taktiež definície použitých atribútov (element ⟨*attributeTypes*⟩) a pomocou elementu ⟨*definitions*⟩ umožňuje vyčlenenie opakujúcich sa údajov jednotlivých objektov s cieľom sprehľadniť výslednú štruktúru. Príklad vidíme práve na obrázku 2 na objekte objednávky (element ⟨*dataField*⟩), ktorý sa pomocou atribútu *defRef* odkazuje na definíciu uvedenú na začiatku súboru.

3.1 Procesy z hľadiska simulácie

V nasledujúcej sekcii budeme hľadať informácie, ktoré musíme do klasického EPC doplniť, aby sme vedeli vytvoriť pravidlá pre jednotlivých agentov. Uvážme najprv základné požiadavky na proces a následne aj rozšírenia potrebné pre výslednú simuláciu:

vstupy - vstupné objekty na základe používanej varianty EPC môžu byť dvoch typov, a to buď informácia o aktivácií procesu v danom reťazci procesov (tj. prítomnosť kontrolného toku prostredníctvom predchádzajúcich elementov (funkcií, udalostí či konektorov)), alebo dátové objekty potrebné pre vykonanie daného procesu. Pre vstupné procesy potrebujeme navyše poznať dve kľúčové hodnoty atribútov - ich povinnosť (či sú pre vykonávanie tohoto procesu povinné) a ich využívanie počas trvania procesu. Úroveň využívania objektu je nutná pre možnosť simulovania procesov na hrubšej úrovni detailov, kedy v rámci priebehu dlhšieho procesu nie je tento objekt plne využívaný.

výstupy - medzi výstupné objekty zaraďujeme opäť predanie informácie o kontrolnom toku do nasledujúcich elementov a nastavenie výstupných hodnôt príslušných atribútov výstupných dátových objektov. Ďalej však ešte potrebujeme vedieť či má vzniknúť nový výstupný objekt, alebo sa má modifikovať hodnota objektu, ktorý bol pri priebehu procesu použitý a teda je nutné, aby v tomto druhom prípade niesol výstupný objekt odkaz na príslušný vstup

aktéri - aktéri sú prepojený s daným procesom a určujú, ktorí agenti sa majú na vykonávaní tohoto procesu podieľať. Podobne ako pre vstupné objekty u nich potrebujeme poznať ich povinnosť a úroveň využitia.

miesto - keďže je cieľovou platformou určitý virtuálny svet, agenti budú stelesnený a budú sa v ňom pohybovať, môžeme mať určené miesto, kde sa má tento proces vykonávať. Jeho určenie bude relatívne vzhľadom na určitý objekt.

priebeh procesu - ak chceme simulovať proces, potrebujeme pre jeho úspešné dokončenie poznať dĺžku jeho trvania. Tá by štandardne mala byť zadaná ako parameter pre proces, pričom v samotnom priebehu simulácie by jej skutočná hodnota bola modifikovaná v závislosti na vhodnej pravdepodobnostnej distribúcii. Ak však zoberieme do úvahy aj možnosť prerušenia vykonávania takéhoto procesu (čo je vzhľadom na zamýšľané použitie rozumná požiadavka), potrebujeme ďalej poznať jeho prioritu v porovnaní s ostatnými procesmi, ale hlavne celkovú priebehovú funkciu procesu, aby bolo možné u atribútov výstupných objektov nastaviť hodnoty odpovedajúce danému časovému okamžiku. Pre úplnosť je nutné poznamenať, že priebehová funkcia istých procesov sa vo všeobecnosti môže líšiť v závislosti od využitia voliteľných vstupných objektov, kooperácie viacerých agentov či miesta vykonávania procesu.

Uvedené rozšírenia nám poskytnú dostatok informácií pre simuláciu procesov, pričom vďaka určitej forme abstrakcie (hlavne v súvislosti priebehovou funkciou) sme schopný simulovať aj netriviálne procesy a nie je nutné vytvárať procesný návrh až do úplných detailov. Na druhú stranu rozšírení, ktoré sme momentálne do návrhu nezahrnuli, by bolo možné nájsť ešte niekoľko (napríklad definovanie úrovne vyťaženia vstupného objektu tiež pomocou funkcie alebo definovanie zložitejších logických podmienok nad vstupmi procesu) a určujú smer, ktorým by bolo možné ďalej prácu rozvíjať.

3.2 Reprezentácia rozšírených procesov

Ako už bolo naznačené, väčšina rozšírení je zastúpená doplnením jednoduchých atribútov (tj. reálne číslo, logická hodnota a pod.), čo je vďaka pravidlám EPML umožňujúcim do EPC elementov (ako *function*, *event*, atď.) vložiť element *attribute*, ľahko realizovateľná úloha. Na obrázku 2 vidíme príklad rozšíreného jazyka EPML, kde je zobrazená časť súboru popisujúceho proces z obrázku 1, konkrétne je vybraná funkcia "Odoslanie objednávky" spolu so súvisiacimi udalosťami, dátami a aktérom. Príklad ukazuje jednak základné elementy jazyka EPML ako sme ich v krátkosti špecifikovali na začiatku sekcie 3, ale taktiež vidíme doplnené atribúty pri jednotlivých vstupných objektoch (ako sú objednávka – *dataField* a zákazník – *par-*

```
<epml>
...
<attributeTypes>
   <attributeType typeId="orderState" />
   <attributeType typeId="zakaznikID" />
   <attributeType typeId="utilization" />
   <attributeType typeId="necessity" />
   <attributeType typeId="existingRef" />
   <attributeType typeId="priority" />
   <attributeType typeId="location" />
   ...
</attributeTypes>

<definitions>
   <definition defID="0003">
      <name>Objednavka</name>
      <attribute typeRef="zakaznikID" value="Zakaznik"/>
   </definition>
   ...
</definitions>

<directory name="Root">
<epc epcId="1" name="Online Shopping">
 ...
<event id ="3">
   <name>
     Objednavka vytvorena a pripravena k odoslaniu
   </name>
</event>

<arc id ="102">
   <flow source ="3" target ="4"/>
</arc>

<dataField id="32" defRef="0003" >
   <attribute typeRef="orderState" value="Nova" />
   <attribute typeRef="utilization" value="1.0" />
   <attribute typeRef="necessity" value="true" />
</dataField>
<relation id="73" from="32" to="4" />

<participant id="21">
   <name>Zakaznik</name>
   <attribute typeRef="utilization" value="1.0" />
   <attribute typeRef="necessity" value="true" />
</participant>
<relation id="74" from="21" to="4" />

<function id ="4">
   <name>Odoslanie objednavky</name>
   <attributeType typeRef="priority" value="10" />
   <transitionFunction className="cz.simphi.functions.
    Heaviside">
      <defaultParameters>
         <parameter type="divFunction"
            value="cz.simphi.distributions.GaussDistribution"/>
      </defaultParameters>
      <outputObjects>
         <expectedFinishTime value="5">
            <parameter type="diversity" value="1" />
         </expectedFinishTime>
         <outputDataObject idRef="31">
            <attribute typeRef="orderState" value="Odoslana" >
               <parameter type="diversity" value="0" />
            </attribute>
         </outputDataObject>
      </outputObjects>
   </transitionFunction>
</function >

<dataField id="33" defRef="0003">
   <attribute typeRef="existingRef" value="32" />
</dataField>
<relation id="75" from="4" to="33" />

...
</epc>
</directory>
</epml>
```

Obrázok 2. Príklad procesu v rozšírenej EPML notácii

ticipant) a taktiež pri výstupnom objekte (opäť objednávka reprezentovaná elementom *dataField* doplnená o odkaz na existujúci vstupný objekt).

Problematickou časťou rozšírenia je tak priebehová funkcia, ktorej uloženie v podobe XML je vzhľadom k možnému vysokému počtu dimenzií a parametrov (závisí na všetkých ostatných vlastnostiach procesu) v praxi nepoužiteľné. Zavádzame preto do EPML nový element $\langle transitionFunction \rangle$, ktorého predkom je *function* a ktorý odkazuje na triedu vyššieho jazyka[5] implementujúcu priebeh daného procesu, pričom v jazyku XML ponechávame jej nastavenia. V príklade na obrázku 2 tak vidíme použitie jednoduchej prechodovej funkcie, ktorá reprezentuje Heavsidovu skokovú funkciu. Návrh rozšírenia pre prechodové funkcie bol do prekladača jazyka EPML implementovaný tak, že spracovanie XML elementov vnorených v *transitionFunction* zabezpečuje samotná trieda reprezentujúca túto funkciu. Dôvodom pre toto rozhodnutie je vysoká variabilita výstupných dimenzií rôznych funkcií, ich parametrov a vplyvov týchto parametrov na priebeh funkcií jednotlivých hodnôt výstupných objektov. Vďaka tomuto oddeleniu tak môžeme pre popis priebehu procesu použiť ľubovoľnú funkciu, keďže nebudeme obmedzený možnosťami jej nastavenia.

Popíšme teraz elementy určujúce vlastnosti funkcie odpovedajúcej príkladu na obrázku 2, pričom podobné prvky sa nachádzajú aj v iných funkciách, ktoré budú v druhej fáze výskumu implementované. Kľúčovými prvkami pre proces sú nastavenie dĺžky jeho trvania, určenie konečných hodnôt (a ich možnej variability) u atribútov výstupných objektov a taktiež nastavenie parametrov funkcie, reprezentujúcej priebeh procesu. Väčšina z uvedených vlastností je v našom príklade zachytená elementom $\langle outputObjects \rangle$, kde v elemente $\langle expectedFinishTime \rangle$ zaznamenávame očakávanú dĺžku procesu a pomocou elementov $\langle outputDataObject \rangle$ nastavujeme hodnoty atribútov výstupných objektov. Oba druhy nastavení je možné doplniť ďalšími parametrami (pomocou elementu $\langle parameters \rangle$). Tie môžu jednak určovať mieru variability výstupnej hodnoty (parameter *divFunction* ukazujúci na pravdepodobnostnú distribúciu a jej parameter *diversity* reprezentujúci rozptyl), ale v prípade komplikovanejších funkcií pomocou nich taktiež určujeme vlastný priebeh výstupných hodnôt (napríklad v prípade sigmoidy jej strmosť). Opakujúce sa parametre je možné z dôvodu zjednodušenia uviesť na začiatku popisu prechodovej funkcie v elemente $\langle defaultParameters \rangle$.

Vzhľadom k tomu, že nutnosť programovania priebehových funkcií procesov pri ich návrhu nie je žiadúca, budú implementované triedy popisujúce jednoduché funkcie (ako napríklad lineárnu, skokovú, sigmoidu a pod.) a taktiež triedy reprezentujúce zložitejšie funkcie, kde príkladom môže byť zloženie výsled-

[5] pre implementáciu je využívaná jazyk Java

nej funkcie z funkcií jednoduchých (t.j. funkcie sa budú líšiť pre jednotlivé výstupné objekty a ich atribúty).

Dôležitým predpokladom pre spracovanie takto rozšírených procesov je nutná ich plná inštanciovanosť (tj. musia byť zadané všetky vymenované atribúty). Tento fakt je možné po teoretickej stránke v rámci EPML vyriešiť najmä za využitia elementu *definitions*, praktická implementácia však bude riešená až pri tvorbe samotného nástroja, ktorý však spadá mimo rozsah nášho výskumu.

3.3 Prevod rozšírených procesov na pravidlá

Uveďme teraz interpretáciu jednoduchého príkladu z obrázku 2 v podobe pravidiel a priebehu jeho simulácie. V popise správania sa agenta reprezentujúceho zákazníka *Zákazník*, by sa tak v predpokladoch pravidla nachádzala podmienka na výskyt udalosti *Objednávka vytvorená a pripravená k odoslaniu*, dostupnosť agenta (teda agent nevykonáva inú činnosť s vyššou prioritou) a na existenciu objednávky tohoto zákazníka v stave *Nová*. Po splnení podmienok by bolo prostredníctvom priebehovej funkcie spustené vykonávanie procesu týmto agentom a po jeho skončení (vzhľadom k použitej skokovej funkcii) by došlo k nastaveniu stavu použitej objednávky na *Odoslaná* a k vyvolaniu udalosti *Objednávka odoslaná*.

Pochopiteľne, uvedený príklad bol ukážkou jednoduchej transformácie a pri implementácii končeného virtuálneho sveta bude realizovaný preklad komplexnejších procesov (napr. procesov s rozčlenenými tokmi, variabilnejšími vstupnými a výstupnými objektmi a takisto hierarchických procesov).

4 Implementácia a nadväzujúca práca

Implementácia, rovnako ako aj ciele samotnej práce, je rozdelená na dve časti. Prvá odpovedá vytvoreniu prekladača rozšíreného jazyka EPML do podoby všeobecných pravidiel, kde pre ich validáciu využívame systém JBoss Rules[11] a v ďalšom priebehu bude nasledovať vytvorenie finálnej simulácie, ktorá bude odpovedať časti reálnej firmy. S tým súvisí namodelovanie odpovedajúceho virtuálneho sveta, vo vhodnom simulačnom multiagentovom systéme, pričom pravdepodobne bude využitý nástroj IVE [12].

5 Záver

V článku bolo predstavené zadanie a postup riešenia práce, ktorá si za cieľ kladie vytvoriť simuláciu firmy s využitím autonómnych agentov. V doterajšom priebehu riešenia, ktorý je pokrytý týmto textom, bol kladený dôraz na špecifikovanie vhodného jazyka pomocou ktorého by bolo možné tento cieľ naplniť.

Rozsahom tejto práce však naznačená problematika rozhodne nie je vyčerpaná. Drobné možnosti rozšírenia (za účelom zreálnenia simulácie) boli uvedené priamo v texte, no možnosti sú oveľa širšie. Je to z dôvodu, že procesný spôsob návrhu práce a fungovania multiagentového systému nie je používaný, no jeho uplatnenie by mohlo byť okrem simulácie firiem použiteľné aj napr. v oblasti formalizovaných lekárskych doporučení a preto bude výskum v tejto oblasti pokračovať.

Práca na tomto texte bola čiastočne podporená grantovým projektom GA UK ČR 351/2006/A-INF/MFF, grantom "Information Society" pod číslom projektu 1ET100300517 a grantom Ministerstva Školstva ČR MSM0021620838. Poďakovanie patrí taktiež Petrovi Kocábovi zo spoločnosti SoftDeC s r.o.[6].

Referencie

1. Keller G., Nüttgens M., and Scheer A.-W., Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Technical Report Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes, January 1992
2. Int. Organisation for Standardisation: ISO 9001–4, Quality Systems. `http://www.iso.org`
3. IDS Scheer AG: ARIS Design Platform, ARIS Simulation. `http://www.ids-scheer.com`
4. Sierhuis M., Modeling and Simulating Work Practice, Universiteit van Amsterdam, 2001
5. Jennings N.R., Faratin P., Norman T.J., O'Brien P. and Odgers B., Autonomous Agents for Business Process Management. Int. Journal of Applied Artificial Intelligence, 14, 2000, 145–189
6. Klügl F., Herrler R., and Fehle M., SeSAm: Implementation of Agent-Based Simulation Using Visual Programming. In Proceedings of the AAMAS 2006, 2006
7. Clancey W.J., Sachs P., Sierhuis M., and van Hoof. Brahms R., Simulating practice for work systems design. International Journal of Human-Computer Studies, 49, 1998, 831—865
8. Rosemann M. and van der Aalst W., A Configurable Reference Modelling Language. Information Systems, 32, 2007, 1–23
9. Mendling J. and Nüttgens M., EPC Markup Language, Technical Report, Vienna University of Economics and Business Administration, 2005
10. Mendling, J. EPML XML Schema, `http://wi.wu-wien.ac.at/home/mendling/EPML/EPML_12.xsd`, Version 1.2
11. JBoss Rules, Version 3.0.6, `http://www.drools.org`
12. IVE, stránky projektu, `http://mff.modry.cz/ive/`

[6] http://www.softdec.cz

Jak rychle prototypovat chování umělých bytostí: Pogamut 2*

Ondřej Burkert, Rudolf Kadlec, Jakub Gemrot, Michal Bída, Jan Havlíček, and Cyril Brom

Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra software a výuky informatiky
Praha, Česká republika
ondra@atrey.karlin.mff.cuni.cz, brom@ksvi.mff.cuni.cz
URL: http://artemis.ms.mff.cuni.cz

Abstrakt *Zájem o nasazování umělých bytostí v různých aplikacích od her přes výukové a terapeutické nástroje po film neustále roste. V souvislosti s tím se objevují nové výzkumné problémy (např. problematika umělých emocí či sociálních interakcí). Výzkumníci však často tráví neúnosně mnoho času vytvářením vlastního vývojového prostředí pro bytosti a integrací bytostí do virtuálního světa. Tento článek představuje platformu pro rychlý vývoj chování umělých bytostí vtělených do trojrozměrného světa počítačové hry. Platforma nabízí sadu knihoven pro snadnou definici chování umělé bytosti a integrované vývojové prostředí, které umožňuje rychlé ladění kódu, parametrizaci a vytváření experimentů. Platforma je především určena pro výzku. Díky souboru tutoriálů je ovšem přístupná i širší odborné veřejnosti a lze ji využít také jako výukový nástroj.*

1 Úvod

V dnešní době stále stoupá zájem o umělé bytosti. Umělými bytosmi rozumíme speciální softwarové agenty dle Wooldridge [1] vtělené ve virtuálním světě. Vývoj chování umělých bytostí však není snadný. Důvody jsou zřejmé: agenti jednají v dynamickém, nepředvídatelném, interaktivním světě, jejich úkoly zahrnují, mimo pohybu po světě, různorodé netriviální úkoly, agenti spolu mohou také sociálně interagovat, projevovat emoce apod.

Příklady takových umělých bytostí můžeme nalézt například v moderních počítačových hrách [2], výukových hrách [3], terapeutických nástrojích [4] a virtual storytellingu [5]. Mnoho odlišných aplikací s sebou bohužel nese mnoho různých vývojových prostředí. Kvalitní vývojové prostředí je nezbytným předpokladem pro rychlý vývoj agentů. Studenti a výzkumnící, kteří chtějí vyvíjet vlastní agenty, však nemají k dispozici kompletní stáhnutelnou aplikaci. Vývoji samotného agenta tak musí předcházet náročný vývoj vlastního vývojového prostředí i virtuálního světa, či připojení vývojového prostředí ke světu počítačové hry (např. Unreal Tournament, Quake). Dostupná vývojová prostředí jsou (a) komerční (např. AI-Implant [7], Xaitment [9]), (b) proprietární řešení jednotlivých výzkumných skupin (např. [6]), (c) volně šiřitelné aplikace pro usnadnění vývoje umělých bytostí.

Komerční nástroje jsou pro nováčky v oboru nevhodné ze tří důvodů. Zaprvé předpokládají znalost mnoha pokročilých algoritmů umělé inteligence, které implementují. Zadruhé vyžadují integraci do konkrétního virtuálního světa. Připojení do něj však obecně není triviální záležitost. Zatřetí jsou poměrně drahé.

Proprietární řešení jsou obecně nepoužitelná, neboť jsou určena pro konkrétní virtuální svět a zkoumaný problém. Navíc bývají často nedostatečně dokumentovaná a podporovaná.

Mezi volně šiřitelné nástroje určené pro tvorbu vlastních agentů patří například projekt Gamebots [10], který představuje rozhraní pro připojení agentů do hry Unreal Tournament 1999 [15], nebo F.E.A.R. [11], který k integraci agentů do hry Quake přidává framework pro návrh a vývoj agentů. Bohužel tyto aplikace neobsahují moduly, jako je integrované vývojové prostředí, podpora experimentů, knihovna modulů pro agentovy základní funkce, manažer žurnálů apod. Tyto moduly jsou nezbytné pro rychlý vývoj chování agentů. Naším cílem je tuto mezeru zaplnit a nabídnout nekomerční, stáhnutelnou platformu pro rychlý vývoj umělých bytostí.

Představovaná platforma se jmenuje Pogamut 2 a navazuje na naši předchozí práci (Pogamut 1 [12]). Nabízí navíc následující sadu nástrojů: (1) komunikaci agenta s virtuálním světem, (2) pomocné knihovny – senzory, správu paměti agenta, inventář, reprezentaci světa, A* algoritmus (algoritmus pro hledání cest [13]) atd., (3) integrované vývojové prostředí (IDE) s širokou podporou vývoje, ladění a experimentů, (4) POSH [14] – rozhodovací systém (action selection mechanism - ASM).

Pogamut 2 je určen: a) pro výzkumné projekty zaměřené na chování umělých bytostí (mezi jinými výzkum spolupráce agentů a výzkum emocí), b) pro výuku problematiky umělých bytostí na akademické půdě. Jako virtuální 3D svět byl použit svět hry Unreal Tournament 2004 [15] (UT04). UT04 nabízí mimo editoru lokací mnoho předdefinovaných předmětů, lokací a typů her.

* Tato práce byla podpořena granty GA UK 1053/2007/A-INF/MFF, GA UK 351/2006/A-INF/MFF a programem "Information Society" pod projektem 1ET100300517.

Obrázek 1. Záběr z hry Unreal Tournament 2004^{TM}.

Beta verzi Pogamutu 2 lze nalézt na našich stránkách[1]. Obsahuje veškerou funkcionalitu prezentovanou v tomto článku. Momentálně pracujeme na dokončení pokročilých nástrojů, přesněji na video tutoriálech a zpětným přehráváním zaznamenaných simulací a experimentů včetně inspekce zaznamenaných dat (např. stav agenta, stav ASM).

Článek má následující strukturu. V druhé kapitole probereme cíle projektu. Třetí kapitola popisuje architekturu Pogamutu 2. Čtvrtá kapitola nastiňuje styl práce s aplikací. Článek uzavírá diskuze řešení, popis probíhajících prací a shrnutí.

2 Cíle projektu

Pogamut 2 si klade následující cíle:

1. Rozšiřitelnost a modularita – licence LGPL[2] umožňuje zásahy do kódu, architektura počítá s připojením libovolných ASM (např. SOAR[16]) a dalším rozšiřování IDE a knihoven.
2. Uživatelsky přívětivé prostředí – IDE podporující intuitivní vývoj a ladění.
3. Paralelizace – díky architektuře klient/server jsou odděleny UT04 a ASM, lze tedy dělit zátěž na více strojů při zapojení mnoha agentů najednou.
4. Rychlý začátek práce – dokumentace, příklady a tutoriály snižují startovní čas
5. Snadná validace modelu – podpora experimentů usnadňuje validaci

Komerční nástroje [7], [8], [9] splňují všechny požadavky mimo experimentů. Dostupným volně šiřitelným nástrojům (F.E.A.R., Gamebots, JavaBots [17], Tielt [18], Pogamut 1 apod.) nejčastěji chybí komplexní IDE a podpora experimentů.

[1] `http://artemis.ms.mff.cuni.cz`
[2] Lesser General Public License

3 Architektura Pogamutu 2

Pogamut 2 se skládá z šesti základních modulů: (1) UT04, (2) Gamebots2004 (GB04), (3) Parseru, (4) Klienta, (5) IDE a (6) ASM. Většinu modulů jsme naprogramovali v jazyce Java. Výjimku tvoří UT04, což je komerční software, a GB04, který vznikl úpravou a rozšířením starší verze (GB [10]). Nyní představíme podrobněji jednotlivé moduly.

Unreal Tournament 2004 je počítačová hra, kterou používáme jako virtuální svět pro agenty. Umožňuje připojení až 30 agentů do jedné lokace, což je tedy horní hranice i pro Pogamut 2. UT04 lze modifikovat dvěma způsoby. Zaprvé pomocí editoru, který umožňuje například vytváření nových lokací a úpravu stávajících. Zadruhé lze upravovat přímo skripty definující chování hry, předmětů a postav ve hře. Skripty jsou zapisovány v interním skriptovacím jazyku UnrealScript. Kombinací těchto způsobů lze vytvořit v podstatě libovolný virtuální svět.

Gamebots 2004 je server zabudovaný do UT04 starající se o komunikaci mezi avatarem ve hře a jejím externím ASM. GB04 je adaptací původních GB určených pro UT99 [15]. Mezi rozšíření oproti GB patří posílání všech předmětů a navigačních bodů v lokaci před zahájením simulace, automatický raytraycing, příkazy pro ovládání serveru a nahrávání záznamů simulace.

Parser převádí řetězce z GB04 na Java objekty, které posílá dál ke Klientovi. Hlavním účelem tohoto modulu je optimalizovat komunikaci s Klientem a umožnit tak připojení většího počtu agentů.

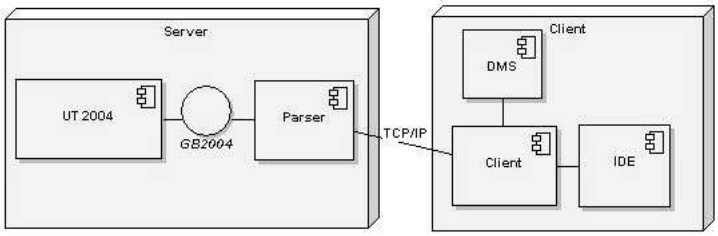
Klient je balík Java tříd. Nabízí: (a) krátkodobou paměť na externí vjemy, (b) příkazy - primitiva pro kontrolu těla agenta, (c) inventář předmětů, které agent posbíral, (d) metody pro pohyb a navigaci v lokaci (zahrnují i interní A*).

IDE představuje plug-in pro NetBeans^{TM} [19]. IDE pomáhá ve všech podstatných částech vývojového cyklu: implementaci, ladění a experimentech. Obsahuje:

- podporu skriptování chování agentů (Java, Python, reaktivní plánovač POSH [14]).
- správu knihovny dostupných agentů.
- ladící nástroje – inspektor interních proměnných agenta, pohledy na agentovu paměť, přehled žurnálů z ASM a komunikace, atd.
- sadu metod pro definici experimentů – uživatel může definovat počáteční konfiguraci, počet opakování, ukončovací podmínku apod.

4 Práce s Pogamutem 2

Obecně lze proces vytváření umělé bytosti rozdělit do následujících kroků: (1) definice modelu, (2) imple-



Obrázek 2. Architektura platformy je následující: UT04, GB04 a Parser na jednom stroji zajišťují běh simulace, propagaci podnětů k agentovi a provedení vybraných akcí. ASM, Klient a IDE mohou běžet na jiném stroji a starají o rozhodování a zobrazují práci agenta uživateli.

mentace modelu, (3) ladění implementace, (4) ladění parametrů modelu a (5) experimentů. IDE Pogamutu2 bylo navrhnuto tak, aby podporovalo poslední čtyři kroky. V této sekci nastíníme, jak je toho dosaženo.

Implementace modelu. Platforma aktuálně podporuje vytváření ASM v Javě, Pythonu nebo v POSHi. Balík tříd Klienta zajišťuje komunikaci s GB04 a nabízí vysokoúrovňové rozhraní pro ovládání agenta. Stará se také o paměť agenta a inventář. IDE obsahuje správu projektů (Obr. 3, okno 1) a editor se zvýrazňováním syntaxe (3.2).

Ladění. Ladění v IDE je složeno z osmi částí, které jsou vhodné pro odstraňování chyb i parametrizaci. Seznam běžících serverů a agentů (3.6) pomáhá se správou více agentů. Introspekce a vlastnosti platformy (3.3) nabízejí rychlý přehled o interních proměnných agentů i platformy. Žurnály (3.4) zobrazují zprávy zaznamenávané v komunikaci mezi Parserem a GB04, interní zprávy platformy a zprávy ASM. Žurnály lze filtrovat podle typu zprávy, navíc jsou podle typů barevně odlišeny. Ruční ovládání agenta umožňuje umístění agenta na konkrétní pozici. IDE nabízí možnost zastavit simulaci a prohlédnout si aktuální situaci ve virtuálním světě společně s žurnály. Zastavení simulace je obsaženo v panelu ovládání serveru (3.5), který dále umožňuje nastavovat rychlost hry, lokaci, měnit pozice jednotlivých agentů či vytvářet nahrávky z aktuální simulace.

Ladění parametrů. Modely jsou typicky velmi citlivé na nastavení mnoha parametrů. IDE umožňuje nastavování parametrů za běhu. Nástroje zmíněné v předchozím odstavci také slouží k rychlejšímu nalezení vhodného nastavení parametrů.

Experimenty. Celková evaluace modelu je možná díky specifikaci experimentů v IDE (lze specifikovat kontrétní lokaci, počet agentů, jejich startovní pozice, vybavení, ukončující podmínku, počet opakování experimentu). Do simulace lze rovněž přidávat breakpointy. Představené vlastnosti umožňují snadnou práci s experimenty a jejich opakování.

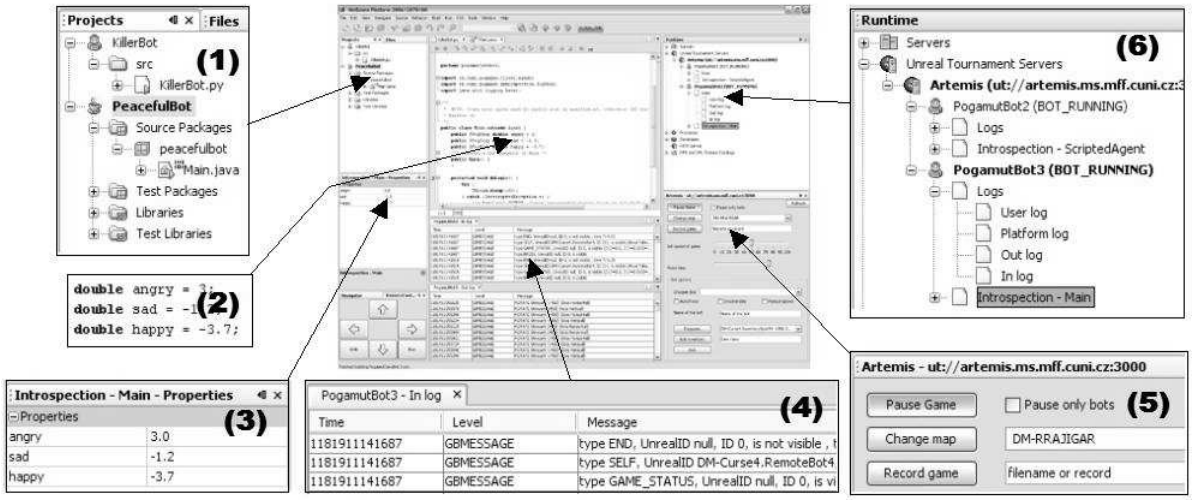
5 Diskuze a probíhající práce

Hlavní přínos Pogamutu 2 spočívá v integrování vývojového prostředí, bohaté knihovně předdefinovaných metod pro design agenta a možnosti vytváření agenta v Pythonu za použití reaktivního plánovače POSH. Nicméně systém má jistá omezení. K serveru lze připojit maximálně 30 agentů, není tedy vhodný pro masové simulace. Dalším aspektem je tok času; rychlost hry lze měnit jen do jisté míry, což činí platformu nevhodnou například pro nasazení evolučních algoritmů.

Nyní pracujeme na implementaci pokročilých nástrojů. Patří mezi ně podpora více skriptovacích jazyků a dalších ASM (např. SOAR [16]), videotutoriály a timeline. Timeline umožní zaznamenat simulaci společně s žurnály. Následně bude možné přehrát simulaci simultánně se zobrazením výpisu ze žurnálů, a tak sledovat přehledně dění v simulaci i změny v ASM agenta. Tento nástroj tak bude užitečný při odstraňování chyb a parametrizaci.

6 Závěr

Představili jsme platformu pro vývoj umělých bytostí v komplexním virtuálním světě počítačové hry Unreal Tournament 2004. Platforma obsahuje knihovny pro tvorbu agenta. Knihovny obsahují moduly pro vnímání, motorická primitiva, paměť, inventář a reprezentací světa. Klíčovou komponentou systému je IDE, které nabízí inspektor proměnných, ovládání serveru, pohledy na žurnály, agenta a paměť, editor skriptů atd. Dohromady tyto moduly tvoří platformu pro rychlé prototypování chování umělých bytostí ve světě UT04.



Obrázek 3. Ukázka aktuální verze IDE v ladícím módu. Popis jednotlivých oken se naléza v následujícím textu.

Platforma je zamýšlena pro pětifázový vývoj agenta. Fáze jsou: definice modelu (specifikace), implementace modelu, odstraňování chyb v implementaci, parametrizace, experimenty. Poslední čtyři jsou podporovány platformou. Vývojář je tak ušetřen zbytečné práce s komunikací, pamětí, reprezentací objektů světa atd. a může se zaměřit přímo na řešený problém.

Platforma je primárně určena pro výzkum. Konkrétně je to výzkum (1) emocí, (2) navigace v prostoru a (3) spolupráce agentů. Dále je možno využít platformu pro výuku. Používání platformy je intuitivní a přímočaré. Obsáhlý soubor tutoriálů a instruktážních videí ještě víc zkracuje startovní čas. Naše skupina zamýšlí použití platformy na MFF UK pro demonstraci praktických problémů umělých bytostí studentům se zájmem o tuto oblast. Snadná práce s platformou může být lákavá pro komunity hráčů, kteří již dnes experimentují se svými oblíbenými hrami [20], [21].

Literatura

[1] Wooldridge M. and Jennings N.R., Intelligent Agents – Theories, Architectures and Languages. In: Volume 890 of Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995

[2] Microsoft: Halo 2. URL: `http://www.bungie.net` [24. 4. 2007]

[3] Tactical Iraqi. URL: `http://www.tacticallanguage.com.` [24. 4. 2007]

[4] Hodges L.F., Anderson P., Burdea G.C., Hoffman H.G., and Rothbaum B.O., Treating Psychological and Physical Disorders with VR. IEEE Computer Graphics and Applications, 2001, 25–33

[5] Mateas M. and Stern A., Façade: An Experiment in Building a Fully-Realized. Interactive Drama. Game Developers Conference, 2003

[6] Cavazza M., Charles F., and Mead S.J., Developing Re-Usable Interactive Storytelling Technologies. IFIP World Computer Congress 2004, Toulouse, France, 2004

[7] Engenuity Technologies Inc.: AI-Implant. URL: `http://www.ai-implant.com` [24. 4. 2007]

[8] Softimage: XSI. URL: `http://www.softimage.com` [24. 4. 2007]

[9] X-AItment GmbH: X-AItment, URL: `http://www.x-aitment.net` [24. 4. 2007]

[10] Adobbati R., Marshall A.N., Scholer A., and Tejada S.: Gamebots: A 3d Virtual World Test-Bed for Multi-Agent Research. In: Proceedings of the 2nd Int. Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada, 2001, URL: `www.planetunreal.com/gamebots` [24. 4. 2007]

[11] Champandard A.J., AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors. New Riders, 2003, URL: `http://fear.sourceforge.net` [24. 4. 2007]

[12] Brom C., Gemrot J., Bida M., Burkert O., Partington S.J., and Bryson J.J., POSH Tools for Game Agent Development by Students and Non-Programmers. Proc. of CGAMES 2006, Dublin, Ireland, 2006, 126–133. URL: `http://ail.jinak.cz` [24. 4. 2007]

[13] Higgins D., Generic A* Pathfinding. In: AI Game Programming Gems (Mark DeLoura, ed.). Charles River Media, 2000

[14] Bryson J.J., Inteligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agent. PhD Thesis, MIT, Department of EECS, Cambridge, MA, 2001
[15] Epic Games: UnrealTournament 2004. URL: http://www.unrealtournament.com [24. 4. 2007]
[16] University of Michigan: SOAR. URL: http://sitemaker.umich.edu/soar/home [24. 4. 2007]
[17] JavaBots. URL: http://utbot.sourceforge.net [24. 4. 2007]
[18] Molineaux M., Aha D.W.: TIELT: A Testbed for Gaming Environments. Proceedings of the Twentieth National Conference on Artificial Intelligence (Intelligent Systems Demonstrations), Pittsburgh, PA: AAAI Press, 2005
[19] Sun Microsystems, Inc: Netbeans. URL: http://www.netbeans.org [24. 4. 2007]
[20] Counter Strike bots. URL: http://www.cstrike.ro/bots.php [24. 4. 2007]
[21] Bots United. URL: http://www.bots-united.com [24. 4. 2007]

Semantic-based analysis of discussions in SAKE *

Peter Butka[1], Ján Hreňo[2], and Marián Mach[1]

[1] Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics
Technical University of Košice, Letná 9, 04200 Košice, Slovakia
Peter.Butka@tuke.sk, Marian.Mach@tuke.sk
[2] Faculty of Economics, Technical University of Košice, Letná 9, 04200 Košice, Slovakia
Jan.Hreno@tuke.sk

Abstract. *In this article we describe an approach for semantic based analysis of discussions within Semantic-based Groupware System (GWS) in SAKE project. SAKE (Semantic Agile Knowledge-based E-government) is a STREP Project sponsored by the European Union starting March in 2006. The overall objective of SAKE is to specify, develop and deploy a holistic framework and supporting tools for an agile knowledge-based e-government that will be sufficiently flexible to adapt to changing and diverse environments and needs. We give a brief overview of the approach that will be applied for evaluation of discussion, discussions threads and discussion posts. Part of the approach is also an analysis of the discussants regarding positive or negative reactions on their contributions, extended by usage of concepts from argumentation ontology. Main goal is to identify some rating of the users involved in the discussion and support moderators in evaluation of discussion result and/or identify discussants skills useful for another similar case discussion.*

1 Introduction

Many organizations work with knowledge resources based on the textual documents and often works also using some communication tools, discussion forum is one of them. Discussion forums (boards) are websites for an online discussion group where users, usually with common interests, can exchange open messages [1]. It typically shows a list of topics people are concerned about. Users can pick a topic and see a thread of messages and replies about it and post their own message.

Forums can be distinguished in very simple and more sophisticated means depending on the purpose of use. For instance they can differ among steps in the discussion process and include polling-elements to focus discussion (and close others) or include the possibility to rate postings. Besides elements to create the "most active participant of the day/week" can be included to attract new user groups, notification elements to support the discursive elements can be implemented to notify participants when their postings were commented. Also various means of authentication can be distinguished - from open, anonymous forums to registration opportunities via digital signature. Besides, forums can be supported by several facilitators and therefore need a sophisticated system of user-rights-management in the backend. Discussion forums are distinguished from chat rooms by structured interaction around the threads and that extend normally over a period of days or weeks rather than hours.

We are proposing the "extraction of the skills" algorithm for the analysis of the users within forums. It will be based on the identification of discussion topic and rating of the users regarding their contributions in threads. The resulting algorithm will combine statistical rating system based on the number of the posts by the users, and the sophisticated element based on usage of the ontologies. In SAKE project a main focus is on the discussion of experts in some expert group, e.g. about new regulation which has to be prepared for usage in government or self-government business process.

SAKE is a STREP Project sponsored by the European Union starting March in 2006. The overall objective of SAKE is to specify, develop and deploy a holistic framework and supporting tools for an agile knowledge-based e-government that will be sufficiently flexible to adapt to changing and diverse environments and needs. More general information about objectives and architecture can be found in [2], some details regarding groupware system in [3]. Some basic information needed for this contribution will be presented in the paper.

* The work presented in the paper is supported by the EC within the FP6 IST 027128 project "SAKE - Semantic-enabled Agile Knowledge-based E-government", by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the project No. 1/4074/07 "Methods for annotation, search, creation, and accessing knowledge employing metadata for semantic description of knowledge", and by the Slovak Research and Development Agency under the contract No. RPEU-0011-06 (project PoZnaŤ).

2 One adaptive approach to analysis of discussions

This chapter shortly describes one approach to the numerical analysis of the discussions and discussants presented in [4]. Author in this work used simple methods for information extraction from texts. The core of the work is based on the discussion threads evaluation algorithm alterations. It provides also the document search system on the basis of lisp programming languages keywords indexed by retrieved email contributor's information. At the end of the work series of experimental results are analyzed and interpreted.

The experiments were realised on discussions from the forum about lisp programming. First, messages have been pre-processed and those with the lisp code have been identified using Naive Bayes and/or heuristic algorithm (based on regular expressions).

2.1 Discussion threads evaluation

Discussion thread is tree structure of discussion contributions (messages), where edges between nodes are relations sender-responder. *Popularity of message* is value of "quality" of contribution in discussion thread according to degree of discussing inside the particular sub-tree of the discussion thread. *Weight of the contributor* is "authority" value of the current contributor within whole discussion group.

Usually, discussion is opened in order to find answer for some question, problem, to interested someone with knowledge about the problem. Two cases are possible: someone is interested and discussion starts, or discussion is stopped because theme is too trivial or difficult for others. Here discussion can be seen also as "voting" for "authority" of contributor to others. More discussed starting contributions lead to higher authority of starting author. Similarly, popularity of contribution grows better with reactions from better rated authors.

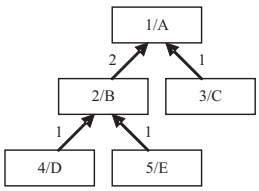


Fig. 1. Example of discussion thread presented in tree structure - starting contribution of author A, reactions from other contributors B, C, D, E. Edges are valued by the weights of the particular contributions - popularity.

Weight of the contributors is changing within growing of whole discussion group by activity of the contributor. Design of the weighting method is based on the incremental acquisition of the contributor's authority in the community. Particular contributions are ordered by the posted date in threads and threads are ordered by the date of their starting contributions.

Threads are analyzed using two procedures. First, actual thread is evaluated according to popularities of the particular contributions. Then, weights of the contributors are adapted according to evaluated thread. Outputs of the process are popularities of particular messages and authorities of contributors for every thread in the collection. In first iteration (in computing of oldest discussion thread) are all weights of the authors set to same value, in this case 1 (start is same for everybody). Authority of contributors depends on the popularity of their contributions. In particular thread popularity of messages is based on the values of author's weights and popularities of messages with reaction to them. Then popularity is computed using:

$$y_j = \sum_{i=1}^{k} w_i \cdot y_i, \tag{1}$$

where y_j is popularity of message j, w_i is weight of contributor and y_i popularity of message i. If node is list of the tree, popularity is 1 (computing starts with list nodes and moves to root). For the computing of authorities change next formula is used:

$$w_k^{(n+1)} = w_k^{(n+1)} + \frac{y_i}{y_R} \cdot \frac{n}{N}, \tag{2}$$

where new weight of contributor k is based on old one plus some adaptive coefficient, y_i is popularity of contribution i, y_R is popularity of root contribution, n is number of contributors in thread and N is number of all contributors in forum.

Based on the weight w_i of contributor i and average of weights $\bar{w}$ of authors in whole discussion group it is possible to compute rating of contributors, which provides information how different is weight of the author according to average weights in whole group. Then rating R (in %) is:

$$R = \left(\frac{w_i}{\bar{w}} \cdot 100\right) - 100. \tag{3}$$

2.2 Advantages and disadvantages of the approach

According to previous analysis author also designed some modifications, where popularity of messages is computed using specific part of the thread, or for update of authority, where average of popularities of whole contributions in thread is used instead of the yR.

Interesting result is design of types of the typical contributors based on the counts of starting contributions by author or reacting ones, which of them contains code, which of them not. For example, let m is number of author's starting discussion (messages which open new thread) and n is number of all author's contributions. Then if we found that m/n is more then 0.75, contributor is someone who very often starts new threads, so he is member of type "often asking" users (as usually new thread is starting because of process of finding solution for some problem). Using this concept it is possible to identify several types of users.

Advantages of this approach are possibility to automatically rank users according to their activity in discussion forum, simple proposal for identification of typical groups of users and evaluation of popularities of particular messages within thread. Basically, it means that there is a method for evaluation of authority of contributors with the outlook of the available documents (messages) as well as some respect of their authors inside the community.

One of the disadvantages of the work was uncertainty in author's identification as they are only identified by email address, which probably has changed for some of the users in evaluated forum during time. Another disadvantage is that context of the message is not used in analysis and also some differentiation of types of reactions inside the thread should be important for ranking of authors, e.g. to have some possibility to evaluate quality of contribution (at least some positive or negative reaction).

According to this we want to extend and/or change former approach and try to avoid consequences emerged from the previous disadvantages. In our system user management of forum will be responsible to right identification of the users and their roles (moderator, group creator). Also we want to extend approach with possibility for users to rank quality or relevance of particular messages, and also connect the messages to some context at least in form of the keywords-based annotations of the messages, so the users will be ranked also according to semantic context (this will be useful when new group is needed and we can try to suggest some experts from the similar cases with their ranking) as well as according to types of reactions on quality and relevance of their contributions.

3 Re-use of approach in SAKE

There are two main reasons for the algorithm presented in the previous chapter not to be sufficient for our needs within the SAKE project. First, there is no difference between posts relevancy which are relevant to the discussion topic and the irrelevant posts. We have to find a way to distinguish between the scoring of the posts according to the semantics of the posts. Secondly, it is very important in the discussions whether the reply is positive or negative to the original post (both can be very relevant). These are two main aspects we want to take into account in the proposal of the algorithm - relevancy and consensus.

As we are using the ontologies for representing knowledge about the topics as well as for the capture of the agile changes in the SAKE environment, we will try to reuse ontologies for our purposes also.

Our first proposal is to use the special ontology for the capture of the relevancy and the consensus properties for the particular posts in the discussion forums. Next, we will develop the algorithms, which will provide the measure of the relevancy for the particular users to particular topics via reasoning over the ontology.

3.1 Argumentation ontology

As stated previously first we need is the sufficient ontology which will store the relevancy and consensus measures for the discussion forums. We have identified the core ontology which seems to be relevant for our proposal. It is the DILIGENT argumentation ontology [5]. The screenshot from the ontology visualisation in the Protege ontology editor is on Fig.2.

We have to modify this ontology to meet particular needs of our user partners in the SAKE project as well as we have to develop the way we will fill in the instances of the ontologies during the discussion evaluation. It can be done automatically or manually by the user.

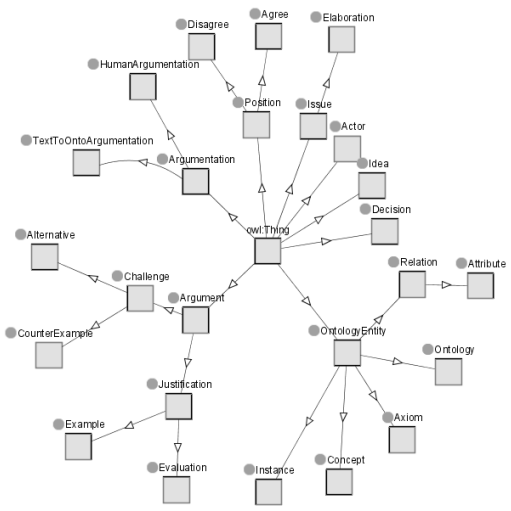


Fig. 2. Protege snapshot of the argumentation ontology.

As there are limited resources for the research in this field in the scope of the process, we will concentrate on simpler, manual approach. In the manual approach it will be the users, who will set the relevance and consensus in the posts. Any user or only selected users (moderators) will be allowed to select the consensus and the relevance in the form of sufficient visualisation of the argumentation ontology.

For the purposes of our project only selection from the list of the predefined values will be possible. Discussion forums in the SAKE project are mainly used for the discussion of the experts during the decision process, so we can assume the moderators are willing to spend the time needed for the manual addition of the values. In the possible future enhancements there is of course the vision to automatically rate the posts according to the semantic analysis of the posted text. As for now, it is out of the scope of our project. But simple keyword-based annotation of messages will help to understand context of contributions.

3.2 Keyword-based annotation of the messages

According to fact that we want also to know context of messages (what they are all about), some annotation of user's contributions is needed. In our SAKE pilot application (e.g. preparing of the new general binding regulation in local authority) there is planned the annotation of the closed cases. These annotations will be used in search for similar older cases, and of course will be reusable also for discussion analysis (if discussions are available).

One of the reusable information is also context of actual case. Actual case in SAKE is for example process of the preparing of new regulation and in the moment of creation and starting of this new process responsible person (e.g. mayor in case of the local authority environment) will input several keywords as a base for the annotation of the case.

On the other hand, annotation is needed also for the actual cases discussions. This can be done again manually or automatically. Here it is different to previous part about using of some argumentation-like functionalities of the discussion forums. It is somehow possible to expect that simple evaluation of the relevancy and quality of the contribution will be done manually, e.g. by selection from the list of reactions. In case of the annotation of messages according to difference in pilots (only keyword-based description of domain in some of them) it is preferred to use automatic keyword-based annotation. This can be achieved by help of the CMS (Content Management System) part of the SAKE - we can use already annotated documents, analyse them and use information which keywords lead to which annotations of documents as the annotation of messages in the forum and threads.

3.3 Reasoning over the argumentation ontology and use of results of analysis

When the argumentation ontology is available, it has to be used somehow to provide us the information it was designed for. Via the combination of the (probably modified) numerical methods described in the chapter 2 and the reasoning over the argumentation ontology we have to propose the users with the coefficient, which will express the relevancy and the consensus between any two of the forums, posts, threads, users, topics. We will be able to answer the questions like: "How relevant is the user to the forum?", "How relevant is the forum to the topic?", or "How consent is the user to the topic?". These questions are still to be identified from the user partners of the project.

Main advantage of the analysis is to provide users by the information about activities and relevancy of users within their discussions regarding different topics and problems. This will lead to possibility to rank users, so the moderator (starter) of a new case can invite better ranking people for participation in a new expert group.

4 Conclusion

This paper describes one simple approach to analysis of discussions and possibility to re-use some concepts of this work in our project. We have identified advantages and disadvantages of the approach as well as parts in which we want to add some novelty based on the semantic elements of the system and opportunities of the project. Proposed ideas will be extended, refined, implemented and tested within groupware component of the SAKE system, which is work-in-progress now.

References

1. Demonet consortium, Report on current ICTs to enable participation. Project deliverable 5.1, 2006
2. Stojanovic N., Mentzas G., and Apostolou D., Semantic-Enabled Agile Knowledge-Based e-Government. In Proceedings of the Semantic Web Meets eGovernment 2006, California, USA, 2006, 27–29
3. Butka P. and Hreňo J., Semantic-Based Groupware System for SAKE. In Proceedings of the 1st Workshop on Intelligent and Knowledge oriented Technologies, WIKT 2006, Bratislava, 2007, 71–73
4. Lukáč G., Information Extraction from Texts (in Slovak). Master Thesis, KKUI FEI TU Koice, 2007
5. `http://diligentarguont.ontoware.org/2005/10/arguonto`

Využití n-gramů pro odhalování plagiátů*

Zdeněk Češka

Katedra informatiky a výpočetní techniky, Západočeská univerzita
Univerzitní 8, 306 14 Plzeň, Česká republika
zceska@kiv.zcu.cz

Abstrakt *Stále rostoucí popularita Internetu a zvyšující se dostupnost různých dokumentů nám přináší i jisté problémy. Jedním z mnoha příkladů je množství pokusů o kopírování cizích prací, s vizí ulehčit si vlastní námahu. To s sebou přináší i rozvoj metod jak plagiátora identifikovat. Tento článek objasňuje metody pro detekci plagiátů a přibližuje náš výzkum v současnosti probíhající na ZČU. Zájem je věnován především metodě využívající n-gramy pro detekci překrývajících se částí dokumentů a odstranění problémů s posunem textu. K extrakci n-gramů vyšších řádů jsou na konci článku porovnány různé metody.*

1 Úvod

Se stále rostoucí popularitou Internetu se zvyšuje množství volně dostupných dokumentů. Pro uživatele Internetu je tak stále snadnější vyhledat vhodný dokument a místo pracného vymýšlení část dokumentu okopírovat a vydávat jej za vlastní práci. Ve školství se jedná o zvláště závažný problém, který je podpořen řadou serverů[1] nabízejících volně ke stažení různorodá témata. To samozřejmě působí nemalé škody, a proto se hledají různá řešení, jak tomu předcházet [1,2].

Ochrany znemožňující tisk a kopírování obsahu dokumentů prostřednictvím CTRL+C jsou nedostatečné. Často se podaří nalézt slabinu v zabezpečení a ochranu obejít. Navíc nejrůznější specializované ochrany obvykle vyžadují instalaci dodatečných nástrojů, což zabrání čtení příslušného dokumentu na počítačích, kde nejsou k dispozici administrátorská práva.

V boji proti plagiátorství je potřeba se vydat cestou vyhledávání dokumentů v rozsáhlých databázích. Vlastní myšlenka ochrany spočívá v psychologii, kdy si plagiátor kopírování rozmyslí, neboť by mohl být snadno odhalen při porovnání s databází již existujících dokumentů. Je zřejmé, že pro co nejvyšší účinnost musí databáze obsahovat velké množství existujících dokumentů. Faktorem úspěchu je též užití efektivních metod, které dokáží v krátkém čase a správně identifikovat plagiát.

* Tato práce byla částečně podporována z prostředků Národního Programu Výzkumu II, projekt 2C06009 (COT-SEWing).

[1] www.schoolsucks.com, www.cheathouse.com, www.seminarky.cz

Metoda relativních frekvencí, představená systémem SCAM [3], je založena na srovnání četností slov mezi dvěma dokumenty. Pozdější metody, jako je například hledání běžných trojic slov v systému Ferret [5], využívají kombinace slov pro nalezení částečných překryvů mezi dokumenty. Nejdále se dostávají metody využívající n-gramy vyšších řádů (5 a více), které identifikují překrývající se části textu a umožňují jejich vizualizaci. Metody postavené na n-gramech, stejně jako předešlé metody, jsou odolné vůči posunu textu uvnitř dokumentů. K němu dochází při mazání, vkládání nebo přepisování částí vět, což jsou obvyklé techniky pro zakrytí okopírovaného textu a znemožňují přesné porovnání textových řetězců.

2 Přehled metod pro detekci plagiátů

2.1 Metoda relativních frekvencí

Metoda založená na relativních frekvencích [3] vychází z tradičního vektorového modelu, který byl upraven pro nalezení podobnosti dvou dokumentů R a S. Do výpočtu jsou zahrnuta pouze slova, která splní stanovenou podmínku

$$\epsilon - \left(\frac{F_i(R)}{F_i(S)} + \frac{F_i(S)}{F_i(R)}\right) > 0, \qquad (1)$$

kde ϵ je konfigurovatelný parametr v intervalu $(2, \infty)$ a $F_i(D)$ je frekvence termu i pro dokument D. Pokud frekvence $F_i(R)$ nebo $F_i(S)$ je pro daný term nulová, podmínka není splněna.

Na základě podmínky (1) stanovíme množinu $c(R, S)$, která zahrnuje všechna běžná slova z obou dokumentů. Koeficient ϵ je tedy toleranční faktor určující zda bude dané slovo zahrnuto.

Mezi dokumenty R a S určíme dvě asymetrické míry podobnosti $subset(R, S)$ a $subset(S, R)$. V případě $subset(R, S)$ uvažujeme na kolik procent je dokument R podmnožinou druhého dokumentu S, kde

$$subset(R, S) = \frac{\sum_{w_i \in c(R,S)} \alpha_i^2 \cdot F_i(R) \cdot F_i(S)}{\sum_{i=1}^{N} \alpha_i^2 \cdot F_i^2(R)}. \qquad (2)$$

α_i představuje váhu asociovanou s výskytem termu i. Podobně odvodíme $subset(S, R)$. Výsledná podobnost je maximum z obou asymetrických měr

$$sim(R, S) = \max\{subset(R, S), subset(S, R))\}. \qquad (3)$$

Po výpočtu je nutné ověřit, zda $sim(R,S)$ není větší než jedna a hodnotu oříznout na interval $\langle 0,1\rangle$.

2.2 Hledání běžných trojic

Tato metoda, prezentovaná na systému Ferret [5], počítá podobnost dvou dokumentů z celkového počtu společných trojic slov. Aplikací Jaccard-Tanimoto koeficientu lze podobnost mezi dokumenty R a S vyjádřit vztahem

$$sim(R,S) = \frac{|triplet(R) \bigcap triplet(S)|}{|triplet(R) \bigcup triplet(S)|}, \qquad (4)$$

kde výstupem funkce $triplet(D)$ je množina všech extrahovaných trojic slov z dokumentu D.

2.3 Karp-Rabin algoritmus

U tohoto algoritmu [4], který je určen pro porovnání textových řetězců, si lze všimnout prvního náznaku užití n-gramů. K urychlení výpočtu se využívá hašovací funkce, jejíž výsledné hodnoty se porovnávají namísto znaků. Pro stanovení minimální shodné délky řetězců, která bude algoritmem registrována, zvolíme okénko velikosti k. Řetězec je reprezentován hašovací funkcí

$$H(c_1 \ldots c_k) = c_1 \cdot b^{k-1} + c_2 \cdot b^{k-2} + \ldots + c_{k-1} \cdot b + c_k, \quad (5)$$

kde c_i představuje ordinální hodnotu znaku a b je libovolná zvolená báze pro výpočet vhodné hašovací funkce. Výpočet následující hodnoty $c_2, c_3, \ldots, c_{k+1}$ lze urychlit použitím předchozích hodnot

$$H(c_2 \ldots c_{k+1}) = (H(c_1 \ldots c_k) - c_1 \cdot b^{k-1}) \cdot b + c_{k+1}. \quad (6)$$

Výslednou podobnost dokumentů R a S určíme z počtu stejných hašů při postupném porovnání všech podřetězců délky k mezi dokumenty R a S.

3 Užití n-gramů pro detekci plagiátů

Jak bylo naznačeno v úvodu, pro odhalení plagiátu je nutné určit společné části obou porovnávaných dokumentů. Na základě společných částí lze vyvodit procentuální podobnost a stanovením prahové hodnoty rozhodnout, zda se jedná či nejedná o plagiát. Často je vhodné ponechat konečné rozhodnutí na uživateli a jako nápovědu zobrazit překrývající se části dokumentů.

Protože práce s celým textem najednou je komplikovaná, metody hledající překryv společných částí, dělí text na množství malých kousků, které lze mezi sebou snadněji porovnat. Ze shody například 300 kousků textu z 1000 lze vyvodit, že překryv mezi dokumenty je 30%.

3.1 Dělení textu

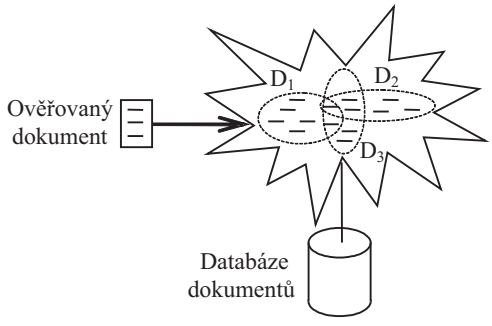
Postupy pro dělení textu na menší části [6] lze shrnout se všemi výhodami i nevýhodami do následujících třech řešení:

1. Dokument je rozdělen dle vět. Nevýhodou je příliš veliký délkový rozptyl, který lze částečně kompenzovat dělením souvětí, ale i tak mohou být některé věty velmi krátké nebo naopak dlouhé.
2. Dokument je rozdělen podle předem definovaného slova nebo funkční hodnoty hašovací funkce, která může být společná pro více slov. Výhodou hašovací funkce je redukce potřebného paměťového prostoru. V tomto případě již nedochází k velkým rozptylům jako u předchozího řešení. Nevýhodou je jazyková závislost, která muže vést v nedosažitelnost požadované dělící hodnoty.
3. Dokument je rozdělen na pravidelné části po n slovech, což zajistí stejnou délku všech vytvářených částí textu. Pro úsporu paměťového prostoru lze opět využít hašovací funkce. Rozdělení textu *"Měna během posledního dne zažila na trhu drastický propad"* na pravidelné části po třech slovech bude vypadat následovně:
   - *"Měna během posledního"*
   - *"dne zažila na"*
   - *"trhu drastický propad"*

Po rozdělení dokumentů na menší části můžeme porovnávat jednotlivé části ověřovaného dokumentu vůči částem dokumentů uložených v databázi. Uchovávány jsou pouze unikátní části textu, které jsou sdíleny příslušnými dokumenty. Při vyhledávání se pro maximální urychlení využívá prostředků databáze a její schopnosti indexovat texty. Schéma příslušné databáze je zobrazeno na obrázku 1.

3.2 Odstranění problémů s posuvy textu

Při kopírování dokumentu nikdo nekopíruje celý dokument 1:1. Obvyklé se jedná o kopírování několika



Obrázek 1. Databáze pro vyhledávání plagiátů.

vět nebo odstavců. Důslednější plagiátoři pak přeformulují část věty nebo alespoň nahradí některá slova. Vkládáním nebo mazáním textu se v dokumentu různě posouvá text. Tato změna ovlivní i tvorbu rozkouskovaného textu. Dojde například ke stavu, kdy v databázi budou uloženy části *"Měna během posledního"* a *"dne zažila na"*, kdežto náš vyhledávaný text bude *"během posledního dne"*. Při vyhledávání se jednotlivé části nepřekryjí a daná část nebude do výpočtu zahrnuta.

Řešením zmíněného problému je vytvořit z textu n-gramy. Při generování n-gramů řádu n, který se shoduje s n užitým pro dělení textu, získáme pro každou část textu všechny možné varianty posunu. Nevýhodou jsou zvýšené nároky na paměť, protože se počet částí textu zvýší n-krát. N-gramy stačí generovat pouze pro jednu ze stran, tzv. pro ověřovaný dokument nebo dokumenty uložené v databázi. Na opačné straně lze text rozdělit po n slovech.

Pro příklad použijeme předchozí text *"Měna během posledního dne zažila na trhu drastický propad"*, který pro úsporu místa převedeme na posloupnost čísel `1 2 3 4 5 6 7 8 9`. Posloupnost rozdělíme na pravidelné části po třech slovech ($n = 3$) a uložíme do databáze `1 2 3 | 4 5 6 | 7 8 9`.

V případě, že text někdo okopíruje a pokusí se o jeho zakrytí, může provést následující tři operace:

a) Odstranění slova z textu
- *"Měna během posledního dne zažila na trhu propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4 5 | 4 5 6 | 5 6 7 | 6 7 9`
- Shoda: `1 2 3` | `4 5 6` | ~~`7 8 9`~~

b) Vložení nového slova do textu
- *"Měna během posledního dne zažila na českém trhu drastický propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4 5 | 4 5 6 | 5 6` **`10`** `| 6` **`10`** `7 |` **`10`** `7 8 | 7 8 9`
- Shoda: `1 2 3` | `4 5 6` | `7 8 9`

c) Záměna slova v textu
- *"Měna během posledního dne prodělala na trhu drastický propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4` **`5a`** `| 4` **`5a`** `6 |` **`5a`** `6 7 | 6 7 8 | 7 8 9`
- Shoda: `1 2 3` | ~~`4 5 6`~~ | `7 8 9`

Z příkladu je vidět, že přidání, odebrání nebo záměna jednoho slova ve větě způsobí maximálně jednu neshodu při porovnávání n-gramů s rozkouskovaným textem. Z toho lze vyvodit, že při dělení dokumentu po n slovech, kde n je stupeň užitých n-gramů, by musela být provedena záměna nejméně každého n-tého slova pro kompletní zmatení algoritmu. V případě delších dokumentů je takováto záměna bez kompletního přepsání téměř nemožná.

3.3 Porovnávání dokumentů s databází

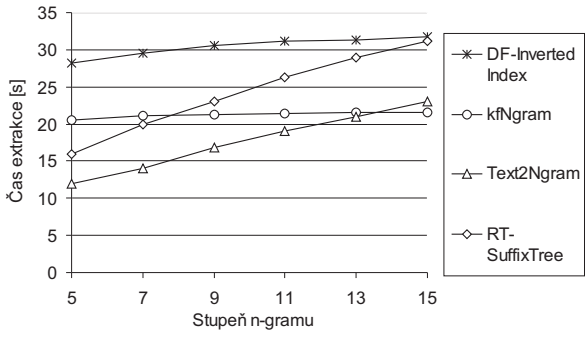
V předchozí sekci byla objasněna metoda porovnávající rozkouskovaný text s n-gramy. Celkem lze odvodit čtyři různé varianty, kde každá má jisté výhody i nevýhody. Tabulka 1 zobrazuje všechny možné varianty. Z uvedených přístupů je pro reálné nasazení a svou úspornost nejvhodnější varianta N:1. Varianta 1:N je vhodná v případě, že preferujeme maximální rychlost, bez ohledu na dostupný paměťový prostor. Poslední varianta N:N je spíše pro experimentální účely, kde se předpokládá hlubší zpracování textu a vyhledávání závislostí mezi dokumenty. Naproti tomu varianta 1:1 je z praktického hlediska nepoužitelná, protože není odolná vůči posunům textu, čímž je identifikace plagiátu značně znesnadněna.

4 Porovnání metod extrakce n-gramů

Náš další výzkum je zaměřen na výběr významných n-gramů, které budou použity pro zvýšení přesnosti detekce plagiátů a zúžení množiny porovnávaných dokumentů. K tomuto účelu jsme provedli srovnání metod pro extrakci různých stupňů n-gramů a výpočet jejich četností.

Do porovnání jsme zařadili aplikaci Text2Ngram [9] implementující metodu Suffix Tree, kfNgram [8] implementující metodu Suffix Array, vlastní implementaci Suffix Tree pojmenovanou RT-SuffixTree [7] a implementaci invertovaného indexu pojmenovanou DF-InvertedIndex [7]. Extrahovány byly n-gramy řádu 5, 7, 9, 11, 13 a 15 z testovacích korpusů o velikosti 5MB a 20MB. Testovací korpusy byly vytvořeny náhodným výběrem vět z korpusu Reuters Volume 1.

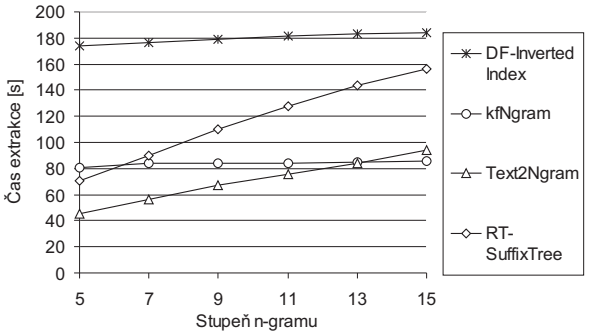
Na obrázcích 2 a 3 jsou zobrazeny výsledky našeho porovnání. K tomu byl použit počítač Intel Core 2 Duo E6600, 2GB RAM a 1TB HDD (2x500GB RAID-0) s instalovaným operačním systémem Windows XP Professional SP2.



Obrázek 2. Extrakce n-gramů z 5MB korpusu.

Varianta	Přístup ověř. dok.	Přístup databáze	Odolnost vůči posunu	Rychlost vyhledávání	Paměťové nároky
1:1	Dělený text	Dělený text	Ne	Vysoká	Nízké
1:N	Dělený text	N-gramy	Ano	Vyšší	Vysoké
N:1	N-gramy	Dělený text	Ano	Nižší	Nízké
N:N	N-gramy	N-gramy	Ano	Nižší	Vysoké

Tabulka 1. Metody porovnávání dokumentů s databází.



Obrázek 3. Extrakce n-gramů z 20MB korpusu.

Z provedených testů je vidět, že metoda Suffix Tree (aplikace Text2Ngram a RT-SuffixTree) je pro výpočet vyšší řádů nevhodná. Její výpočetní čas lineárně roste pro zvyšující se řád n-gramů. Naproti tomu metoda Suffix Array (aplikace kfNgram) má téměř konstantí průběh pro vyšší řády. Metoda invertovaného indexu (aplikace DF-InvertedIndex) má podobný průběh, ale je několikanásobně pomalejší než Suffix Array.

5 Budoucí práce

Současnou metodu využívající n-gramy pro detekci plagiátů zamýšlíme zpřesnit bonifikací významných n-gramů během výpočtu. Kromě toho plánujeme provést testování vlivu různých délek n-gramů a jejich kombinací v prostředí českého jazyka.

Pro detekci plagiátu je nutné provést porovnání se všemi dokumenty v databázi. Našim dalším cílem je tvorba otisků dokumentů, které by zúžily množinu porovnávaných dat. Též předzpracování, omezující množství dokumentů na témata vztahují se k ověřovanému dokumentu, by zrychlilo výpočet.

6 Závěr

V tomto článku jsme nastínili problematiku plagiátorství a využití n-gramů pro odhalování plagiátů. Zároveň byly objasněny různé varianty aplikace n-gramů na straně ověřovaného dokumentu a databáze, včetně jejich výhod i nevýhod. Pro extrakci vyšších řádů n-gramů a výpočet jejich četností jsme provedli srovnání metod Suffix Tree, Suffix Array a invertovaného indexu. Z výsledků je patrné, že metoda Suffix Array dosahuje téměř konstatního času pro různé řády, a proto je lepší volbou pro naše experimentování. Náš další výzkum se ubírá směrem k bonifikaci významných n-gramů během výpočtu. Pro rychlejší detekci plagiátu zamýšlíme tvorbou vhodných otisků omezit množinu porovnávaných dat.

Annotation

N-gram utilization for plagiarism detection

Growing popularity of Internet has brought the possibility to download a lot of different documents. This paper describes some common methods relating to the widely spread plagiarism. Employing n-grams is discussed in detail to detect overlapping documents and to avoid issues caused by text shifting. At the end of this paper we compare various methods for higher n-gram sizes extraction.

Reference

1. Clough P., Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies. Internal Report CS-00-05, Department of Computer Science, University of Sheffield, 2000
2. Clough P., Old and New Challenges in Automatic Plagiarism Detection. Plagiarism Advisory Service, 10, 2003
3. Shivakumar N. and Garcia-Molina H., SCAM: A Copy Detection Mechanism for Digital Documents. In Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, Austin, 1995
4. Charras C. and Lecroq T., Handbook of Exact String-Matching Algorithms. King's College London Publications, 2004. ISBN 0-9543006-5-3.
5. Lane P., Lyon C., and Malcolm J., Demonstration of the Ferret Plagiarism Detector. In Proceedings of the 2nd International Plagiarism Conference, NewcastleGateshead, 2006
6. Pataki M., Plagiarism Detection and Document Chunking Methods. The Twelfth International Word Wide Web Conference, Budapest, 2003
7. Tesar R., Fiala D., Rousselot F., and Jezek K., A Comparison of Two Algorithms for Discovering Repeated Word Sequences. The 6th International Conference on Data Mining, Text Mining and Their Business Applications, Skiathos, Greece, 2005. ISBN 1-84564-017-9.
8. Fletcher, W.: kfNgram. `http://www.kwicfinder.com/kfNgram/`
9. Zhang, L.: Text2Ngram. `http://homepages.inf.ed.ac.uk/s0450736/ngram.html`

Experimental platform for the Semantic Web*

Jiří Dokulil, Jaroslav Tykal, Jakub Yaghob, and Filip Zavoral

Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic
{jiri.dokulil, jaroslav.tykal, jakub.yaghob, filip.zavoral}@mff.cuni.cz

Abstract. *The Semantic Web is not widespread as it has been expected by its founders. This is partially caused by lack of standard and working infrastructure for the Semantic Web. We have built a working, portable, stable, high-performance infrastructure for the Semantic Web. This enables various experiments with the Semantic Web in the real world.*

1 Suffering of the Semantic Web

The main goal of the Semantic Web is to create a universal medium for exchange of data. The Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently [1].

Unfortunately, it seems, this goal has not yet been reached, albeit years of research by numerous researchers and large number of published standards.

We believe the Semantic Web is not yet widespread due to three prohibiting facts: missing standard infrastructure for Semantic Web operation, lack of interest from significant number of commercial subjects, and last but not least absence of usable interface for common users.

A nonexistence of a full-blown, working, high-performance Semantic Web infrastructure is one of the main disablers for Semantic Web common use. Whereas the "old web" has clearly defined infrastructure with many production-ready infrastructure implementations (e.g. Apache, IIS), the Semantic Web has only experimental fragments of infrastructure with catastrophic scalability (e.g. Sesame, Jena).

Current standards of the Semantic Web do not allow it to be used by common users. Whereas any user of WWW can easily navigate using hyperlinks in an available, production-quality WWW client, a Semantic Web user currently has only a choice from a set of complicated query languages (e.g. SPARQL [8], SeRQL [3]). These query languages are not intended for casual users, only small number of people are able to use them, which is one of the factors that prevent spreading of the Semantic Web.

We have concentrated our effort to elimination of the first problem and we will describe our implementation of the Semantic Web infrastructure.

Next chapters are organized as follows: after an overview of the infrastructure there is a description of gathering and parsing documents from web in section 3. Section 4 describe import interfaces and their implementation. Section 5 deals with semantic querying and query evaluation. Final two sections contain performance comparison and conclusions.

2 Semantic Web infrastructure overview

We have recognized and described the problem of a missing, standard infrastructure for the Semantic Web in our article [10], where we have proposed a general Semantic Web infrastructure. During the last year we have made a significant progress: we have implemented full-blown, working, fast, scalable infrastructure for the Semantic Web.

The figure 1 depicts the overall scheme of our infrastructure. In this picture rectangles represent active processes, diamonds are protocols and interfaces, and grey barrels represent data-storages. All solid-line shapes depict implemented parts of our infrastructure, whereas all dashed-line shapes represent possible experimental processes. We have designed and implemented an object oriented library for a data-storage access independent on background data-storage implementation.

2.1 SemWeb repository

The heart of our infrastructure is the SemWeb repository. It is responsible for storing incoming data, retrieving results for queries, and holding the ontology used for the data. It consists of the SemWeb data-storage, which is responsible for holding Semantic Web data in any format (e.g. any relational database, Kowari). Import interface enable fast, parallel data storing, and hide details about background

* This research was supported in part by the National programme of research (Information society project 1ET100300419).

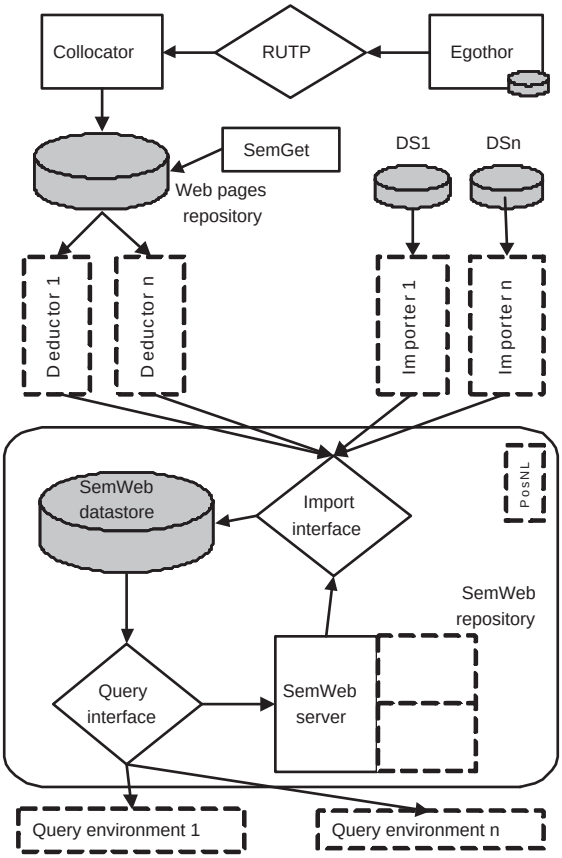


Fig. 1. Infrastructure overview.

SemWeb data-storage import capabilities. It will be described in greater detail in the chapter 4. The query interface has two difficult tasks: to be independent on a query language or environment and to be independent on the SemWeb data-storage query capabilities. More about this part of our infrastructure in the chapter 5. The last part of the SemWeb repository is the SemWeb server. It is a background worker that computes inferences, makes data unifications, and fulfills the task of a reasoner as well. It utilizes import and query interfaces for data manipulation.

We believe that we do not need to have accurate data and computed inferences in the moment of the data import. Just like the real world, the knowledge about the world changes at each moment and we are not able to catch it in one snapshot. Therefore post-processing data by the SemWeb server and computing some additional data in background is acceptable and feasible.

2.2 Import paths

We have two distinguishable sources of data. The simplest one is a data import through importers from external data-storages. The task of importers is mapping external data-storage data-scheme to the SemWeb repository ontology. The second source of data crawls the wild Web using the Egothor web crawler [6]. More about this path in the next chapter.

2.3 Query environments

Query environments present outputs from SemWeb repository. They make queries using query interface of the SemWeb repository and present results of the queries to users in any feasible manner. As an example, we have implemented a SPARQL compiler, which translates a subset of SPARQL queries to our query interface requests.

2.4 Portability

We have implemented nearly all parts (excluding Egothor implemented in Java) in ISO/IEC 14882 C++, which brings full portability among different systems and compilers. Moreover, it allows us to implement bindings to other broadly used languages e.g. Java or C#.

3 Retrieving web documents

In the original proposal [10], there was a direct Egothor plugin for semantic data and metadata acquisition. This runtime structure would have several disadvantages:

- The plugin dedicated to semantic experiments runs within the same environment as a general-purpose web robot. It may cause deficiencies in stability and performance.
- Many semantic experiments need to apply several algorithms to the data set. Multiple data acquisition would cause unacceptable load to both the extractor and data providers.
- The web robot could not be dedicated to semantic data acquisition - it executes tasks for a lot of clients.

We have decided to separate the web gathering and the semantic processing, both in time and space. The current infrastructure implementation is depicted on figure 1.

The document is converted into a stream of SAX events which enables us to process its internal structure more comfortably. This stream is then sent by the Robot UDP Transmitter Protocol (RUTP) [5] to the Collocator. This server reads document requests, converts them to RUTP commands, sends them to the robot, receives streams of SAX events, completes them, computes tree indexes and in case of successful transmission stores each parsed document into the

database. All these activities are driven by SemGet module, which is basically an API wrapped by a command-line user interface.

The database stores each document in a structure similar to a XML Region Tree [7] or a NoK pattern tree [11]. The main feature of this structure is query efficiency - for a given element, all of its ancestors or descendants in an element set can be queried with optimal cost.

4 Import interfaces

This chapter gives a brief overview of data structures and functions used to import data into SemWeb data store.

4.1 Data structures

The basic property of the import interface is a definition of internal memory structure for data insertion (sometimes called RDF Graph) and functions which provide connection to the data store. The internal memory structure can be filled by both RDF triples and reifications. The content of this structure is periodically saved into the data store.

Every data store that supports this interface should implement at least:

- *InitializeConnection(repository_name, user_name, password, parameters).* This function initializes connection to data store. We assume that every data store is identified by at least name, login name and password. Other data store specific parameters can be inserted into the last argument.
- *InitializeInserts(Import type).* This function initializes an insert into data store. Parameter "Import type" determines whether batch insert is used.
- *FinishInserts().* This function finishes an insertion and propagates all triples into the data store.
- *InsertTriple(triple).* This function inserts a triple into the internal memory structure.

4.2 Implementation

We have implementation the interfaces in C++. The data is stored in an Oracle relational database [4].

Import type Some components (e.g. SemWeb server) query the data and when they deduce some formerly unknown knowledge, they insert the information back into the data store. These information is typically represented by a small number of triples, because the quality of these information is more important then their quantity. Other components (e.g. Importers) insert large amount data into the data store. The goal for these components is to import data quickly.

Conclusion: The import interface for any data store should support two modes:

- *Insert immediate*, where data are inserted immediately when InsertTriple function is called.
- *Batch insert*, inserts data into a temporary space and after the import is finished, the whole data is saved into the data store.

Our implementation supports both modes.

4.3 Local cache

Due to performance optimization we had to add a cache for inserted triples into the import interface. The cache is usable with data stores based on both relational database and other data stores with remote access.

In the case of relational database based data store, it is better to insert triples in shorter transactions, but do not commit the transaction after each inserted triple. It prevents extensive record locking and too long response times produced by frequent commits.

In case of other types of data stores, caching can help mitigate negative effect of high network latency. The interface can send more triples in one request and thus eliminate useless waiting for the network.

5 Query interface

We have developed an interface to query the data. The interface is based on simple graph matching and relational algebra. Simple graph matching allows only one type of query. It consists of a set of RDF triples that contain variables. Result of the query is a set of possible variable mappings. This set can easily be interpreted as a relation with variable names used as a schema for the relation.

Relational algebra operations (e.g. joins or selection) are used on the relations created by simple graph matching. These operations are widely known from SQL, which was a major argument for this choice. Database developers are already familiar with these operations and a lot of work has been put into optimizing these operations.

So far, we decided to support only some of the common relational operations. Since the schema of elementary relations (results of basic graph patterns) consists of variable names and so it is defined in the query and not in the database schema, we use only natural joins. Variable names are used to determine which columns should the join operation operate on.

5.1 Selection

Selection operation revealed several problems specific to RDF querying. While in traditional relational algebra it is easy to maintain type information for each column, it is not possible in RDF. Even a simple query can produce result that contains values with different data types in one column.

With this in mind, we have to consider behavior of relational operators when it comes to different data types. For instance, in SPARQL [8] the operators should consider data types for each value separately, so one operator in one query compares some values lexicographically by their string value and some other values numerically by their value.

This is a serious performance problem that for instance makes it impossible to use indexes to evaluate expressions like $x < 5$ and especially $x < y$. On the other hand, such behavior is often not necessary because the user has certain idea about data type of x in $x < 5$. So we decided to make the type information part of the query. Then $x <_{integer} 5$ yields true for integers smaller than 5, false for integer greater or equal to 5 and error if x is not integer. This error always removes the whole row from the result.

This definition makes translation of queries to SQL more simple and efficient since it can be easily evaluated by functional index that stores integral values. Conditions like $8 < x \; and \; x < 10$ can be evaluated by simply traversing small part of this index.

5.2 Query language

We decided not to create yet another SQL-like query language. Since the query interface is intended to be used not by people but rather software, the query interface is actually a set of classes. A query is basically just a tree of instances of these classes. Had we created a query language, our form of query would basically be a derivation tree of a query in that language.

An example of a simple query tree:

- Natural join
  - Left natural join
    - Basic graph pattern P1
    - Basic graph pattern P2
  - Basic graph pattern P3

In this query we have three basic graph patterns P1, P2, and P3, that each produce a relation. Then the relations produced by P1 and P2 are joine using natural left join and the result is joined using plain natural join with the result of P3.

5.3 Query example

Following C++ code shows a simple query that outputs first and last name of all people that have both of them and whose last name is either "Tykal" or starts with "Dokulil".

```
Triples triples; triples.push_back(Triple(
  Variable("x"),
  URI(L"http://example.org/last_name"),
  Variable("y")
)); triples.push_back(Triple(
  Variable("x"),
  URI(L"http://example.org/first_name"),
  Variable("z")
)); Table *query_tab=
  new Filter(
    BasicGraph(triples),
    OrExpression(
      TestSubstringExpression(
        NodeExpression(Variable("y")),
        NodeExpression(Literal(L"Dokulil")),
        true, false
       ),
      EQExpression(
        NodeExpression(Variable("y")),
        NodeExpression(Literal(L"Tykal"))
      )
    )
  );

std::vector<Variable*> vars; vars.push_back
(new Variable("y"));
vars.push_back(new Variable("z")); Query
query(vars,query_tab,false); Query query_table
(query);
```

The query consists of a basic graph query with two triples and three variables. Then a selection is applied to the result and finally a projection is used to return only columns with first and last name.

5.4 Query evaluation

We did not want to limit ourselves to just one system for data storage. Since the beginning of development we have been using four different data storages with several other in mind. Each of the systems offered different query capabilities from just evaluating all stored RDF triples to sophisticated query languages.

If we placed strong requirements on query capabilities of those systems, we could use only a few of them. On the other hand, we wanted to provide a more sophisticated query interface than that provided by all of the systems. Furthermore we wanted to use as much of the systems' query capabilities as possible.

The contrast between complex query interface we wanted to give to the user and only very basic query

capabilities provided by data storage system made it obvious that our system must be capable of evaluating the queries itself.

By implementing all operations within our system, we have reduced the requirements for the data storage engine to just one. The engine has to be able to list all stored triples. Thus the system is capable to use extremely simple storage engine that does nothing but read RDF triples from a Turtle file [2].

One of the other storage engines is a Oracle database. It would be highly inefficient to use the database only to list all triples, since we want to make some relational algebra operations on the data and Oracle have spent years optimizing these operations.

As a result, our system is capable of evaluating any query itself, but tries to use any help the storage engine can provide. This makes adding a new storage engine very simple since all that has to be done is implement listing of all stored triples. The same goes for adding new features to the query interface. Once the feature is implemented in our system, it is immediately available with all storage engines. Of course, performance of query evaluation will probably be suboptimal.

5.5 Evaluation algorithm

Since every storage engine can have specific capabilities and limitations, we could not establish a set of rules to decide, what can be evaluated by the engine and what has to be evaluated by our system. Thus each engine must also contain an algorithm to determine whether it is able to evaluate a query.

For query Q the evaluation plan is found like this:

- If the storage engine can evaluate Q then this evaluation plan is used.
- If Q is basic graph pattern then it is decomposed into individual triples, evaluation plan for each triple is found and the results are joined together by our system.
- If Q is an algebraic operation, evaluation plan for each operand is determined and the operation is applied to the results by our system.

The limitation of this algorithm is, that it does not try to rearrange the query in order to achieve better performance either by choosing a more efficient evaluation plan or by allowing the storage engine to evaluate greater part of the query itself, which will probably be more efficient. This problem is a more complex version of optimization of relational algebra expressions and will be a subject of our further research.

6 Performance tests

6.1 Test environment

The test environment consist of two machines. The first one hosts a Oracle database server (2x CPU Xeon 3.06 GHz with hyper-threading, DB instance was assigned 1.0 GB RAM) and the second one is an application server (2x CPU Quad-Core Xeon 1,6 GHz, 8GB RAM).

All tests used large data containing 2.365.479 triples (303 MB Turtle [2] file).

6.2 Data import

Implementation showed us several bottle-necks of our solution and helped us improve the performance. The first touchstone we focused on was the speed of data load into the data store. If we compare data stores from [9] then our test implementation has very good results.

Data were loaded in 100k triples batches. The whole load took 1 hour and 54 minutes, out of which 1 hour and 14 minutes were spent transferring data from source data file to temporary tables in the database and other 18 minutes were spent on cleanup actions.

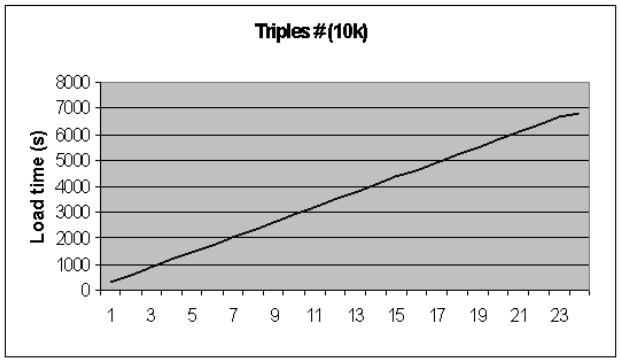


Fig. 2. Average load time 2,3 M triples into SemWeb data store.

Our implementation has almost linear dependency on the size of the input data. Figure 2 shows that our implementation has better scalability than a Sesame-DB data store.

Comparison with Sesame-db repository The main goal of this test was to compare the SemWeb repository with an existing solution based on a relational database. The Sesame-DB repository was connected to a local instance of Oracle database. The

SemWeb repository was connected to the same instance. We tried to load 150 000 triples into each of them. The SemWeb repository loads this data in 780 seconds. The Sesame-db finished loading near 118 000 loaded triples. The error was low space in the TEMP tablespace (512 MB).

The SemWeb repository was primarily designed to work with a huge semantic data whereas the Sesame-db was probably designed to store only smaller semantic data. So the Sesame database schema and SQL statements are not written appropriately for loading of a large amount of data.

6.3 Query performance

Although we have tried to implement the algorithms used in query evaluation in an efficient manner the algorithms themselves are only basic versions so the performance of the query evaluation leaves a lot of space for improvement.

We have tested three different storage engines:

- BerkeleyDB based storage that stores triples in a B-tree
- fully in-memory engine
- Oracle-based RDF storage

Since most other RDF storages are implemented in Java, using them as a storage engine in an efficient manner is not an easy task and will be addressed in the future.

First, we measured the performance of evaluation of the query presented in section 5.3.

The BerkeleyDB-based storage engine required 1.8 seconds to complete the query, while in-memory engine took only 0.7 seconds. The performance of Oracle-based engine was the worst, requiring 6.4 seconds.

We have expected these results. The current in-memory engine is read-only and is optimized for best performance in queries similar to the one we tested. On the other hand, we used the Oracle database only to provide us with plain RDF triples and performed the join operations in our system. But this is not the main reason for the bad performance. The problem is, that the Oracle database is placed on another server and network delays for each returned triple add together. Had we used the Oracle database to join and filter the results the performance would have been much better due to smaller network trafic and better optimization of joins in Oracle. Our measurements showed that time required to evaluate this query is around 0.2 seconds.

7 Conclusion

We have implemented and thoroughly tested the infrastructure for gathering, storing and querying semantic data. We have focused our efforts on effectivity, extensibility, scalability and platform independency. Both our experiences and benchmarks show that this goal is feasible. Our infrastructure is currently used as a platform for further semantic web research. We expect to enhance both interfaces and functionality to support these semantic experiments. As a long-term goal, we plan to interconnect diverse semantic repositories, possibly with different implementation. Such interface-based loosely coupled network could become a nucleus of really usable semantic web, both for academic and practical purposes.

References

1. W3C Semantic Web Activity Statement, 2001. http://www.w3.org/2001/sw/Activity
2. Beckett D., Turtle - Terse RDF Triple Language, 2004. http://www.dajobe.org/2004/01/turtle/
3. Broekstra J. and Kampman A., SeRQL: A Second Generation RDF Query Language. In Proceedings of the Workshop on Semantic Web Storage and Retrieval, Netherlands, 2003
4. Dokulil J., Tykal J., Yaghob J., and Zavoral F., SemWeb Semantic Repository. Technical Report, Faculty of Mathematics and Physics, Charles University, 2007
5. Galambos L., Robot UDP Transfer Protocol, 2007. http://www.egothor.org/RFC/RUTP-v02.pdf
6. Galamboš L., Dynamic Inverted Index Maintenance. International Journal of Computer Science, 1, 2, 2006
7. Jiang H., Lu H., Wang W., and Ooi B.C., Xr-Tree: Indexing XML Data for Eefficient Structural Joins. In Proceedings of the 19th International Conference on Data Engineering (ICDE03), IEEE, 2003, 1063-6382/03
8. Prud'hommeaux E. and Seaborne A., SPARQL Query Language for RDF. W3C Working Draft, 2005, http://www.w3.org/TR/2006/WD-rdf-sparql-query-20060220/
9. Wang S., Guo Y., Qasem A., and Heflin J., Rapid Benchmarking for Semantic Web Knowledge Base Systems. Technical Report LU-CSE-05-026, CSE Department, Lehigh University, 2005
10. Yaghob J. and Zavoral F., Semantic Web Infrastructure using DataPile. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Itelligent Agent Technology, Los Alamitos, California, IEEE, 2006, 630–633, ISBN 0-7695-2749-3
11. Zhang N., Kacholia V., and Tamer M., A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML. In Proceedings of the 20th International Conference on Data Engineering (ICDE04), IEEE, 2004, 1063-6382/04.

Personal telemetric system - Guardian

Dalibor Janckulík, Jan Martinovič, Ondřej Krejcar, and Petr Vašíček

VŠB - Technical university of Ostrava, Faculty of Electrical Engineering and Computer Science
{dalibor.janckulik.st1,ondrej,krejcar,jan.martinovic}@vsb.cz, petr.vasicek@centrum.cz

Abstract. *This project deals with the problems of utility of mobile equipment working in the biomedicine field, particularly telemedicine. This field is relatively new; it focuses on the observation of the life functions from a distance. Practically developing system works with the ECG sensor connected to mobile equipments, such as PDA / Embedded, based on operation system Microsoft Windows. The whole system is based on the architecture of .NET Compact Framework, and other products, such as SQL Server by Microsoft too. This work also deals with the comunication of mobile equipments with sensors and with the server via Bluetooth, WiFi, and GPRS/EDGE. The mobile equipment used serves primarily for measuring and the processing of data from the sensors and their visualization as a graph. The data are also given to the server for further processing and the analysis of the current health of the patients, due to small efficiency of the mobile equipments[1]. The main task we deal in the server part of application is receiving of the data via web services and further processing, management and analysis of this data. For the analysis of received data and further evaluation of the electrocardiogram, there is a self-organizing neural network [2].*

1 Motivation

The basic idea is to create a system which controls important information about state of wheelchair bound (monitoring of ECG and pulse in early phases, then other optional to patients monitoring in hospitals or people working in extremely hard conditions. The biggest limitation is the accessibility of measuring devices in acceptable and adaptive size or comfortable to have about one [1].

Measured device was tested in extreme conditions in a cryogen room [1] in Teplice (-136C), where the final system will be installed. Implementation of the data transmission security was not solved. Whole system is classified as "work in progress" system and it is in testing phase where we found mistakes and repaired them.

2 System architecture

This system consists of several relative-connected parts that can communicate among themselves, so they can approach their function. These parts are: measuring sensor, PDA or embedded device, and server for data process and evaluation. Beside these, most important parts that are worth mentioning are various kinds of accessories, such as GPRS, WiFi, GSM, Bluetooth or GPS modules. By means of them we can communicate. Mostly we use the fastest technology in signal coverage.

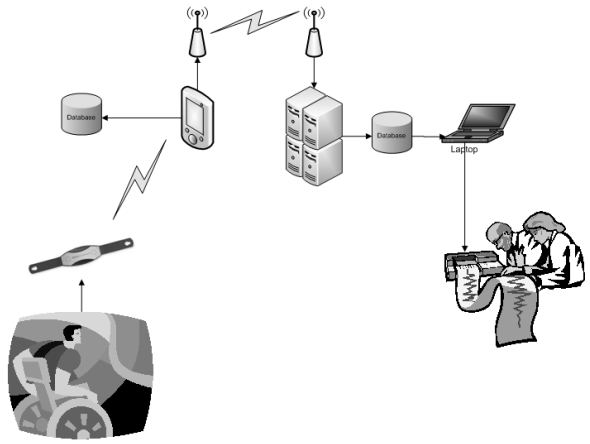


Fig. 1. Measurement schema.

Data acquisition and data transmission are the most important parts of this system. Responsible for correct working are: correct sensor configuration, sensor calibration, data transfer synchronization, and relative communication between sensors and data receiver.

The administrator can add or remove a new patient by means of a PDA program. This step is represented in a server with WEB service. Patient's data is removed or added into a database file.

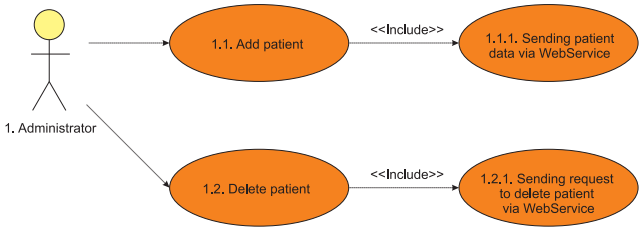


Fig. 2. Administrators possibilities.

[1] Cryogen room - room with very low temperature, nowadays using low temperature for medical acts

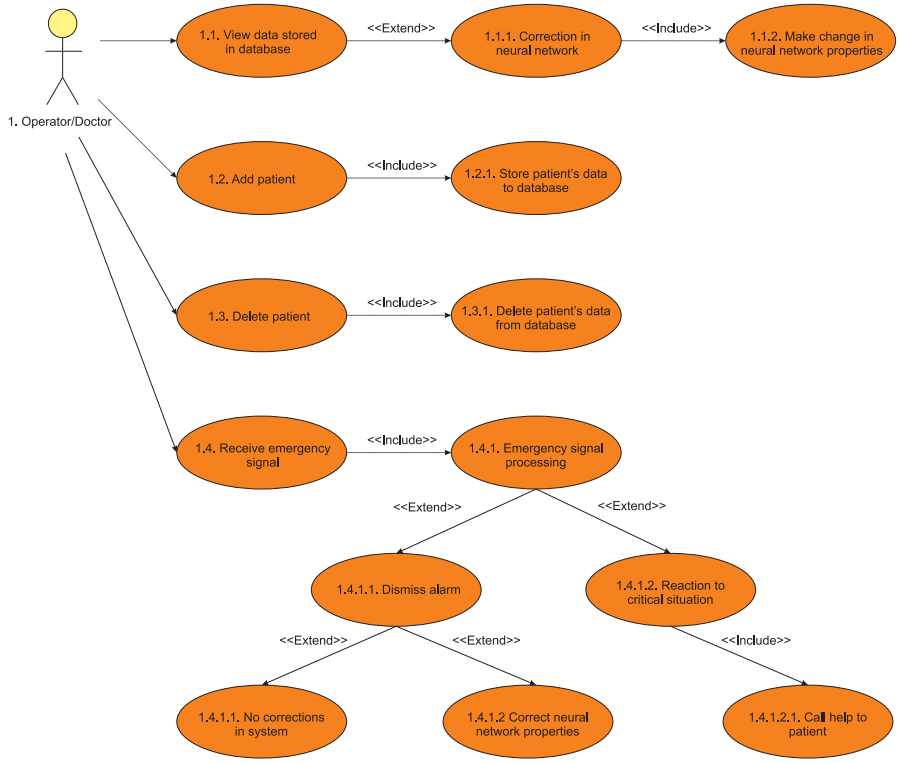


Fig. 3. Operators/Doctors role.

System can display saved data from database file. A doctor can configure or set a neurone network. A change of that is shown in the XML file enshrining, where the neurone network setting of the patient is kept.

The doctor receives the information about downgrade of patient's status. In the case of doctor's reaction, he sends for an expert assistance, such as a helicopter or an ambulance. In case of false alarm, he can configure a neurone network or leave it unchanged, if that was a sporadic incorrect interpretation.

The application is communicating with an ECG Measurement Unit (Corbelt or Blue Keg) through a virtual serial port using wireless Bluetooth technology. Then, after pushing a button, all necessary parameters are set and the communication may begin. Measured data are stored in a Memory Card to a database in MS SQL Server 2005 Mobile Edition [1]. During implementation, several problems were found, resulting from faults and simplicity of the .NET Compact Framework architecture.

- Communication - the Framework SP1 is needed for support of Virtual Serial Port for HP devices with Windows Mobile 2003 operating system.
- Virtualization - graph visualization without Draving.2D nape space.
- Data storage - writing speed, data storage demands, automatic incrementation of the ID of a data record.

The performance of available devices seems insufficient for sequential access; parsing of incoming packets

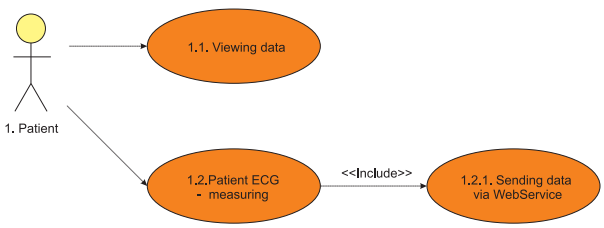


Fig. 4. Patients role.

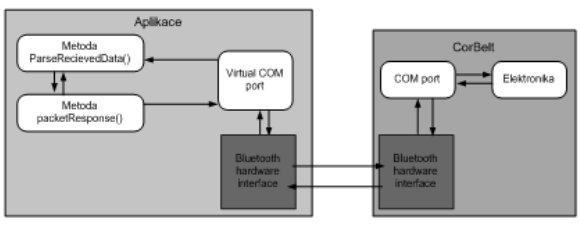


Fig. 5. Bluetooth communication.

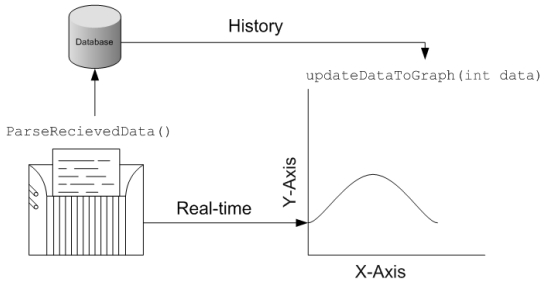


Fig. 6. Packet parsing.

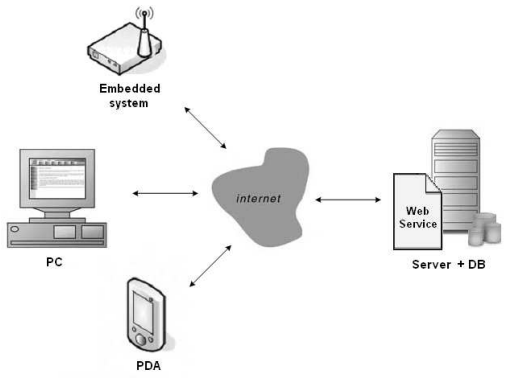


Fig. 7. Web service communication.

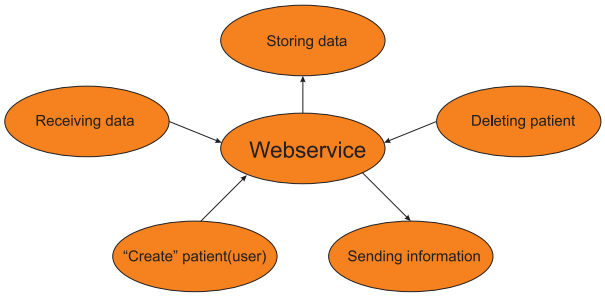


Fig. 8. Web service possibilities.

is heavily time-consuming. Pseudo paralleling [2] is required. If Windows Mobile OS versions 2003 to 5.0 are used, the processing of data from a professional EKG is not realizable due to thread count limitations. A newer operating system can be used to solve this.

Current application is highly specialized and written to accommodate specific hardware. Usage of any other hardware is not possible. This is due to different methods of packet folding, which are unique on each device [3]. This is partly caused by the length of the Telemedicine branch. Operating of the device is most simplified with the least possible count of steps regarding user registration, measurement device connection and the measurement itself.

2.1 Server part

In order to run a server, an operating system supporting IIS is needed. IIS is an Internet Information Server application allowing users to connect to the web server by the well-known HTTP protocol.

The web service transfers data between the server and PDA / Embedded devices. It reads the data, sends acknowledgments, stores the data to the database and reads them from it. The service is built upon ASP.NET 2.0 technology. The SOAP protocol is used for the transport of the data, which are in XML format. That is an advantage since it allows the communication of multiple different technologies and platforms.

The Wireless ECG approaches a real professional ECG with data rate as high as 800 records per second [3]. That makes 48 000 records per minute and 2,880,000 per hour. Considering 100 patients, the value gets to 288,000,000 records per hour. Even if the server accepted only 50 records per second, the sum of records for 100 patients per hour would be 18 million records. That is an extreme load for both server and the database system; hence a better way of storing data is needed.

[2] pseudoparallel - process switching

Methods that devices communicating with the web service can use include:

- receiving measured data
- receiving patient data
- deleting of a patient
- patient data sending

To observe measured data effectively, visualization is needed. A type of graph as used in professional solutions is an ideal solution. To achieve this in a server application, a freeware Zed Graph library can be used.

For data analysis, neural nets are a convenient solution. However, there are problems in the automatic detection of critic states. Every person has its own specific ECG pattern. What is completely normal for one person can indicate crisis for another. The Neural net has to learn to distinguish critical states of each patient separately.

The basic characteristics of a neural net:

- 10 x 10 neurons
- values with decimal point values for precise results
- the learning is based on 3-4 minutes of recording, which is approximately 36000-45000 recorded values
- incorrect values are filtered out
- filtering decreases the amount of values to about 10%, which is still well enough for learning

- the learning cycle with 4200 values takes approximately 30 seconds (CPU PIII 1.2 GHz) with C# implementation
- after each data receiving, the net responds with YES/NO (accepted/declined)
- interruption can be invoked on demand, thus making correction of the net by a new value [2]

To make the specialist's or operator's intervention possible, the system must dispose with a user-friendly interface, possibly imitating those on medical appliances. This area is still in early development.

3 Data analysis

The data acquired from measuring device are incomprehensible for a man, because they are represented by a HEX [3] format packet. The data needs to be stripped of redundant information like packet numbering and transformed to a recognizable state - a graph.

This is done during the packet parsing in PDA / Embedded device, where HEX information 2 bytes in length transform to a binary state, where the data are carried on the first 12 bits. They are transformed to a decadic state and are sent to further processing - sending the data to the server, storing them to a local database or visualization.

The visualization using PDA / Embedded is just a simple visualization of a spline in a real-time or historical graph. In the server application, the visualization is far more complex, with the possibility of storing the current spline as an image, printing, or zooming.

Neural net of the SOM [4] [6, 7] type on the server is 10x10 in size. The initial weight of each neuron is random. The weighs are assigned progressively by learning. Finally, the whole structure is ready to accept data to analyze. Each patient has his own neural net stored on a server in XML format.

4 Summary

The evolution of Telemedicine is unstoppable and apparent; therefore ways need to be found to improve the quality of hospital services, spa services or hazardous environment workplaces. The area of software products working on embedded devices in hazardous environments or PDAs in personal healthcare is still opened and unoccupied. Personal healthcare products are freely available [4] and with minor modifications on hardware they can be used for data acquisition using wireless Bluetooth or ZigBee technologies. By means of data transfers, the acquired data can be gathered in database systems providing access for our personal doctor, who can be alerted in case of trouble.

Why are there systems available to protect our properties and not our health? Is it not the most valuable property of ours?

[3] HEX - hexadecimal order

[4] Self-Organizing Maps - neuron network type

References

1. Janckulík D., Personální telemetrický systém - Osobní mobilní ordinace, bakalářská práce, VŠB - TUO 2007 (09.05.2007)
2. Vašíček P., Osobní biotelemetrický systém, diplomová práce, VB - TUO 2007 (09.05.2007)
3. Corbelt, BlueEKG datasheet `http://www.corscience.de` (30.01.2006)
4. `http://www.taylorgifts.com` (14.04.2007)
5. Předpovídání pomocí neuronových sítí: `http://neuron.felk.cvut.cz/courseware/data/chapter/36nan060/s23.html` - (16.4.2007)
6. Samoučící se neuronová síť - SOM, Kohonenovy mapy `http://automatizace.hw.cz/rservice.php?akce=tisk&cisloclanku=2006051401` (13.3.2007)
7. Honkela J., Kaski S., Kohonen T., Lagus K., Paatero V., Saarela A., and Salojarvi J., Self Organization of a Massive Document Collection. Neural Networks, IEEE Transactions on, Volume 11

Možnosti modelovania softvéru na úrovni návrhových vzorov*

Ľubomír Majtás and Pavol Návrat

Ústav informatiky a softvérového inžinierstva, Fakulta informatiky a informačných technológií
Slovenká technická univerzita v Bratislave, Ilkovičova 3, 842 16 Bratislava, Slovensko
majtas@fiit.stuba.sk

Abstrakt *Návrhové vzory sa za posledné desaťročie stali veľmi užitočným nástrojom na skvalitnenie procesu vývoja softvéru. Modelovacie techniky, ktoré sa v súčasnosti používajú pri návrhu, však tento trend stále nereflektujú. Obsahujú len malú podporu na vyjadrenie, že určité prvky návrhu predstavujú inštanciu vzoru. Tento príspevok poukazuje na jedno z možných riešení tohto stavu. Nami uvažovaný prístup tkvie v možnosti modelovať softvér na úrovni návrhových vzorov, pričom sa snaží zachovať prepojenie medzi prvkami modelov na úrovni vzorov a úrovni bežného objektovo orientovaného návrhu. Opiera sa pri tom o technológie definované v rámci štandardu Modelom riadená architektúra (MDA), ktoré vychádzajú z použitia modelov na rôznych úrovniach abstrakcie a transformácií medzi týmito modelmi.*

1 Úvod

Asi najznámejšie uplatnenie pojmu vzor v softvérovom inžinierstve priniesla práca GoF [3], v rámci ktorej autori identifikovali a podľa definovanej šablóny podrobne opísali 23 návrhových vzorov. Súčasťou opisu každého vzoru je slovný opis jeho hlavnej myšlienky, príklad jeho vhodného použitia (vrátane ukážky zdrojového kódu), opis riešenia, ktoré vzor poskytuje a diskusia o jeho dôsledkoch či alternatívach. Hlavnou časťou opisu riešenia je zobrazenie jeho modelu pomocou štandardných OMT / UML diagramov. Tie slúžia vhodne na zachytenie príkladu použitia vzoru, no nedokážu plne zachytiť jeho celú štruktúru ani myšlienku. Vzory opisujú ako skupinu spolupracujúcich tried, objektov, metód či iných OO stavebných prvkov, zatiaľ čo vzory môžeme skôr považovať za skupinu spolupracujúcich rolí [6]. Tie môžu, ale nemusia, byť hrané práve OO stavebnými prvkami z ukážkových diagramov. Práca GoF nedefinuje také diagramy, ktoré by explicitne zachytili podstatu vzoru. Existujú však práce iných autorov, ktoré sa o to snažia v rôznej podobe (napr. [6], [1], [5], [7]).

Pri používaní návrhových vzorov v praxi sa spravidla stretávame iba so štandardnými UML diagramami. Obsahujú konkrétne inštancie vzorov len vo forme spolupracujúcich OO prvkov, pričom informáciu, že ide o použitie nejakého vzoru, naznačujú len pomenovania týchto prvkov. Presná informácia o tom, akú rolu hrá daný prvok diagramu, chýba. Potom sa môže ľahko stať, že návrhár spätne nerozpozná, že ide o použitie určitého vzoru, čo v konečnom dôsledku môže znamenať stratu významu jeho nasadenia, či iné komplikácie. Tým, že v diagramoch štandardne neuchovávame explicitné informácie o vzoroch, často sa v návrhu strácajú, čím sa sami pripravujeme o mnohé výhody, ktoré plynú z ich použitia. Následne vývojári medzi sebou opäť komunikujú na úrovni OO návrhu, zatiaľ čo by mohli pracovať s vyššou úrovňou abstrakcie, ktorú vzory prinášajú. Vzory navyše poskytujú elegantné možnosti rozšírení či iných zmien v návrhu, ktoré sa týmto strácajú.

2 Možnosti práce na vyššej úrovni abstrakcie

Rozšírenia diagramov zachytávajúce informácie o prítomných inštanciách vzorov [2] predstavujú prínos pri modelovaní softvérového návrhu s použitím vzorov. Stále však kladú dôraz na rozmiestnenie a spoluprácu tried, pričom informácie týkajúce sa vzorov sú až druhoradé. Znamená to, že návrhár pracujúci s takýmito modelmi robí stále na nižšej úrovni abstrakcie než umožňujú vzory. Samotné značky týkajúce sa vzorov sú do diagramov pridané až po vytvorení modelu tried. To znamená, že návrhár musí najskôr sám pochopiť vzor, manuálne vytvoriť jeho inštanciu v návrhu a tú označkovať tak, aby bolo možné vzor spätne rozpoznať.

Oveľa zaujímavejší prístup by bol, keby bolo možné pracovať na vyššej úrovni abstrakcie - na úrovni vzorov. Návrhár by definoval, aký vzor chce použiť a ako ho chce pripojiť do kontextu riešeného systému. Nástroj by následne zabezpečil vytvorenie inštancie vzoru, ktorú by korektne napojil k ostatným častiam návrhu podľa špecifikácie používateľa. Na to, aby bolo možné nad niečím podobným uvažovať, je potrebné zadefinovať dva základné nástroje:

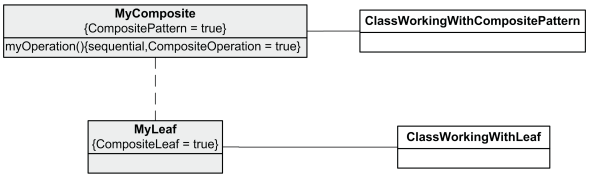
- Meta-model, podľa ktorého by bolo možné modelovať na vyššej úrovni abstarakcie vzorov.
- Metódy, ktoré by transformovali modely na vyššej úrovni abstrakcie do modelov na klasickej OO úrovni.

* Táto práca bola podporená Vedeckou grantovou agentúrou Slovenskej republiky grant č. 1/0162/03.

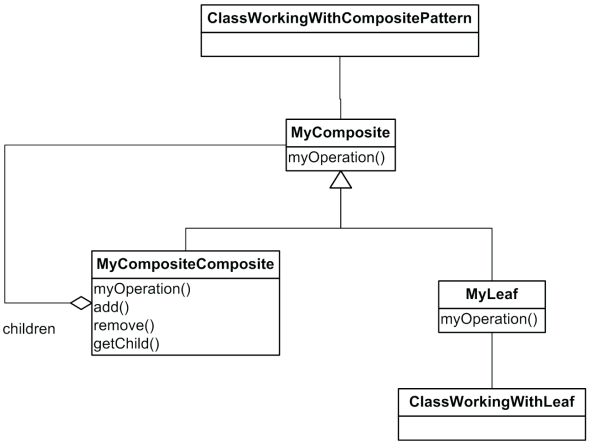
Pri uvažovaní o takomto spôsobe riešenia je vhodné využiť výsledky súvisiace s iniciatívou Modelom riadená architektúra (MDA). MDA prístup umožňuje modelovať na vyššej úrovni abstrakcie nezávisle od platformy, na ktorej by mal byť výsledný systém nasadený. Následne by malo byť možné po dodaní špecifikácií jednotlivých platforiem transformovať takto vytvorené modely do modelov, ktoré sú postavené pre konkrétne platformy. Z nich by sa následne malo dať vytvoriť konkrétne, v rámci platformy použiteľné riešenie (napr. zdrojové kódy, konfiguračné súbory a pod.).

Pre naše potreby definovania návrhu na vyššej úrovni abstrakcie vzorov sa javí MDA prístup ako vhodné riešenie. Za platformovo špecifické modely (PSM) môžeme považovať klasické UML diagramy obsahujúce konkrétne aplikované inštancie vzorov, z ktorých je neskôr možné vygenerovať zdrojové kódy. Za platformu možno považovať definície štruktúr návrhových vzorov z pohľadu OO návrhu. Zostáva definovať formu zápisov platformovo nezávislých modelov (PIM) a nástroje, ktoré budú použité na transformáciu z PIM do PSM.

Pred výberom jazyka PIM je nutné špecifikovať požiadavky, ktoré budú od neho vyžadované. Najdôležitejšou je možnosť modelovania na úrovni vzorov, pričom je nutné mať možnosť pripojiť prvky modelu na úrovni vzorov do kontextu ostatného OO návrhu. Tu sa javí ako najvhodnejšia možnosť používať UML modely, pričom samotné prvky na úrovni vzorov budú odlíšené od ostatných pomocou vlastného UML Profilu.



Obrázok 1. PIM so vzorom Composite.



Obrázok 2. PSM so vzorom Composite.

3 Modelovanie vzorov

Nasledujúca kapitola ukáže príklady modelovania na vyššej úrovni abstrakcie pre vzory Composite a Decorator vrátane ukážky ich možnej spolupráce.

3.1 Composite

Cieľom vzoru Composite podľa práce GoF je umožniť zhromažďovanie objektov do stromových štruktúr, pričom by malo byť možné pracovať s jedným objektom rovnako ako so skupinou.

Keď sa pozrieme na vzor z pohľadu, ako by sa mal modelovať, môžeme si všimnúť, že jediné, čo potrebujeme definovať, je trieda (prípadne skupina tried), ktorá má byť uložiteľná v stromovej hierarchii. Ostatné informácie ako napríklad konkrétna forma vytvárania stromovej štruktúry nie je pre nás pri modelovaní na vyššej úrovni podstatná. Takéto informácie by mali byť obsiahnuté v konfigurácii vytvárania inštancie vzoru a nie neoddeliteľnou súčasťou modelu.

Z tohto dôvodu sme sa rozhodli modelovať vzor Composite pomocou dvoch prvkov modelu: prvku reprezentujúceho samotný vzor a triedy, ktorá má byť pomocou vzoru ukladaná. Na obrázku č. 1 sa nachádza ukážka modelovania vzoru na vyššej úrovni. Pomocou UML Profilu (a pre prehľadnosť taktiež šedým podfarbením) sú odlíšené prvky vzorov od ostaných OO prvkov. Prerušovaná čiara reprezentuje vzťah na úrovni vzoru. Prvok *MyComposite* predstavuje inštanciu vzoru ako celok. Jeho súčasťou je aj metóda *myOperation* predstavujúca metódu, ktorú dokáže vykonávať celá hierarchia a rovnako každý jej člen. Trieda *MyLeaf* predstavuje triedu, ktorá môže byť pomocou vzoru uložená. Diagram zachytáva okrem prvkov na úrovni vzoru aj triedy na úrovni bežného OO návrhu, pričom tie dokážu navzájom spolupracovať: diagram obsahuje jednu triedu pracujúcu s celou hierarchiou poskytovanou vzorom a jednu triedu, ktorá dokáže pracovať len s triedou predstavujúcou list v hierarchii vzoru.

Taký model môže byť transformovaný do modelu klasického OO návrhu. Výsledok transformácie pre predchádzajúci model sa nachádza na obrázku č. 2. Ten obsahuje korektne vytvorenú inštanciu vzoru Composite spolu s triedami, ktoré so vzorom spolupracujú tak, ako to definuje model na obrázku č. 1.

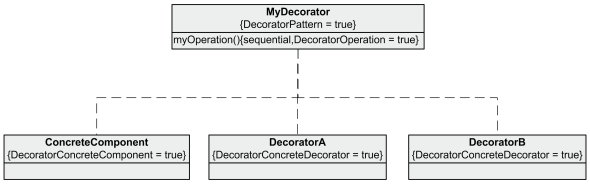
3.2 Decorator

Vzor Decorator slúži na dynamické pridávanie zodpovednosti triedam. Predstavuje flexibilnú alternatívu ku klasickému rozširovaniu systému pomocou pridávania potomkov do hierarchií dedenia. Jeho myšlienka spočíva v definovaní tried, ktorých objekty majú byť dekorovateľné a dekorátorov, ktoré majú rozširovať funkcionalitu tried (dekorovať triedy). Princíp dekorovateľnosti spočíva v postupnom volaní metód rozširujúcich dekorátorov v rámci volania pôvodnej metódy.

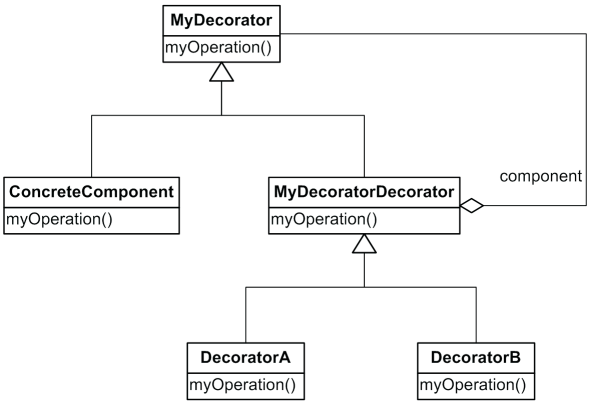
Keď sa opäť pozrieme na vzor Decorator z pohľadu modelovateľnosti, môžeme si všimnúť, že prioritou je v rámci modelu definovať dekorovateľné triedy a ich dekorátory. Forma, ako tieto triedy konkrétne spolupracujú za účelom dosiahnutia svojho cieľa, pre nás nie je v danom momente podstatná.

Modelovať vzor sme sa rozhodli podobným spôsobom ako v prípade vzoru Composite: definovaním prvku predstavujúcim inštanciu vzoru a triedami predstavujúcimi dekorátory a dekorovateľné triedy. Prvok predstavujúci samotný vzor navyše obsahuje definíciu metódy, ktorej funkcionalita môže byť v rámci inštancie vzoru dekorovateľná. Obrázok č. 3 zachytáva ukážku vyššieho modelu vzoru Decorator pracujúceho s jednou dekorovateľnou triedou (*ConcreteComponent*) a dvomi dekorátormi (*DecoratorA* a *DecoratorB*).

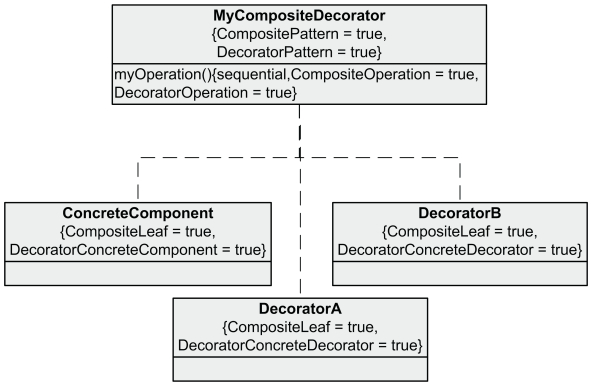
Obrázok č. 3 zachytáva výsledok transformácie modelu z obrázku č. 4 do bežného modelu na úrovni OO návrhu.



Obrázok 3. PIM so vzorom Decorator.



Obrázok 4. PSM so vzorom Decorator.



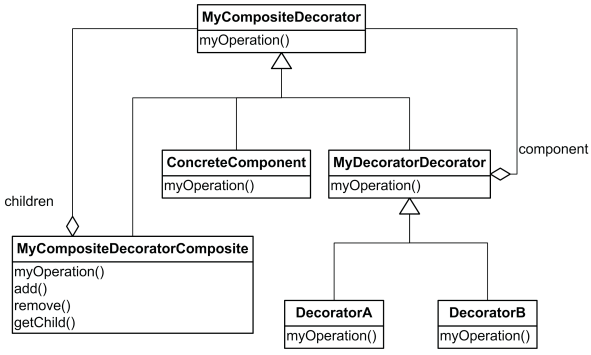
Obrázok 5. PIM s kompozíciou vzorov Decorator a Composite.

3.3 Kompozícia viacerých vzorov

Podobne ako sme pristupovali k modelovaniu jednoduchých inštancií vzorov, môžeme pristúpiť aj k ich kompozícii. Predviesť možnosti modelovania na vyššej úrovni môžeme na spolupráci už opísaných vzorov Composite a Decorator, ktorých spojenie umožňuje vytvárať hierarchické štruktúry dekorovateľných objektov.

Kompozíciu vzorov modelujeme veľmi podobne ako sme v predchádzajúcich príkladoch modelovali samostatné inštancie vzorov. Opäť definujeme jeden prvok modelu, v tomto prípade predstavujúci kompozíciu viacerých vzorov. K tomuto prvku pripojíme triedy definujúce role jednotlivých vzorov. Príklad takto definovanej kompozície sa nachádza na obrázku č. 5.

Model môžeme transformovať podobne ako v predošlých prípadoch do klasického OO návrhu. Jedným zo spôsobov, ako to môžeme urobiť, je transformovať ho po častiach - najskôr vytvoriť inštanciu jedného vzoru a následne druhého. Obrázok č. 6 obsahuje výslednú kompozíciu inštancií vzorov.



Obrázok 6. PSM s kompozíciou vzorov Decorator a Composite.

3.4 Transformácie medzi modelmi

Úlohou transformácie nemá byť len vytvoriť "nejakú" inštanciu vzoru. Každý vzor je možné navrhnúť a implementovať viacerými korektnými spôsobmi, úlohou transformácie je aj identifikovať v danej situácii najvhodnejší variant vzoru a v takej podobe pripraviť jeho inštanciu. Ako príklad môžeme definovať rôzne varianty inštancií opísaných vzorov Composite a Decorator.

Composite
Najčastejšie opisované varianty vzoru na úrovni návrhu sa líšia v umiestnení metód pre pridanie alebo odstránenie prvkov hierarchie. Tieto metódy môžu byť umiestnené buď v rámci najvyššej triedy celej hierarchie (hráč roly Component) alebo len v rámci triedy slúžiacej na zoskupenie ostatných tried (hráč roly Composite) [4]. Ak sa metódy umiestnia do najvyššej triedy hierarchie, zjednotí sa síce celé jej rozhranie, ale na druhú stranu sa tieto metódy stanú bezvýznamné pre triedy, ktoré neobsahujú ďalšie podtriedy (listy). Určiť, ktorý zo spôsobov je vhodnejší, nie je jednoduché, pretože závisí od toho, akým spôsobom si vývojár želá pracovať so vzorom. Preto rozhodnutie o variante v prípade samostatnej inštancie vzoru Composite by malo byť definované používateľom, napríklad v konfigurácii transformácie.

Decorator
V prípade vzoru Decorator možno identifikovať alternatívy vo forme použitia či nepoužitia rozhrania zastrešujúceho konkrétne dekorátory. V tomto prípade sa môže javiť voľba alternatívy jednoduchšie: ak je definovaných viac dekorátorov, použiť toto rozhranie, v prípade jedného dekorátora rozhranie stráca význam, a preto ho môžeme vynechať.

Kompozícia vzorov Composite a Decorator
Pri vytváraní kompozície viacerých vzorov je nutné vybrať také alternatívy jednotlivých spájaných vzorov, aby bolo možné zabezpečiť ich vzájomnú spoluprácu čo možno najjednoduchšie. V rámci príkladu definovanom v kapitole 3.3 došlo k použitiu vzoru Decorator so samostatným rozhraním pre dekorátory, pretože inštancia obsahuje viac dekorátorov. Súčasne bol použitý variant vzoru Composite s metódami na pridávanie a odoberanie objektov len v rámci kompozitnej triedy, nakoľko pridanie metód do celej hierarchie by znamenalo pridanie týchto metód aj do tried definovaných vzorom Decorator, čo by zbytočne komplikovalo návrh.

4 Zhodnotenie

Cieľom nášho príspevku je poukázať na možnosti modelovania návrhu softvéru na rôznych úrovniach abstrakcie s použitím návrhových vzorov. Vychádza pritom z MDA prístupu, ktorý je postavený na myšlienke používania modelov na rôznom stupni nezávislosti od použitej platformy. Prínosom takého spôsobu modelovania je možnosť pracovať, uvažovať a komunikovať na úrovni vzorov podľa ideí autorov, ktorí vzory definovali. S využitím MDA prístupu môžeme nielen modelovať na úrovní vzorov, ale naše modely transformovať do bežných OO modelov a následne do zdrojových kódov.

Príspevok poukazuje na možnosti modelovania na vyšších úrovniach abstrakcie, no nepokúša sa riešiť túto komplexnú problematiku ako celok. Za jeho primárny cieľ možno považovať prezentácie príkladov, pomocou ktorých by mali byť podobné úlohy riešiteľné. Pri vytváraní príkladov sa vynárali ďalšie otázky, na ktoré bude potrebné v budúcnosti zodpovedať. Možno ich zhromaždiť do už spomínaných dvoch skupín: otázky ohľadom špecifikácie meta-modelu pre prácu s vzormi a otázky ohľadom transformácií medzi modelmi. Ďalej sa pokúsime nájsť odpovede na tieto otázky, čoho výsledkom by mohlo byť zadefinovanie komplexnej metódy na modelovanie so vzormi na vyššej úrovni abstrakcie.

Referencie

1. Dietrich J., Elgar Ch., A Formal Description of Design Patterns Using OWL. Australian Software Engineering Conference, 2005
2. Dong J., UML Extensions for Design Pattern Compositions. Journal of Object Technology, 1., 5, 2002
3. Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley 1995
4. Jakubík J., Izolácia všeobecných častí vzoru Composite. Zborník konferencie Objekty 2005, Ostrava 2005
5. Majtás, Ľ., Určenie vhodného spôsobu ukladania modelov návrhových vzorov. Zborník konferencie Objekty 2006, Praha 2006
6. Mak J.K.H., Choy C.S.T., Lun D.P.K., Precise Modeling of Design Patterns in UML. 26th International Conference on Software Engineering, 2004
7. Smolárová M., Návrat P., and Bieliková M., A Technique for Modelling Design Patterns. Knowledge-Based Software Engineering – JCKBSE'98, IOS Press 1998

Zvýšení účinnosti komprese HTML souborů vhodným předzpracováním*

Václav Matouš and Michal Žemlička

MFF UK, Malostranské nám. 25, 11800 Praha 1, Česká republika
venca.matous@centrum.cz, zemlicka@ksi.mff.cuni.cz

Abstrakt *Vyhledávače potřebují uchovávat obrovské množství dokumentů zejména ve formátu html. Je proto užitečné co nejúčinnější kompresí redukovat jejich velikost – a to při zachování rychlého přístupu k souborům (dekomprese nesmí trvat moc dlouho). Dalším omezením je, že mnoho stránek žádné verzi normy html neodpovídá. Často tedy není možné těžit ze znalosti html formátu. Rozhodli jsme se proto zvýšit účinnost existujících aplikací gzip a bzip2 předzpracováním komprimovaných dokumentů. Předzpracování je založeno na nahrazování nejčastějších značek elementů a jejich atributů kratšími značkami. To zrychluje inicializaci slovníku aplikace gzip a zjednodušuje vstup pro bzip2. Téměř ve všech případech je velikost zkomprimovaných souborů menší v případě předzpracování než bez něj. Výjimku tvoří velmi malé soubory.*

1 Úvod

HTML dokument obsahuje semistrukturovaná data, tedy vedle běžně zobrazovaných dat (typicky textů) obsahuje i data, která určují jeho strukturu. HTML dokumenty jsou tedy jak texty, tak strukturovanými dokumenty. K jejich kompresi tak můžeme využívat běžné metody komprese textu – včetně velmi účinných slovních či slabikových. Vzhledem k přítomnosti HTML značek není dokument textem v přirozeném jazyce, a proto tato komprese nedosáhne své plné účinnosti.

Můžeme se také pokusit komprimovat HTML dokumenty jako semistrukturovaný (XML) text. Ke kompresi XML dokumentů bylo vyvinuto několik kompresních metod využívajících vlastností XML. Příkladem takového algoritmu může být například XMill [1]. Důležitým předpokladem těchto specializovaných algoritmů je dobře formovaný (z pohledu značek dobře uzávorkovaný) komprimovaný dokument. Bohužel, specifikace HTML dokumentů jsou v tomto směru dosti tolerantní, takže ani u dokumentu splňujícího specifikaci není zaručena jeho dobrá formovanost, např. `<b><i>.....</b></i>` vyhovuje specifikaci, ale není dobře formovaná část dokumentu. Dobrou formovanost vyžaduje až specifikace pro XHTML, takže pro tyto dokumenty lze výše zmíněné kompresní algoritmy použít. Podíváme-li se ale na zastoupení jednotlivých verzí HTML na Internetu, zjistíme, že nejčastěji se vyskytujícím typem dokumentu jsou dokumenty psané podle specifikace verze 3 nebo 4 (dokonce lze stále najít i dokumenty psané podle specifikace verze 2). Specializované algoritmy využívající struktury dokumentu nelze tedy pro kompresi dokumentů stažených z Internetu obecně použít.

Výsledkem obou pohledů na HTML dokumenty je jejich komprese pomocí obecných metod, jakými jsou např. gzip [2] nebo bzip2 [3], které nevyžadují žádnou strukturu, ani slova nebo slabiky.

2 Motivace pro použití předzpracování

Protože HTML dokumenty používají pouze omezenou množinu HTML elementů a jejich atributů, které jsou navíc více či méně frekventované a jejich použití je dáno příslušnou HTML specifikací, můžeme si dovolit předpokládat výskyt určitých elementů a atributů v komprimovaných dokumentech. Předzpracování v našem podání spočívá v náhradě řetězců reprezentujících tyto často se vyskytující elementy nebo atributy kratšími symboly – nepoužitými znaky.

Tato myšlenka předzpracování komprimovaného textu není obecně nová. Řada podobných metod pro anglické texty obsahující pouze dolních 128 znaků je popisována v [4], kde se například výskyty slov z obvykle pevného slovníku před kompresí nahradí znaky z horních 128 znaků, nebo místo slov se používají Q-gramy (Q-tice nejčastěji se vyskytujících po sobě jdoucích znaků), jejichž výskyty jsou rovněž před kompresí nahrazeny horními 128 znaky. Pro kompresi takto předzpracovaných dokumentů jsou následně použity běžné kompresní algoritmy.

3 Zkoumané metody předzpracování

Na základě důvodů vyslovených v předchozím odstavci jsme se rozhodli vylepšit účinnost komprese HTML dokumentů použitím předzpracování před samotnou kompresí. Pro předzpracování jsme použili dvě podobné metody, které se liší především způsobem kódování slovníku a které při testech používaly stejný (statický) slovník, aby se jejich dosažené výsledky daly objektivně porovnat.

* Tento výzkum byl částečně podporován programem "Informační společnost" jako projekt 1ET100300517.

3.1 Volba slovníku

Obě metody používají statický slovník, proto by měl slovník obsahovat takové řetězce (názvy elementů a atributů), které se často vyskytují ve velké části HTML dokumentů. Při výběru řetězců, kterými jsme naplnili slovník, jsme zohlednili statistiky, které uvádějí nejčastěji používané elementy a atributy [5] v dokumentech na Internetu a dále jsme zohlednili nejčastěji se vyskytující objekty v HTML dokumentech, jakými jsou seznamy, odstavce, tabulky apod.

Naše heuristika při tvorbě slovníku:

1. Tentýž element se může v rámci jednoho dokumentu vyskytnou jednou s atributy a podruhé bez atributů. Aby se zvýšil počet řetězců, které budou nahrazeny stejným znakem, obsahuje slovník počáteční značky vybraných elementů ve tvaru `<` a název elementu, např. `<div`. Tento řetězec bude tedy nahrazen jak při výskytu `<div>`, tak při výskytu `<div class="">`.
2. Koncová značka žádného elementu nesmí dle specifikace obsahovat žádný atribut. Proto pro každý vybraný element, který má koncovou značku, obsahuje slovník tuto značku celou, např. `</div>`.
3. Pro vybrané elementy, u kterých se s nejvyšší pravděpodobností výskyt nějakého atributu očekává, nebo které mají alespoň jeden atribut povinný, obsahuje slovník začátek počáteční značky (viz bod 1) doplněný zprava o jednu mezeru, tedy `<img␣`, `<link␣`, `<meta␣`.
4. Hodnoty atributů mohou být v HTML dokumentech uzavřeny v jednoduchých uvozovkách, např. `border='1'`, uvozovkách, např. `border="1"` nebo nemusejí být uzavřeny vůbec, např. `border=1`. Aby byl slovník použitelný pro všechny tyto případy, jsou názvy atributů uváděny ve tvaru: `border=`.
5. Názvy elementů a atributů se mohou až do verze 4 psát jak malými, tak velkými písmeny nebo jejich libovolnou kombinací. Podle zkoumaných dokumentů jejich autoři nejčastěji používají buď názvy psané samými malými, nebo naopak samými velkými písmeny. Vzhledem k tomu, že při vyhledávání řetězců ze slovníku pro nahrazení znakem hraje velikost písmen roli, obsahuje proto slovník pro každý vybraný řetězec obě možnosti zapsání, například `<div` a `</div>` i `<DIV` a `</DIV>` nebo `border=` i `BORDER=`. Vyskytne-li se přípustný zápis elementu `<dIv>`, není tento výskyt nahrazen příslušným znakem. Podobný způsob zápisu elementů a atributů se naštěstí poměrně vzácný.

Námi používaný slovník obsahoval 56 řetězců zastupujících značky elementů nebo názvy atributů, pro každý z nich jak variantu zapsání samými malými, tak samými velkými písmeny. Celkem tedy 112 řetězců.

Pro oddělené kódování tagů velkými a malými písmeny jsme se rozhodli proto, že nechceme poškodit ani nevhodně zapsané dokumenty.

3.2 Popis metody 1

Kódování slovníku Ke kódování slov z používaného slovníku, která se vyskytla v komprimovaném dokumentu, používá tato metoda ty znaky z celkových 256, které se v tomto dokumentu nevyskytují. Slova ve slovníku mají pevné pořadí. Podle tohoto pořadí první slovo ze slovníku, které se v dokumentu vyskytlo, je v předzpracovaném dokumentu nahrazeno prvním znakem, který není v dokumentu použit. Kódování v pořadí dalších slov je prováděno stejným způsobem.

Informace o mapování vyskytnuvších se slov na nevyužité znaky musí být uchována v předzpracovaném dokumentu, aby bylo možné provést zpětnou rekonstrukci původního dokumentu. Proto je do předzpracovaného dokumentu vkládána na začátek hlavička tvořená bitovou mapou. Prvních 256 bitů určuje, který znak byl v původním dokumentu použit – na jeho pozici je v bitové mapě 1, u nepoužitých znaků 0. Druhou část mapy tvoří bity, kde každý odpovídá právě jednomu slovu ze slovníku. Vyskytlo-li se příslušné slovo v původním dokumentu, je na jeho místě 1.

Ze způsobu kódování plyne jediné omezení na použití této metody – počet použitých znaků v původním dokumentu a slov ze slovníku, která se v dokumentu vyskytla, musí být nejvýše 256. Musíme ale jedním dechem dodat, že u žádného ze skoro 40000 souborů, na kterých byla metoda testována, nenastala situace, že by tento počet byl vyšší než 256. Tato metoda může být použita nejen pro dokumenty v anglickém jazyce, ale obecně pro dokument v téměř libovolném jazyce – zatím jsme netestovali stránky psané v jazycích s velmi velkou abecedou.

Fáze předzpracování Samotné předzpracování se skládá celkem ze tří fází:

1. První fáze by se dala označit za počítání statistik. Komprimovaný soubor se čte a zjišťuje se, které znaky se v souboru vyskytují a také která slova ze slovníku. Vzhledem k tomu, že není vyloučena možnost, aby jedno slovo ze slovníku bylo předponou slova jiného, používá se při hledání řetězců maximální shoda. Jinými slovy, pokud bude ve slovníku slovo `<h` i slovo `<html`, pak po přečtení znaků `<` a `h` není zahlášeno slovo `<h`, ale čtou se další znaky, aby se zjistilo, zda nenásledují znaky t, m, l.
2. Druhou fází je vytvoření mapování nalezených slov ze slovníku na nepoužité znaky. Způsob viz Kódování slovníku. Vytvořená mapa se zapíše na začátek předzpracovaného dokumentu.

Velikost souborů v KB	≥ 10	10–20	20–30	30–40	40–50	50–60	60–70	70–80	80–90	90–100	> 100
celkem souborů	16929	13565	4340	1878	871	514	180	116	70	68	107
gz lepší	2880	51	1	0	1	0	0	0	0	0	0
mgz1 lepší	14001	12461	3187	962	293	50	19	3	1	1	0
bz lepší	183	663	451	236	156	43	24	13	17	9	22
mbz1 lepší	7	424	727	694	427	425	138	100	53	60	85

Tabulka 1. Počty nejlépe zkomprimovaných souborů – porovnání metody 1 upravující vstup pro gzip nebo bzip2 a použití samotných aplikací gzip a bzip2.

3. Za zapsanou hlavičku z druhé fáze se do předzpracovaného souboru zapíše jeho obsah. V podstatě jde o původní soubor, ve kterém jsou výskyty slov ze slovníku nahrazeny příslušnými znaky. Vyhledávání slov pro nahrazení se samozřejmě provádí stejným algoritmem jako ve fázi 1.

Jak je patrné z popisu algoritmu předzpracování, jsou zapotřebí navíc dva průchody komprimovaným souborem (v první a ve třetí fázi).

Zpětná rekonstrukce Na rozdíl od fáze předzpracování odpadá počítání statistik souboru. Zpětná rekonsturkce má tedy jen dvě fáze:

1. Přečte se hlavička z předzpracovaného dokumentu a na jejím základě se provede rekonstrukce mapování slov ze slovníku na nepoužité znaky. První použité slovo ze slovníku bylo v předzpracovaném dokumentu nahrazeno prvním nepoužitým znakem, druhé slovo druhým nepoužitým znakem atd.
2. Čte se tělo předzpracovaného dokumentu. Pokud přečtený znak nezastupuje žádné slovo ze slovníku, tento znak se pouze zreprodukuje na výstup. V opačném případě se místo tohoto znaku na výstup pošle příslušný nahrazený řetězec.

Zpětná rekonstrukce si vystačí pouze s jedním průchodem předzpracovaným souborem. Zrekonstruovaný dokument je totožný s dokumentem, který byl vstupem původního předzpracování, neboť jde o bezeztrátovou úpravu.

3.3 Popis metody 2

Kódování slovníku Tato metoda se snaží využít ke kódování celého dokumentu pouze dolních 128 znaků. Lze ji použít pouze na dokumenty, ve kterých se nevyskytují znaky s kódem 128 a vyšším. Slova ze slovníku, které se v dokumentu vyskytla jsou kódována v dokumentu nepoužitými symboly s kódy nižšími než 128. Slova jsou stejně jako u předchozí metody ve slovníku uspořádána v pevném pořadí. První (dle tohoto pořadí) použité slovo je kódováno prvním nepoužitým znakem z dolních 128, druhé slovo druhým nepoužitým znakem atd.

Informace o kódování se do hlavičky předzpracovaného dokumentu ukládá také ve formě bitové mapy. Tentokrát si ale vystačíme se 128 bity, kde 1 znamená, že příslušný znak se v dokumentu vyskytl a 0 značí, že tento znak mohl být využit pro kódování slovníku. Druhou část mapy tvoří bity, kde každý odpovídá právě jednomu slovu ze slovníku. Význam bitů je shodný jako u metody 1.

Z popsaného způsobu kódování plyne jedno nepříjemné omezení – počet slov slovníku, která se v dokumentu vyskytla, nesmí být větší než počet nepoužitých dolních 128 znaků. Jinak bychom nebyli schopni zakódovat všechna slova ze slovníku. Toto omezení se nakonec ukázalo být velice silné – viz Výsledky testů.

Fáze předzpracování a zpětná rekonstrukce Fáze předzpracování u metody 2 jsou prakticky totožné s fázemi u metody 1. Stejně tak zpětná rekonstrukce původního dokumentu.

4 Výsledky testů

V této části budeme používat následující označení, která určují, která metoda předzpracování se kterým kompresním algoritmem byla použita:
mgz*i* – přezpracování metodou *i*, komprese gzip;
mbz*i* – přezpracování metodou *i*, komprese bzip2.

Jako implementace algoritmu gzip a bzip2 byly při provádění testů použity knihovny zlib [6] a libbzip2 [7]. Při testování jsme použili téměř 40000 webových stránek stažených z různých akademických webů.

Dosažené výsledky jsou shrnuty v tabulkách 1, 2 a 3, kde jsou po řadě porovnání původních kompresních metod a doplněných o předzpracování metodou 1, v tabulce 2 s předzpracováním metodou 2 (uvažovány byly pouze soubory, kdy předzpracování metodou 2 bylo možné) a v tabulce 3 jsou porovnány původní algoritmy s oběma jejich modifikovanými variantami.

Úspory dosažené předzpracováním (došlo-li k nim) se většinou pohybují v desítkách až stovkách byte v případě algoritmu gzip, a desítek byte v případě algoritmu bzip. Není to příliš výrazné zlepšení, ale i malým zlepšením se může stát, že velikost souboru klesne pod hranici násobku alokační jednotky.

Velikost souborů v KB	≥ 10	10–20	20–30	30–40	40–50	50–60	60–70	70–80	80–90	90–100	> 100
počet souborů	6075	2003	322	86	32	10	8	2	1	3	6
gz lepší	1378	7	0	0	1	0	0	0	0	0	0
mgz2 lepší	4655	1674	148	49	7	0	0	0	0	0	0
bz lepší	99	219	53	9	5	2	0	1	0	1	2
mbz2 lepší	4	108	122	28	19	8	8	1	1	2	4

Tabulka 2. Počty nejlépe zkomprimovaných souborů – porovnání metody 2 upravující vstup pro gzip nebo bzip2 a použití samotných aplikací gzip a bzip2.

Velikost souborů v KB	≥ 10	10–20	20–30	30–40	40–50	50–60	60–70	70–80	80–90	90–100	> 100
počet souborů	6075	2003	322	86	32	10	8	2	1	3	6
gz lepší	1377	7	0	0	1	0	0	0	0	0	0
bz lepší	98	211	51	9	5	2	0	1	0	1	2
mgz1 lepší	1664	756	82	28	5	0	0	0	0	0	0
mbz1 lepší	2	49	63	11	10	2	4	0	0	1	2
mgz2 lepší	4586	1649	147	48	7	0	0	0	0	0	0
mbz2 lepší	3	81	68	20	9	6	4	1	1	0	2

Tabulka 3. Počty nejlépe zkomprimovaných souborů – srovnání obou metod předzpracování; testovány pouze soubory, na než je možné aplikovat obě metody předzpracování.

5 Diskuse

Na testovaných kolekcích se ukázalo, že námi navrženým předzpracováním html dokumentů je možné zvýšit zisk komprese stávajících kompresních metod. Tohoto zlepšení však není dosahováno vždy. V našich testech bylo nejčastěji dosaženo nejlepších výsledků kompresním algoritmem gzip po předzpracování – ať již metodou 1, či metodou 2 (tam jen v případech, kdy to bylo možné – tedy na dokumentech, obsahujících pouze znaky s kódy 0 až 127).

Měření potvrdila, že nemůžeme říci, že by jedna z metod předzpracování jednoznačně překonávala tu druhou. Při dalších testech i praktických aplikacích bude nutné nadále uvažovat obě metody předzpracování.

Domníváme se, že data mohla být zatížena výběrem (jde o stránky získané crawlerem Egothor [8] z akademických domén), a tak bude třeba provést další měření i na stránkách z jiných domén, kde se možná setkáme s jinou strukturou dokumentů.

6 Závěr

Navrhli jsme a vyzkoušeli dvě metody předzpracování html dokumentů umožňující ve většině případů zvýšit kompresní poměr dosahovaný stávajícími aplikacemi gzip a bzip.

Námi navržené a testované metody nepřekonávají stávající metody vždy, ale dost často na to, aby bylo smysluplné je přidat do námi vytvářeného úložiště html stránek. Domníváme se, že uchování přídavné informace o použité metodě se zlepšeným kompresním poměrem ve většině případů více než napraví.

Nepříjemnou vlastností navrženého předzpracování je, že vyžaduje další průchod (podobně jako jiné statické metody). Dle našich měření je doba dekomprese jen minimálně prodloužena v porovnání s původními metodami. Prodloužení doby komprese je výraznější, ale stále ještě v přijatelných mezích.

Chceme-li se zaměřit na úsporu místa, potřebujeme-li rychlý přístup k uloženým datům a nevadí-li nám mírné zpomalení vytváření archivu, zdá se, že námi navržené a testované metody předzpracování HTML dokumentů jsou výhodné.

Reference

1. Liefke H. and Suciu D., Xmill: An Efficient Compressor for XML Data. In Chen W., Naughton J.F., Bernstein P.A., (eds), SIGMOD Conference, ACM, 2000, 153–164
2. Gailly J.L. and Adler M., (The gzip home page) `http://www.gzip.org/`
3. Seward J., (The bzip2 and libbzip2 official home page) `http://sources.redhat.com/bzip2/` as visited on 6th February 2005.
4. Skibiński P., Reversible Data Transforms that Improve Effectiveness of Universal Lossless Data Compression. PhD thesis, University of Wrocław, Wrocław, Poland, 2006
5. Google, Web authoring statistics, 2006 `http://code.google.com/webstats/index.html`
6. Gailly J.L. and Adler M., (zlib version 1.2.3) `http://www.zlib.net`
7. Seward J., (libbzip2 version 1.0.4) `http://www.bzip.org`
8. Galamboš L., et al., Egothor 2006, `http://www.egothor.org/`

Kompresia konkatenovaných webových stránok pomocou XBW *

Radovan Šesták, Jan Lánský, and Petr Uzel

Univerzita Karlova v Prahe, Matematicko-fyzikálna fakulta, Katedra softwarového inžinierstva

Abstrakt *XBW [5] je modulárny program na bezstratovú komprimáciu umožňujúci otestovať rôzne kombinácie algoritmov. Najlepšie výsledky sme dosiahli kombináciou XML parseru vytvárajúceho slovník slabík, alebo slov v kombinácii s Burrows-Wheelerovou transformáciou – odtiaľ názov XBW. Motiváciou pre vytvorenie parseru, ktorý dokáže spracovať nevalidné XML a HTML súbory, bol systém Egothor [6] na fulltextové vyhľadávanie. Na súboroch veľkosti okolo 20MB, tvorených stovkami webových stránok, sme dosiahli dva krát lepší kompresný pomer v porovnaní s bzip2 za cenu iba dvojnásobného času. Na menších súboroch má náš program veľmi dobré výsledky oproti konkurencii najmä pre jazyky z bohatým tvaroslovím ako je napríklad slovenčina alebo nemčina. Pre ľubovolné veľké textové súbory náš program poskytuje dobrý pomer medzi kompresiou a časom behu.*

Program XBW umožňuje kombinovať parser a kóder s ľubovolným implementovaným algoritmom na kompresiu. Okrem už spomínanej Burrows-Wheelerovej transformácie, ktorá spolu s MTF a RLE tvorí blokovú kompresiu, sme implementovali slovníkové metódy LZC a LZSS a štatistickú metódu PPM. Kóder umožnuje použiť Huffmanovo a aritmetické kódovanie.

1 Úvod

V tomto článku uvádzame výsledky komprimácie veľkých XML súborov. Motiváciou pre túto prácu je komprimácia dát pochádzajúcich z webu. Pre fulltextové vyhľadávanie je veľmi dôležitá rýchlosť spracovania dotazu, a preto komprimácia nie je vždy vhodná. Na druhej strane archivácia starých verzií webových stránok vyžaduje obrovské množstvo priestoru na diskoch. Navyše sa k týmto dátam až tak často nepristupuje, a preto sa ponúka kompresia ako možnosť riešenia problému s nedostatkom priestoru. XML formát je značne redundantný a preto je možné dosiahnuť veľmi dobrý kompresný pomer. Príbuzné webové stránky, v zmysle ich pôvodu, ohsahujú veľké úseky rovnakých dát. Vďaka týmto vlastnostiam dát, s ktorými pracujeme, sme boli schopní skomprimovať vstup desať násobne.

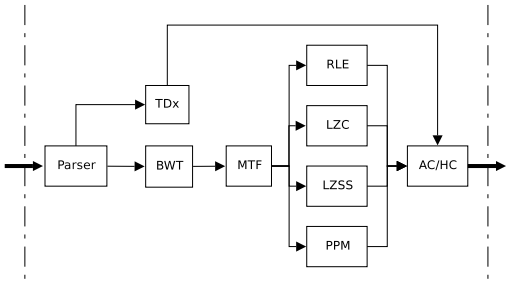
Ako testovacie súbory boli použité XML súbory pochádzajúce zo systému *Egothor* [6]. Tieto súbory majú veľkosť okolo 20MB. Vznikli zreťazením stoviek webových stránok a obsahujú značné množstvo textov. Veľké súbory môžu byť efektívnejšie skomprimované z niekoľkých dôvodov. Pri využití slovníka je lepší pomer veľkosti slovníka ku veľkosti súboru. A najmä entropia textových súborov sa znižuje, čo v praxi znamená, že sa dá efektívnejšie predikovať nasledujúci znak. Problematickým je fakt, že tieto súbory nemajú validnú XML štruktúru a častokrát nie sú dokonca ani dobre formované. Toto nás viedlo k vytvoreniu vlastného parseru, nakoľko nám známe XML parsery neboli schopné tieto súbory spracovať.

V nasledujúcej sekcii popíšeme jednotlivé časti programu XBW a ich vplyv na kompresiu. Potom budú následovať výsledky meraní a porovnanie s rozšírenými kompresnými programami.

2 Implementované metódy

V Obrázku 1 je zobrazené zapojenie jednotlivých častí programu. Všetky časti sú voliteľné. Z algoritmov RLE, LZC, LZSS a PPM je možné zvoliť maximálne jednu možnosť, pretože každý z týchto algoritmov využíva kóder. Implementovaný je v XBW Huffmanov a aritmetický kodér. Najlepšie výsledky pre veľké súbory sme dosiahli s kombináciou Parser + BWT + MTF + RLE + HC, ktorá je využitá ako východzie nastavenie programu.

Implementovaný parser využíva syntax XML súborov na skrátenie výstupu. Vynecháva ukončovacie znaky a dynamicky si vytvára slovník tagov a atribútov. Ostatné dáta delí buď na znaky, slabiky alebo slová a tie sú pridávané do trie. Táto voľba je jedným z parametrov parseru. Využitie abecedy slov je pomerne bežné pre textovú kompresiu, avšak jej použitie na XML súbory tak časté nie je. Nami použité delenie na slová, respektíve slabiky, a taktiež kódovanie



Obrázok 1. Architertúra XBW.

* Práca bola čiastočne podporovaná Grantovou agentúrou Univerzity Karlovej - v rámci projektu Slabiková komprese (číslo 1607 v sekcii A)

slovníku vychádza z práce Lánskeho [1]. Taktiež je možné použiť parser v textovom režime, kedy neprihliada na špeciálnu štruktúru XML súborov a iba rozdeľuje vstup na symboly zo zvolenej abecedy (znaky, slabiky, slová). Ďalším parametrom pre parser je voľba kódovania súboru; podporované sú desiatky kódovaní. Na podporu rôznych kódovaní je použitá knižnica *iconv* [2]. Slovník, ktorý je pri spracovávaní súboru udržiavaný v pamäti vo forme trie sa pri výstupe ukladá pomocou kódovania jej štruktúry. Pre každý uzol sa kóduje vzdialenosť od ľavého syna, počet synov a boolovská hodnota, ktorá určuje, či uzol reprezentuje reťazec.

Trieda metód na kompresiu, vychádzajúca z práce Burrowsa a Wheelera [3], je založená na reverzibilnej transformácii. Táto transformácia sa označuje ako Burrows-Wheelerova transformácia (BWT) [3]. Častokrát sa kombinácia tejto transformácie s následným efektívnym zápisom označuje ako bloková kompresia. Dôvodom je, že vstup sa rozdelí na bloky pevnej veľkosti a BWT sa volá pre jednotlivé bloky. Pri východzom nastavení v našej implementácii kombinujeme BWT s algoritmami MTF a RLE [5]. MTF prečísluje vstup a výsledkom je reťazec, ktorý obsahuje pomerne malé čísla a behy núl. Následovne je aplikovaný algoritmus RLE, ktorý zapíše znak a počet jeho opakovaní. Bitové zapísanie výsledných dát je uskutočnené pomocou kodéru. Dekódovanie je pri použití BWT výrazne rýchlejšie ako kódovanie.

XBW obsahuje aj slovníkové algoritmy LZC a LZSS, ktoré vychádzajú z algoritmov LZ78 a LZ77 [11]. Algoritmus LZC zapisuje na výstup index slova v slovníku, ktorý si vytvára za behu. (Tento slovník je nezávislý na slovníku, ktorý sme spomínali pri parsri.) Algorimus LZSS vyhľadáva najdlhšiu zhodu s reťazcom dostupným v histórii a zakóduje jej pozíciu a dĺžku. Tieto algoritmy sú hojne používané, lebo sú pomerne rýchle a vyžadujú veľmi málo pamäte. Pri použití týchto algoritmov program XBW dosahuje podobné výsledky ako Gzip, ktorý používa slovníkové metódy.

Najnovšia z implementovaných metód je štatistická metóda PPM [4], ktorá kóduje znaky na základe ich pravdepodobnosti po nejakom kontexte. Pravdepodobnosti jednotlivých znakov za kontextami sa počítajú dynamicky. Táto metóda, určená na kompresiu textov v prirodzených jazykoch, je pomerne pomalá a vyžaduje veľké množstvo pamäte.

Finálny bitový výstup je zabezpečovaný kodérom. Implementovaný je Huffmanov [7] a aritmetický kodér. Obe varianty sú implementovaná v statickej aj adaptívnej verzii. Aritmetický kodér využíva Moffatovu dátovú štruktúru a Huffmanov kodér je implementovaný v kanonickej verzii. Voľba Huffmanovho, respektíve aritmetického kodéru je daná parametrom pri kompilácii. Štandartne je zvolený aritmetický kodér, nakoľko má mierne lepší kompresný pomer ako kodér Huffmanov, a najmä je výrazne rýchlejší pri použití adaptívnych verzií.

3 BWT

Burrows-Wheelerovej transformácii sa budeme venovať bližšie, lebo jej použitie v optimalizovanej forme s upraveným vstupom nám umožnilo dosiahnuť výsledky, ktoré budeme prezentovať. BWT pri kódovaní vyžaduje lexikografické utriedenie všetkých suffixov. Výsledný reťazec má na i-tom mieste posledný symbol i-teho suffixu. Predpokladáme, že máme lineárne usporiadanú množinu symbolov Σ, ktorú nazývame abecedou. Pripomeňme, že symbolom v tomto zmysle môže byť aj slovo či slabika.

$X \equiv x_0x_1...x_{n-1}, \forall i \in \{0,..,n-1\}, x_i \in \Sigma$ je reťazec dĺžky n. i-ty *suffix* reťazca X je reťazec $S_i = x_ix_{i+1}..x_{n-1} = X[i..n-1]$. i-ty suffix je menší ako j-ty suffix, ak prvý znak, v ktorom sa nezhodujú, je menší, alebo i-ty suffix je kratší. $S_i < S_j \iff \exists k \in 0..n-1 : S_i[0..k-1] = S_j[0..k-1] \ \& \ (S_i[k] < S_j[k] \vee (i+k = n \ \& \ j+k < n))$. Poradie suffixov udržiavame v *suffixovom poli* SA, pre ktoré platí: $\forall i,j \in \{0..n-1\}$, $i < j \ \rightarrow S_{SA[i]} \leq S_{SA[j]}$.

Výsledkom BWT pre reťazec X je $\tilde{X}$. $\tilde{X} \equiv \tilde{x}_0..\tilde{x}_{n-1}$ kde $\tilde{x}_i = x_{|SA[i]-1|_n}$. Absolútne hodnoty značia operáciu modulo n, ktorú je nutné použiť v prípade, že $SA[i] = 0$.

Repetitívnosť súboru ovplyvňuje kompresný pomer aj čas behu kódovacej fáze BWT. *Dĺžku zhody* pre reťazce značíme $lcp(S_i, S_j) = max\{k; S_i[0..k-1] = S_j[0..k-1]\}$. *Priemerná dĺžka zhody* $AML \equiv \frac{1}{n-1}\sum_{i=0}^{n-2} lcp(S_{SA[i]}, S_{SA[i+1]})$ je údaj, ktorý používame v texte na meranie repetitívnosti súborov.

V programe sú implementované viaceré algoritmy na zotriedenie suffixov s rôznou asymptotickou zložitosťou. Najrýchlejším algoritmom pre nie príliš repetitívne súbory $(AML < 1000)$ je Kaova modifikácia Itohovho algoritmu [8], ktorý má časovú zložitosť $O(AML \cdot n \cdot log\, n)$. Na veľmi repetitívnych súboroch je najrýchlejši algoritmus od Kärkkaina a Sandersa [9] so zložitosťou $O(n)$. Pripomeňme, že voľba algoritmu na BWT neovplyvňuje kompresný pomer, iba časové a pamäťové požiadavky.

Pri blokovej kompresii sa súbor rozdelí na bloky pevnej veľkosti a BWT sa vykonáva nad jednotlivými blokmi. Tento postup je používaný na zníženie časovej a pamäťovej náročnosti, pretože BWT požaduje lineárnu veľkosť pamäti vzhľadom ku veľkosti vstupu. Časová zložitosť väčšiny algoritmov na BWT je asymptoticky superlineárna a BWT pri komprimácii je najpomalšou časťou celej blokovej kompresie. Nevýhodou malých blokov je zhoršenie kompresného pomeru.

Name	Size	AML
xml_cz	24604 KB	2200
xml_en	15016 KB	2052
xml_sl	21050 KB	2472

Tabuľka 1. Korpus.

bpB	Parser=off				Parser=text				Parser=xml			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,907	2,217	2,322	1,399	0,906	2,206	2,296	1,395	**0,894**	2,073	2,098	1,320
xml_en	0,886	2,044	2,321	1,292	0,887	2,044	2,321	1,292	**0,874**	1,915	2,115	1,239
xml_sl	0,710	1,982	2,010	1,205	0,710	1,979	2,003	1,204	**0,700**	1,850	1,797	1,129
TOTAL	0,834	2,093	2,213	1,305	0,833	2,087	2,200	1,303	**0,822**	1,957	1,998	1,234

Tabuľka 2. Vplyv parseru na kompresný pomer pre abecedu znakov.

MB/s	Kompresia								Dekompresia							
	Bez parsru				XML parser - znaky				Bez parsru				XML parser - znaky			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,368	3,587	1,498	0,106	0,457	2,668	1,418	0,091	4,260	4,724	5,257	0,117	2,577	3,156	3,415	0,096
xml_en	0,419	4,028	1,297	0,125	0,544	2,915	1,249	0,104	4,417	4,999	5,417	0,142	2,705	3,397	3,606	0,110
xml_sl	0,386	4,258	1,638	0,119	0,500	2,915	1,497	0,091	4,918	5,236	5,946	0,134	3,012	3,299	3,672	0,097
TOTAL	0,386	3,906	1,485	0,115	0,491	2,810	1,397	0,094	4,509	4,960	5,519	0,128	2,747	3,263	3,548	0,099

Tabuľka 3. Vplyv parseru na rýchlosť kompresie a dekompresie.

Hlavným dôvodom, prečo XBW dosahuje výrazne lepší kompresný pomer ako Bzip2, je použitie BWT nad celým súborom naraz. Náš program beží akceptovateľne rýchlo vďaka predspracovaniu vstupu parserom a využití abecedy slov, ktorá skracuje vstup pre BWT. Veľmi dôležitým dôsledkom použitia parseru je aj zníženie hodnoty AML. Použitie parseru si však vyžaduje použitie algoritmov, ktoré nepracujú klasicky s bajtovou abecedou o veľkosti 256 znakov, ale zvládajú aj abecedu 4 bajtovú. Pri voľbe slov ako abecedy je pre uvedené súbory veľkosť parserom vytvorenej abecedy okolo 50 tisíc.

4 Korpus

Náš korpus tvoria tri súbory, ktoré pochádzajú z indexovacieho systému *egothor*. Prvý je tvorený zreťazenými stánkami v angličtine, druhý v slovinštine a tretí v čestine. V Tabuľke 1 sú uvedené informácie o týchto súboroch. AML je miera repetitívnosti súboru. Informácie o kompresnom pomere programu XBW na štandartných korpusoch Calgary, Cantebury a Silesia sú uvedené v [5].

5 Výsledky

Najprv ukážeme výsledky programu XBW pre rôzne kompresné metódy a vplyv parseru na výsledky. Ďalej sa budeme venovať vplyvu zmeny abecedy. Nakoniec porovnáme výsledky XBW pri optimálnych parametroch s bežne používanými kompresnými programami Gzip, Rar a Bzip2.

Všetky nasledujúce výsledky boli získané za použitia aritmetického kodéru. BWT bežala vždy nad celým vstupom naraz a za ňou nasledoval MTF a RLE (parameter RLE=3). PPM bežalo s parametrami PPM_ exlusions=off a PPM_order=5.

Veľkosť skomprimovaných súborov zahŕňa aj zakódovaný slovník. Slovík je vytváraný vždy keď je použitý parser. Kompresný pomer je uvádzaný v bitoch na bajt.

Čas behu programu bol meraný pod operačným systémom Linux a zobrazuje súčet systémového a užívateľského času. To znamená, že zobrazujeme čas bez čakania na disk. Merania prebehli na PC s procesorom AMD Athlon X2 4200+ s 2GB operačnej pamäte. Údaje sú v megabajtoch za sekundu a uvažuje sa pri kompresii aj dekompresii dekomprimovaná veľkosť súboru.

Tabuľka 2 ukazuje výsledky kompresného pomeru pre rôzne metódy pre abecedu znakov. Tieto výsledky ukazujú vplyv XML parsera, ktorý zlepšuje kompresný pomer zhruba o desať percent.

Nasleduje Tabuľka 3, ktorá ukazuje rýchlosť programu bez parseru a za použitia parseru v XML móde, ktorý vytváral slovník znakov. Výsledky ukazujú, že

bpB	Parser=xml											
	znaky				slabiky				slová			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,894	2,073	2,098	1,320	0,854	1,796	1,841	N/A	0,857	1,683	1,654	N/A
xml_en	0,874	1,915	2,115	1,239	0,836	1,626	1,785	N/A	0,830	1,514	1,558	N/A
xml_sl	0,700	1,850	1,797	1,129	0,664	1,559	1,541	N/A	0,668	1,457	1,390	N/A
TOTAL	0,822	1,957	1,998	1,234	0,783	1,672	1,723	N/A	0,785	1,563	1,539	N/A

Tabuľka 4. Vplyv abecedy na kompresný pomer.

MB/s	Kompresia								Dekompresia							
	XML parser - znaky				XML parser - slová				XML parser - znaky				XML parser - slová			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,457	2,668	1,418	0,091	1,587	0,279	1,477	N/A	2,577	3,156	3,415	0,096	3,986	3,951	3,923	N/A
xml_en	0,544	2,915	1,249	0,104	2,009	0,920	1,093	N/A	2,705	3,397	3,606	0,110	4,006	4,443	4,523	N/A
xml_sl	0,500	2,915	1,497	0,091	1,566	0,443	1,349	N/A	3,012	3,299	3,672	0,097	4,241	4,157	4,237	N/A
TOTAL	0,491	2,810	1,397	0,094	1,666	0,399	1,319	N/A	2,747	3,263	3,548	0,099	4,076	4,135	4,167	N/A

Tabuľka 5. Vplyv abecedy na rýchlosť programu.

bpB	Parser=off	Parser=text			Parser=XML		
		znaky	slabiky	slová	znaky	slabiky	slová
xml_cz	0,907	0,906	0,859	0,862	0,894	**0,854**	0,857
xml_en	0,886	0,887	0,842	0,836	0,874	0,836	**0,830**
xml_sl	0,710	0,710	0,669	0,672	0,700	**0,664**	0,668
TOTAL	0,834	0,833	0,789	0,790	0,822	**0,783**	0,785

Tabuľka 6. Kompresný pomer pre BWT.

v skoro všetkých prípadoch parser program spomaľuje. Dôvodom je, že musíme navyše pracovať so slovníkom a čas, ktorý je ušetrený vďaka miernemu skráteniu vstupu, nestačí kompenzovať čas práce so slovníkom. Výnimkou je kompresia pri použití BWT. Skrátenie vstupu a zníženie jeho repetitívnosti výrazne urýchli BWT, ktorá je časovo najnáročnejšiou časťou blokovej kompresie.

Metóda bežne používaná na kompresiu textov je použitie slov ako symbolu abecedy. V Tabuľke 4 je zobrazený vplyv abecedy na kompresný pomer. Pri textových dátach v angličtine je dosiahnutý najlepší kompresný pomer pri použití slov a metódy BWT. Pri češtine a slovinštine sú lepšie slabiky. Dôvodom je, že čeština a slovinština majú bohaté tvaroslovie. Jedno slovo sa vyskytuje v texte v rôznych tvaroch a každý tvar sa pridáva do slovníka. Pri použití slabík sa do slovníka pridá slovný základ, ktorý môže byť tvorený aj viacerými slabikami, a časované koncovky. Tieto posledné slabiky sú však spoločné pre viaceré slová a preto je aj ich výskytov v texte viac. Pre algoritmy LZx sú výrazne najlepšie slová.

Vplyv veľkej abecedy na rýchlosť je rôznorodý a je možné ho vidieť v Tabuľke 5. Pre všetky algoritmy je dekompresia rýchlejšia pri použití slov ako znakov. Na druhej strane dekomprimácia, ak bol použitý parser so slovami je stále pomalšia ako dekomprimácia bez použitia parseru viď Tabuľka 2. Na druhej strane použitie slov urýchľuje kompresiu iba pri použití BWT. Výrazné zrýchlenie pri BWT je dané skrátením vstupu asi trojnásobne a znížením AML. Výsledky pre PPM pre slová nie sú zverejnené, lebo algoritmus pri nich do hodiny nedobehol.

Na predchádzajúcich výsledkoch je možné vidieť, že najlepší kompresný pomer má algoritmus BWT. Taktiež je evidentné, že parser zlepšuje kompresný pomer pre všetky algoritmy. Najrýchleší pri kompresii je algoritmus LZC a LZSS pri dekompresii.

Naše primárne kritérium je kompresný pomer, a nakoľko metóda BWT má presvedčivo najlepší kompresný pomer, venujeme jej najviac pozornosti. V prípade, že prioritná je rýchlosť kompresie, je vhodné voliť slovníkové metódy.

Tabuľka 6 obsahuje porovnanie kompresných pomerov pri rôznych voľbách parseru, ktorá ukazuje, že najvhodnejšie je delenie na slová pre súbory v angličtine a delenie na slabiky pre súbory v čestine a slovinštine. Výber slov, respektíve slabík závisí na veľkosti

MB/s	Kompresia							Dekompresia						
	Parser=off	Parser=text			Parser=XML			Parser=off	Parser=text			Parser=XML		
		znaky	slabiky	slová	znaky	slabiky	slová		znaky	slabiky	slová	znaky	slabiky	slová
xml_cz	0,368	0,324	1,056	**1,767**	0,457	1,073	1,587	4,260	2,628	4,277	**4,817**	2,577	3,710	3,986
xml_en	0,419	0,364	1,225	**2,128**	0,544	1,330	2,009	4,417	2,494	4,612	**4,764**	2,705	3,981	4,006
xml_sl	0,386	0,331	1,102	**1,790**	0,500	1,135	1,566	4,918	2,639	4,686	**5,442**	3,012	3,685	4,241
TOTAL	0,386	0,336	1,110	**1,853**	0,491	1,150	1,666	4,509	2,598	4,494	**5,002**	2,747	3,765	4,076

Tabuľka 7. Rýchlosť BWT.

Sekundy	kompresia				dekompresia			
	Parser	BWT	MTF	RLE	Parser	BWT	MTF	RLE
xml_cz	4,668	7,788	0,748	0,72	1,98	0,764	0,868	1,328
xml_en	2,364	3,800	0,388	0,448	1,112	0,716	0,440	0,796
xml_sl	3,352	7,404	0,496	0,504	1,592	0,676	0,556	0,916
TOTAL	10,384	18,992	1,632	1,672	4,684	2,156	1,864	3,04

Parser - textový mód za použitia slov; BWT - parameter Itoh; RLE - verzia 3

Tabuľka 8. Čas behu jednotlivých častí programu XBW.

bpB	XBW	Gzip	Bzip2	Rar
xml_cz	**0,108**	0,212	0,176	0,145
xml_en	**0,104**	0,208	0,162	0,107
xml_sl	**0,084**	0,171	0,141	0,115
TOTAL	**0,099**	0,197	0,160	0,125

XBW: textový mód parseru so slovami, Kaov algoritmus pre BWT
Gzip: gzip -9; Rar: rar -m5; Bzip2: bzip2 -9

Tabuľka 9. Kompresný pomer programov.

MB/s	Kompresia				Dekompresia			
	XBW	Gzip	Bzip2	Rar	XBW	Gzip	Bzip2	Rar
xml_cz	1,732	**10,320**	3,170	2,708	4,087	**25,004**	9,430	3,955
xml_en	2,058	**11,587**	3,454	2,689	4,309	**46,926**	11,722	6,137
xml_sl	1,758	**13,713**	3,245	3,190	4,614	**46,986**	13,132	4,775
TOTAL	1,812	**11,634**	3,262	2,853	4,313	**34,629**	11,045	4,640

XBW: textový mód parseru so slovami, Kaov algoritmus pre BWT
Gzip: gzip -9; Rar: rar -m5; Bzip2: bzip2 -9

Tabuľka 10. Porovnanie rýchlostí programov.

súborov a na tvarosloví daného jazyka. Pre jazyky, ktoré majú bohaté tvaroslovie, sú výhodnejšie slabiky. Pre kratšie súbory je taktiež vhodnejšie delenie na slabiky. Voľba delenia na slová, respektíve slabiky, ovplyvňuje počet výskytov symbolov zo slovníka vo vstupnom texte. V programe XBW sú implementované viaceré verzie delenia na slabiky. Vo výsledkoch je použitá voľba *Left*. Podrobnejšie informácie sa nachádzajú v [5]. Zaujímavý je malý vplyv XML módu parseru na kompresný pomer. Toto nie je zapríčené nesprávnou implementáciou parseru, ale vlastnosťami BWT pre veľké bloky. Napríklad pre LZx metódy je vplyv parseru veľmi výrazný. Podrobnejšie výsledky sú opäť v [5].

Nasleduje Tabuľka 7, ktorá ukazuje vplyv parseru na rýchlosť programu. Jednoznačne najrýchlejšie pri kompresii sú slová. Pri dekompresii už až taký veľký rozdiel v rýchlosti nie je. To je práve dôsledkom najvýraznejšieho skrátenia vstupu pre BWT pri použití slov. Kvôli rýchlosti je vhodnejšie použiť parser v textovom móde miesto XML módu najmä pre slová.

Existuje mnoho algoritmov používaných na triedenie suffixov pri BWT. Voľba takéhoto algoritmu silne ovplyňuje celkovú rýchlosť kompresie. Bez použitia parseru zaberá triedenie suffixov pri veľkých blokoch až 90% celkového času behu programu. Viac detailov je v práci [10]. Pre všetky testované súbory je najrýchlejšia Kaova modifikácia Itohovho algoritmu [8].

Pripomíname, že tento algoritmus bol použitý pri všetkých meraniach pri použití BWT.

Čas behu jednotlivých častí progmamu je v Tabuľke 8. Tieto časy ukazujú, v ktorých častiach je najviac priestoru na zrýchlenie.

6 Porovnanie s inými programami

Na porovnanie uvádzame výsledky programov Gzip, Rar a Bzip2. Programy určené na kompresiu XML dát ako napríklad XMLPPM a Xmill nie sú schopné spracovať nevalidné XML súbory. Preto nebolo možné získať ich výsledky na našich dátach. Pri programoch Gzip, Rar a Bzip2 sme použili parametre na najlepšiu možnú kompresiu. V Tabuľke 9 sú uvedené kompresné pomery. Program XBW komprimuje všetky súbory najúčinnejšie a je výrazne najlepší pre súbory, ktoré nie sú v angličtine.

V Tabuľke 10 sú výsledky rýchlostí kompresie a dekompresie. Jednoznačne najrýchlejší je program Gzip. Tento program má však najhorší kompresný pomer a preto rýchlosť nášho programu XBW porovnáme iba s Rar a Bzip2. Kompresia XBW trvá necelý dvojnásobok času minima Rar a Bzip2. Dekompresia u XBW je porovnateľne rýchla ako u Rar. Bzip2 je pri dekompresii asi trojnásobne rýchlejší.

Rýchlosť XBW je dostatočná na bežné použitie, avšak nedosahuje rýchlosť dnešných pevných diskov, a preto keď je prioritná rýchlosť, je vhodnejšie použiť program založený na slovníkových metódach ako napríklad Gzip. XBW má najlepší kompresný pomer, a preto jeho použitie je vhodné najmä na dlhodobú archiváciu.

7 Budúci vývoj

Pri ďalšom vývoji XBW sa chceme sústrediť na dva smery. Prvým je tvorba parseru, ktorý by bol použiteľný aj na binárne dáta. Druhým smerom je zrýchľovanie programu, kde je opäť najviac priestoru pri parsri.

Referencie

1. Lánský, J. and Žemlička M., Compression of Small Text Files Using Syllables. Technical Report 2006/1, Prague, Czech Republic, 2006
2. The Open Group Base, iconv. Specifications Issue 6. IEEE Std 1003.1, 2004
3. Burrows M. and Wheeler D. J., A Block-Sorting Lossless Data Compression Algorithm. Technical Report 124, 1994
4. Cleary J.G. and Witten I. H., Data Compression Using Adaptive Coding and Partial String Matching. IEEE Transactions on Communications, COM, 32, 4, April 1984, 396–402
5. Lánský et al. J., Xbw - Word-Based Compression of Non-Valid XML Documents. `http://xbw.sourceforge.net/`
6. Galamboš L., Egothor. `http://www.egothor.org`
7. Huffman D.A., A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE, 40, 9, 1952, 1098–1101
8. Kao T.-H., Improving Suffix-Array Construction Algorithms with Applications. Diplomová práca. Gunma University, Japonsko, 2001
9. Kärkkäinen J. and Sanders P., Simple Linear Work Suffix Array Construction. In Proc. 13th International Conference on Automata, Languages and Programming, Springer, 2003
10. Šesták R., Suffix Arrays for Large Alphabet. Diplomová práca, Univerzita Karlova, Praha, 2007
11. Ziv J. and Lempel A., A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, 23, 3, 1977, 337–343

Classification of EEG data using fuzzy *k*-NN ensembles*

David Štefka and Martin Holeňa

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
{stefka,martin}@cs.cas.cz

Abstract. *Ensemble methods try to improve quality of classification by creating multiple classifiers and aggregating their outputs. In this paper, we present the use of ensemble methods for classification of EEG data from the project "Building Neuroinformation Bases, and Extracting Knowledge from them", within which a possibility of preventing drivers' microsleeps is studied. A multiple feature subset ensemble method is used to improve the quality of classification of a fuzzy k-nearest neighbor classifier. Two different aggregation schemes are used – the mean value aggregation algorithm outperforming the Sugeno fuzzy integral aggregation algorithm.*

1 Introduction

This work is a part of the project "Building Neuroinformation Bases, and Extracting Knowledge from them" of the Ministry of Education, Youth and Sports of the Czech Republic. Within the scope of this project, possibilities of preventing drivers' microsleeps by analyzing the drivers' EEG spectra and identifying sleepiness is studied. The aim of this paper is to determine whether classifier combining can improve the classification of the EEG data. The results show that classifier combining can yield improvement in both average and variance of error rate of the classification.

The paper is structured as follows. In Section 2, basic information about the project and the EEG data is given. In Section 3, basic aspects of classifier combining are introduced. Section 4 briefly describes some aspects of classification of the particular EEG data. Section 5 contains experimental results and their discussion, and Section 6 then concludes the paper.

2 Data description

The data used in this paper come from an experiment performed at the Laboratory of System Reliability, Faculty of Transportation Sciences, Czech Technical University, Prague. During the experiment, 24 probands with lack of sleep were measured a 19-channel EEG for approx. 1 hour. From these data, approx. 800 segments were selected by an expert neurophysiologist for further processing. The raw EEG data were transformed into frequency spectral densities for frequencies 0-30 Hz, step 1 Hz, using the Burg filter of order 20.

Although 19-channel EEG was measured, resulting in $19 \cdot 31 = 589$-dimensional feature vector for each segment, for further processing, only two channels (corresponding to electrodes T3 and O1) were used, resulting in 62-dimensional feature vector for each segment. Each segment was also classified by the expert into one of the following four classes (mental states):

- *oculi aperti* – proband has open eyes (209 segments)
- *oculi clausi* – proband has closed eyes (33 segments)
- *raven* – mentation; proband is solving part of the Raven test (215 segments)
- *somnolence* – sleepiness (338 segments)

3 Classifier combining

Throughout the rest of the paper, we use the following notation. Let $\mathcal{X} \subseteq \mathbf{R}^n$ be a n-dimensional *feature space*, an element $\boldsymbol{x} \in \mathcal{X}$ of this space is called *pattern*, and let $C_1, \ldots, C_N \subseteq \mathcal{X}$ be disjoint sets called *classes*. We call a *classifier* any mapping ϕ from the following:

- *crisp classifier* – $\phi : \mathcal{X} \to \{1, \ldots, N\}$, where $\phi(\boldsymbol{x})$ is the predicted class label of pattern $\boldsymbol{x}$.
- *possibilistic classifier* – $\phi : \mathcal{X} \to [0,1]^N$, where $\phi(\boldsymbol{x}) = (\mu_1, \ldots, \mu_N)$ are *degrees of classification* to each class. If $\sum_i \mu_i = 1$, the classifier is called *probabilistic*.

The purpose of combining different classifiers in order to get the final class prediction is that the final combined classifier can perform its classification task much better than any of the individual classifiers in the team. Basically, there are two main approaches to classifier combination – *classifier selection* (we use

* The research reported in this paper was partially supported by the Program "Information Society" under project 1ET100300517 (D. Štefka), by the Czech Ministry of Education's grant ME701 "Building Neuroinformation Bases, and Extracting Knowledge from them" (D. Štefka), by the grant No. 201/05/0325 of the Grant Agency of the Czech Republic (M. Holeňa), and by the Institutional Research Plan AV0Z10300504.

RandomMFS. Create an ensemble $(\phi_1, \ldots, \phi_r)$ using a training set $\mathcal{T} = \{\boldsymbol{x_i} \in \mathcal{X} \mid i = 1, \ldots, l\}$, $\mathcal{X} \subseteq \mathbf{R}^n$. Each classifier uses exactly t features.

1. Set $j = 1$.
2. Generate a random mask $\boldsymbol{d} = (d_1, \ldots, d_n)$, $d_i = 1$ indicating that the i-th feature will be used, $d_i = 0$ indicating it will not be used, such that $\sum_{i=1}^{n} d_i = t$. Create a training set $\mathcal{T}_j$ using only those features, for which $d_i = 1$.
3. Create a classifier ϕ_j using training set $\mathcal{T}_j$. If $j < r$, increment j and return to (2), otherwise end with output $(\phi_1, \ldots, \phi_r)$.

Fig. 1. The RandomMFS algorithm.

some rule to determine which classifier to use for the current pattern; only this "expert" classifier is then used for the final prediction), and *classifier aggregation* (where all the classifiers in the team are used for the final decision). In this paper, we deal with classifier aggregation only. Classifier aggregation consists of two steps – first, we have to create a team of classifiers, and then we have to choose some method to aggregate the results of the team.

3.1 Ensemble methods

In the literature, many methods for creating a team of classifiers are described – most of them create an *ensemble* (a set of classifiers of the same type, which differ only in their training sets or in their parameters). The most common ensemble methods are bagging [1], boosting [2], or multiple feature subset (MFS) methods [3].

For classification of the EEG data, the ensemble was created using an algorithm called *RandomMFS*. This algorithm generates an ensemble of classifiers using random subsets of features; it is described in detail in Fig. 1.

3.2 Ensemble aggregation

After an ensemble of possibilistic classifiers $(\phi_1, \ldots, \phi_r)$ has been constructed, we have to use some aggregation strategy to aggregate the results of the individual classifiers. The output of an ensemble can be structured to a $r \times N$ matrix, called *decision profile* (DP):

$$DP(\boldsymbol{x}) = \begin{pmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \vdots \\ \phi_r(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,N} \\ \mu_{2,1} & \mu_{2,2} & \cdots & \mu_{2,N} \\ & & \ddots & \\ \mu_{r,1} & \mu_{r,2} & \cdots & \mu_{r,N} \end{pmatrix} \quad (1)$$

The $i-$th row of the $DP(\boldsymbol{x})$ is the output of the corresponding classifier ϕ_i, and the $j-$th column contains the degrees of classification of $\boldsymbol{x}$ to the corresponding class C_j given by all the classifiers. In other words, $\mu_{i,j}$ is the degree of classification of $\boldsymbol{x}$ to class C_j, provided by classifier ϕ_i.

An overview and experimental comparison of different aggregation rules can be found in [4, 5]. For classification of the EEG data, we used two aggregation methods – mean value [4] and Sugeno fuzzy integral [4, 5].

The mean value aggregation algorithm computes average degree of classification to each class – the aggregated degree of classification to class C_j is computed as the average of the degrees of classification to class C_j through all the classifiers, i.e.

$$\mu_j = \frac{1}{r} \sum_{i=1}^{r} \mu_{i,j}. \quad (2)$$

The Sugeno integral aggregation algorithm takes into account the quality of the individual classifiers. The information about quality of the classifiers is incorporated into a fuzzy measure $g : \mathcal{P}(\{\phi_1, \ldots, \phi_r\}) \to [0, 1]$, $\mathcal{P}$ denoting the power set, and each column of the decision profile is then fuzzy-integrated with respect to g. Although any fuzzy measure can be used, *λ-fuzzy measure* [6] is used most often. For λ-fuzzy measure, we need to define so-called *fuzzy densities* $g(\phi_1), \ldots, g(\phi_r) \in [0, 1]$, which represent the importance (or quality) of the individual classifiers. For our application, we used $g(\phi_i) = 1 - Err(\phi_i)$, where $Err(\phi_i)$ denotes train error rate of the classifier ϕ_i.

After the fuzzy densities are defined, λ is calculated as the only real non-zero root greater than -1 of the equation

$$\lambda + 1 = \prod_{i=1}^{r} (1 + \lambda g(\phi_i)). \quad (3)$$

The fuzzy densities determine values of the fuzzy measure g for the singletons, λ is then used to compute all the remaining values of g. The algorithm of Sugeno fuzzy integral aggregation using λ-fuzzy measure g is described in Fig. 2.

4 Classification of the EEG data

In order to determine which classifier algorithm to use for the creation of an ensemble, we performed some preliminary classification of the data using some common algorithms from the Weka framework [7]. In these tests, instance-based algorithms were particularly successful, the K* algorithm [8] being the best of them.

However, the time complexity of the K* algorithm was too high to use it for classifier combining, and so

Sugeno fuzzy integral. Aggregate the decision profile (1) to $\Phi(\boldsymbol{x}) = (\mu_1, \ldots, \mu_N)$.

1. Compute the fuzzy densities $g(\phi_i) = 1 - Err(\phi_i)$.
2. If $\sum_{i=1}^{r} g_i = 1$, set $\lambda = 0$, otherwise calculate λ as the only non-zero root greater than -1 of the equation (3).
3. Set $j = 1$.
4. Sort the values in the j-th column of $DP(\boldsymbol{x})$ in ascending order, denoting the sorted values $a_{(1)}, \ldots, a_{(r)}$. The corresponding classifiers will be denoted $\phi_{(1)}, \ldots, \phi_{(r)}$, and the corresponding fuzzy densities $g(\phi_{(1)}), \ldots, g(\phi_{(r)})$.
5. Calculate the values of the fuzzy measure g:
   - Set $g_r = g(\phi_{(r)})$.
   - For $i = r, \ldots, 2$, calculate recursively
   $$g_{i-1} = g_i + g(\phi_{(i-1)}) + \lambda g_i g(\phi_{(i-1)}).$$
6. Compute the aggregated degree of classification of $\boldsymbol{x}$ to class C_j:
   $$\mu_j = \max_{i=1}^{r} \{\min\{a_i, g_i\}\}$$
7. If $j < N$, increment j and return to (4), otherwise end with output $\Phi(\boldsymbol{x}) = (\mu_1, \ldots, \mu_N)$.

Fig. 2. The Sugeno fuzzy integral aggregation algorithm [4, 5].

we decided to use the fuzzy k-nearest neighbor classifier [9], instead. The algorithm of fuzzy k-nearest neighbor classifier is described in Fig. 3. For our application, we used Euclidean metric and fine-tuned the constants to $k = 5$ and $m = 2$.

5 Results

In this section, we present results of classification of the EEG data using ensembles of fuzzy k-nearest neighbor classifiers. The ensembles were designed by the RandomMFS algorithm. The size of the ensemble, i.e. the number of the individual classifiers will be denoted r, the number of features each classifier used will be denoted t.

We studied the performance of the ensemble aggregated by two aggregation schemes – mean value and Sugeno integral – for $t = 5, 10, 15, 20$ and $r = 20$ to 200, step 20. We measured the mean value and standard deviation of the ensemble's error rate (in %) from 10-fold crossvalidation. The results for the mean value and Sugeno integral aggregators are shown in Fig. 4. The constant dashed line represents result of the unique, non-combined fuzzy 5-nearest neighbor classifier with $m = 2$, which uses all features – this classifier will be denoted as NC.

From the results, we can see that for sufficiently large ensemble ($r > 140$) and adequate t ($t = 15$ provides the best results), the ensemble performs much better than the NC classifier. For $r = 200, t = 15$, we obtain more than 3% improvement of the average error rate. Moreover, the variance of error rates for ensembles is lower than the variance of error rates for NC classifier.

Fuzzy k-nearest neighbor algorithm. Compute the degree of classification of $\boldsymbol{x}$ to each class $C_1, \ldots, C_N$ using a training set $\mathcal{T} = \{\boldsymbol{x_i} \in \mathcal{X} \mid i = 1, \ldots, l\}$. Parameters of the algorithm: k – number of nearest neighbours taken into account, m – fuzzifier (affecting the influence of the distance between the patterns), metric $||\cdot||$

1. Find the set $\mathcal{K}$ of k patterns from $\mathcal{T}$ closest to pattern $\boldsymbol{x}$ under $||\cdot||$. Let $\mathcal{K}_j$ denote set of patterns from $\mathcal{K}$ which belong to class C_j. Patterns from $\mathcal{K}$ and $\mathcal{K}_j$ will be denoted $\boldsymbol{y}$.
2. For $j = 1, \ldots, N$, compute the degree of classification of $\boldsymbol{x}$ to class C_j using the following formula:
   $$\mu_j = \frac{\sum_{\boldsymbol{y} \in \mathcal{K}_j} 1/||\boldsymbol{x} - \boldsymbol{y}||^{2/(m-1)}}{\sum_{\boldsymbol{y} \in \mathcal{K}} (1/||\boldsymbol{x} - \boldsymbol{y}||^{2/(m-1)})}$$
3. End with output $\phi(\boldsymbol{x}) = (\mu_1, \ldots, \mu_N)$.

Fig. 3. The fuzzy k-nearest neighbor algorithm [9].

If we study the influence of t on the classification, we can see the following. For $t = 5$, the quality of the individual classifiers is too low, and the results of classifier combining are unsatisfactory (comparable to, or even worse than the NC classifier). For $t = 10$, results for Sugeno integral aggregator are still unsatisfactory, while mean value aggregator already provides satisfactory results. For $t = 15$, both aggregators obtain the best results. For $t = 20$, results are still usable, but slightly worse than for $t = 15$. The reason for this may be that the complexity of the individual classifiers is becoming too high, and larger ensembles are needed.

Although the Sugeno integral is thought to be very successful aggregation operator, it was outperformed by the mean value aggregator for this particular data. However, both these aggregation operators belong to a more general class of the so-called *fuzzy t-conorm integral* [10]. Using fuzzy t-conorm integral, further improvement of classification may be made.

6 Summary

In this paper, we presented the usage of classifier combining for improvement of classification of EEG data from the project "Building Neuroinformation Bases, and Extracting Knowledge from them". We showed that classifier combining can yield improvements in

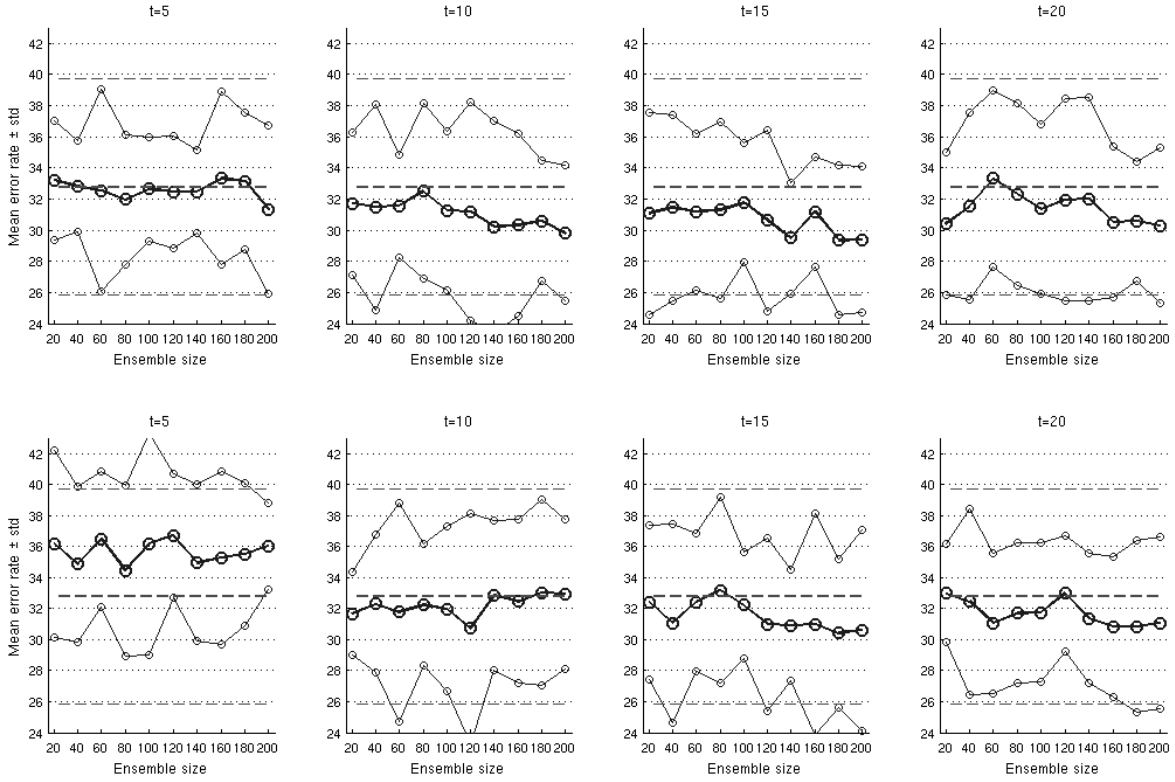


Fig. 4. Mean error rate ± standard deviation of an ensemble created by the RandomMFS algorithm, using $t = 5, 10, 15, 20$ features, using different number of classifiers (Ensemble size). The ensemble was aggregated using mean value aggregator (top four graphs) and Sugeno integral aggregator (bottom four graphs) – solid line. The dashed line corresponds to the NC classifier.

the classification of the data – in particular, using ensembles of fuzzy k-nearest neighbor classifiers, designed by random multiple feature subset method, the error rate of classification can be lowered by about 3% in comparison to a non-combined fuzzy k-nearest neighbor classifier.

References

1. Breiman L., Bagging Predictors. Machine Learning, 24, 2, 1996, 123–140
2. Freund Y. and Schapire R.E., Experiments with a New Boosting Algorithm. In International Conference on Machine Learning, 1996, 148–156
3. Bay S.D., Nearest Neighbor Classification from Multiple Feature Subsets. Intelligent Data Analysis, 3, 3, 1999, 191–209
4. Kuncheva L.I., Bezdek J.C., and Duin R.P.W., Decision Templates for Multiple Classifier Fusion: an Experimental Comparison. Pattern Recognition, 34, 2, 2001, 299–314
5. Kuncheva L.I., Fuzzy versus Nonfuzzy in Combining Classifiers Designed by Boosting. IEEE Transactions on Fuzzy Systems, 11, 6, 2003, 729–741
6. Chiang J.-H., Aggregating Membership Values by a Choquet-Fuzzy-Integral Based Operator. Fuzzy Sets Syst., 114, 3, 2000, 367–375
7. Witten I.H. and Frank E., Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, second edition, 2005
8. Cleary J.G. and Trigg L.E., K*: an Instance-Based Learner Using an Entropic Distance Measure. In Proc. 12th International Conference on Machine Learning, Morgan Kaufmann, 1995, 108–114
9. Keller J.M., Gray M.R., and Givens, Jr. J.A., A Fuzzy k-Nearest Neighbor Algorithm. IEEE Transactions on Systems, Man, and Cybernetics, SMC-15, 4, 1985, 580–585
10. Grabisch M. and Nguyen H.T., Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference. Kluwer Academic Publishers, Norwell, MA, USA, 1994

Using support vector machines in fuzzy classification*

Zdeněk Vyoral[1] and Martin Holeňa[2]

[1] Student of Czech Technical University, Faculty of Nuclear Science and Physical Engineering
Břehova 7, Prague, Czech Republic
zdenek.vyoral@atlas.cz
[2] Institute of Computer Science, Czech Academy of Sciences
Pod Vodárenskou věží 2, Prague, Czech Republic
martin@cs.cas.cz

Abstract. *Fuzzy classification is one of methods used for pattern classification, which is germane to many engineering applications. An output from fuzzy classification is an assignment of patterns to fuzzy classes. There are several methods for fuzzy classification; in this paper we propose a new method based on the soft margin support vector machines classifier (C-SVM). These classifiers are based on statistical learning theory and they are widely used in pattern classification. In our approach, the decision boundary and slack variables obtained from C-SVM are used for the definition of a new optimization problem. The goal is to find an optimal parametrized transformation function $\mathcal{T}$, which transforms the distance of a pattern from decision boundary to its membership degree. Quadratic programming is used to find suitable values for the parameters of $\mathcal{T}$. The development of the full scope of this new fuzzy classification method is still in progress.*

1 Introduction

Classification is a method for separation of patterns to several *output classes*. We model patterns x_i as vectors in $\mathbb{R}^s$, where s is a number of features. In classification, it is expected that we know the assignments y_i of some patterns $x_i \in \mathbb{R}^s$ to their output classes. A set of these assignments is called a *training set*; vectors x_i are termed *inputs* and assigned values y_i *outputs*. Given some new pattern x_i, we want to predict the corresponding y_i using the training set.

In *fuzzy classification* output classes are represented as fuzzy sets $(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m)$ [3]. Using the simple criterion of *maximum membership*; the output class $\mathcal{A}_i$ that the data sample $x_0 \in \mathbb{R}^s$ most closely resembles is found by:

$$i = \underset{i=1,\ldots,m}{\operatorname{argmax}} \{\mu_{\mathcal{A}_1}(x_0), \mu_{\mathcal{A}_2}(x_0), \ldots, \mu_{\mathcal{A}_m}(x_0)\}. \quad (1)$$

There are several methods for fuzzy classification; e.g. *neuro-fuzzy methods* [4] or *genetic-algorithm* based rule selection [2]. We are going to introduce a new method based on C-SVM classification.

* The research reported in this paper has been supported by the grant No. 201/05/0325 of the Grant Agency of the Czech Republic (M. Holeňa).

2 Employed concepts

2.1 Classification with support vector machines

In this section we sketch how support vector machines (SVM) and C-SVM classifiers work. You can find the detailed description and additional references in the monograph [5]. We omit it in this paper because of space restrictions.

In the sequel, we restrict ourselves to *only two* output classes. For two output classes the *training set* (*training examples*) can be defined by

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathbb{R}^s \times \{\pm 1\}. \quad (2)$$

where y_i is $+1$ if x_i belongs to the first class and -1 for the other class.

Given an input space $\mathcal{H}$ equipped with a scalar product $\langle\rangle$ any *hyperplane* in $\mathcal{H}$ can be written as

$$\{x \in \mathcal{H} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}, \mathbf{w} \in \mathcal{H}, b \in \mathbb{R}. \quad (3)$$

In this formulation, $\mathbf{w}$ is a vector orthogonal to the hyperplane. Parameter b determines the position of the hyperplane. For any $x \in \mathcal{H}$ number $\langle \mathbf{w}, \mathbf{x} \rangle / \|\mathbf{w}\| \in \mathbb{R}$ is the length of its projection to the direction of $\mathbf{w}$. If we multiply $\mathbf{w}$ and b by the same non-zero constant, we obtain the same hyperplane. This superfluous freedom can be abolished by using *canonical hyperplanes*.

Definition 1 (Canonical Hyperplane). *The pair* $(\mathbf{w}, b) \in \mathcal{H} \times \mathbb{R}$ *is called a* canonical form *of the hyperplane (3) with respect to* $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{H}$*, if it is scaled so that*

$$\min_{i=1,\ldots,m} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1. \quad (4)$$

Let us consider that the given problem is *linearly separable* in the input space first. A decision function $f_{\mathbf{w},b}$ which correctly classifies all the examples $(f_{\mathbf{w},b}(x_i) = y_i)$ can be defined by:

$$f_{\mathbf{w},b} : \mathcal{H} \to \{\pm 1\};$$
$$\mathbf{x} \to f_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, x \rangle + b), \quad (5)$$

In SVM learning algorithms, a crucial role is played by a notion of a *margin*.

Definition 2 (Margin). *For a hyperplane* $\{\mathbf{w}, b\}$ *and a sequence* $\{(x_i, y_i) \ \ | i \in \hat{m}\}$*, the minimum value*

$$\rho_{\mathbf{w},b} \ := \ \min_{i=1,\ldots,m} \ \ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)/\|\mathbf{w}\| \qquad (6)$$

is called the margin of $\{(x_i, y_i) \ \ | i \in \hat{m}\}$*. If the latter is omitted, it is understood that the training set is meant.*

According to the definition (4), all patterns have distance at least 1 from the hyperplane. Hence, for all correctly separated patterns hold *separation constraints*:

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1, \ i \in \hat{m}. \qquad (7)$$

Among all canonical hyperplanes separating the data, there exists a unique optimal one, distinguished by the maximum margin. It can be constructed by solving the *primal optimization problem*:

$$\underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\text{minimize}} \ \ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \qquad (8)$$

subject to (7).

Let us explain the motivation for minimizing the length of $\mathbf{w}$. If $\|\mathbf{w}\|$ is 1, then the left hand side of (7) would equal the distance from $\mathbf{x}_i$ to the hyperplane. In general, we have to divide $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ by $\|\mathbf{w}\|$ to transform it into this distance. Hence, if we can satisfy (7) for all $i = 1, \ldots, m$ with a $\mathbf{w}$ of minimal length, then the overall margin will be maximized.

Primal optimization problem (8) is a *constrained optimization problem*. In this case, the *Lagrangian* takes the following form:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1) \qquad (9)$$

L has to be minimized with respect to the *primal variables* $\mathbf{w}$ and b and maximized with respect to the *dual variables* α_i (in other words, a saddle point has to be found).

At the *saddle point*, the derivatives of L with respect to the primal variables must vanish, which leads to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \text{ and } \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \qquad (10)$$

So, the *solution vector* $\mathbf{w}$ has an *expansion* in terms of training examples. Only the Lagrange multipliers α_i that are non-zero, correspond to constraints (7) which are precisely met. The corresponding patterns $\mathbf{x}_i$ are called *support vectors*. Thats why these classifiers are called *SVM classifiers*.

Substituting the conditions for the extreme (10), into the Lagrangian (9), we arrive at the *dual form* of the optimization problem:

$$\underset{\alpha\in\mathbb{R}^m}{\text{maximize}} \ W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle, \qquad (11)$$

$$\text{subject to: } \ \alpha_i \geq 0, i \in \hat{m} \text{ and } \ \sum_{i=1}^{m} \alpha_i y_i = 0. \qquad (12)$$

On the substitution of the expansion (10) into the decision function (5), we obtain a final form of the *decision function*, which can be evaluated in terms of scalar products:

$$f(x) = \operatorname{sgn} \left(\sum_{i=1}^{m} \alpha_i y_i \left\langle \mathbf{x}, \mathbf{x_i} \right\rangle + b \right). \qquad (13)$$

2.2 Non-linearly separable case

SVM classifiers, as we presented them, would be able to solve only linearly separable problems. To allow for much more general decision surfaces, *kernels* [5] can be introduced to nonlinearly transform the input data into a high-dimensional feature space using a map $\mathbf{\Phi} : x_i \rightarrow \mathbf{x}_i$, where $\mathbf{x}_i$ is from some feature space $\mathcal{H}$ and $x_i \in \mathbb{R}^s$. The feature space $\mathcal{H}$ and kernel k can constructed so that the classes become linearly separable in it.

The new proposed method doesn't change whether a kernel was used or not. Indeed, recall that a scalar product in the input space can be replaced by a corresponding positive definite kernel $k(\mathbf{x}, \mathbf{x_i})$ wherever it was used [5].

2.3 C-SVM classifier

Because only Support vectors patterns are used to construct the final decision function, one misclassified pattern can change the solution considerably. To allow the possibility of examples violating (7), so-called *slack variables* $\xi_i \geq 0$ can be introduced.

Separation constraints (7) are relaxed to:

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 - \xi_i, \ i = 1, \ldots, m. \qquad (14)$$

Clearly, by making ξ_i large enough, the constraints (14) can always be met. In order not to obtain the trivial solution, where all ξ_i take on large values, we need to penalize them in the objective function. To this end, a term $\sum_i \xi_i$ is included in (8).

An SV classifier based on relaxed primal problem inequality conditions (14) is called *soft margin classifier*. In the simplest case, referred to as the

C-SV classifier, primal optimization problem (8) has following form (with parameter $C > 0$),

$$\underset{\mathbf{w}\in\mathcal{H},\xi\in\mathbb{R}^m}{\text{minimize}} \ \Psi(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i, \qquad (15)$$

subject to the constraints $\xi_i \geq 0, i \in \hat{m}$ and (14).

We can again derive the dual optimization problem.

$$\underset{\alpha\in\mathcal{R}^m}{\text{maximize}} W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \qquad (16)$$

$$\text{subject to } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m}\alpha_i y_i = 0. \qquad (17)$$

Large values of C minimize training error whereas small numbers maximize the margin. However, there is no particular way how to choose the right value for constant C. Therefore a modified classifier *ν-SVM* was proposed. It replaces the parameter C by parameter ν. The latter controls the number of margin errors and support vectors. It appears that the parameter ν is more easily to determine than the parameter C.

3 Fuzzy C-SVM classifier

In this section, we describe how soft-margin classifier can be used for fuzzy classification. SVM classifiers use crisp sets as output classes. In fuzzy classification output classes are modeled as fuzzy sets.

According to the definition of a training set with two output classes, every training pattern belongs either to the first or to the second class. There are none without an assignment. To be consistent with that definition, we model one output class as fuzzy set and the other one as its fuzzy complement. Hence, only one fuzzy set is needed.We call the first fuzzy set *primary* and the other one *complementary* fuzzy class.

Note that the membership of a pattern to complementary output class is dependent on the used fuzzy logic. Unless stated otherwise, we use Lukasiewitz fuzzy complement ($\neg\mu = 1 - \mu$).

Definition 3 (Fuzzy Training Set). *A sequence*

$$(x_1,\mu_1),\ldots,(x_m,\mu_m) \in \mathbb{R}^s \times [0,1] \qquad (18)$$

is called fuzzy training set. *Values μ_i represent membership degree of pattern x_i in the primary fuzzy class.*

SVM classifiers expect y_i values to be ± 1. To be able to use SVM framework, we do the following transformation before we substitute values to SVM equations:

$$y_i = \begin{cases} 1 & \text{for } \mu_i > 0.5 \\ -1 & \text{for } \mu_i < 0.5 \end{cases} \qquad (19)$$

We leave out training examples with membership degree equal to 0.5.

Let us move to the derivation of a transformation function $\mathcal{T}$. The idea is that the transformation function processes the results from a C-SVM classifier and the membership degrees of the patterns are obtained.

When a C-SVM classifier finds the optimal hyperplane for a training set, we can compute for each training pattern $\mathbf{x}_i$ its coordinate with respect to the normal vector of the hyperplane ($d_i = (\langle\mathbf{w},\mathbf{x}_i\rangle + b)/\|\mathbf{w}\|$). We suggest to use these coordinates (*normal coordinates*) as the input to the transformation function $\mathcal{T}$.

Figure (1) depicts a training set in a 2-dimensional feature space. There is a soft margin hyperplane $(\mathbf{w}, b)$ which represents the decision boundary. Patterns x_1 and x_2 are support vectors; $\xi_1 = \xi_2 = 0$. Patterns x_3 and x_4 are within the margin and their membership degrees should be lower than degree of patterns x_1 and x_2 (although they are correctly classified). Pattern x_6 is far away from the margin and its membership degree should be at least as high as the membership degree of support vectors. Patterns x_5 and x_7 are misclassified and their slack variables attain values higher than 1; their membership degree should be very small or zero.

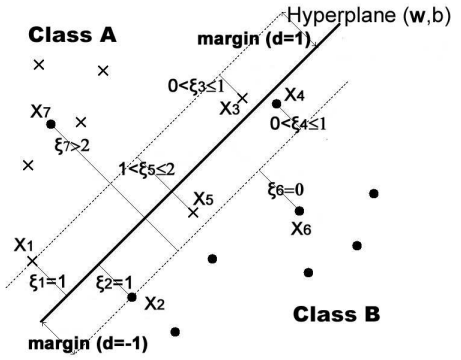


Fig. 1. Geometrical meaning of slack variables.

We suggest that the transformation function ($\mathcal{T} : \mathcal{R} \to [0,1]$) should have following properties:

1. It is a *non-decreasing* function. (The more positive is the normal coordinate of a pattern the higher should be its membership degree in the primary fuzzy set.)
2. Its *domain* are real numbers $\mathbb{R}$. (Classifier has to be able to classify all points. The normal coordinate will be negative for patterns belonging to the complementary fuzzy class.)
3. Its *range* is a subset of the interval $[0,1]$. (The output is the membership degree of a pattern in the primary fuzzy class.)

4. It should be *steeper* in the margin and *flatter* outside.
5. Misclassified patterns should have lower membership degree than correctly classified patterns.
6. In addition, it is reasonable to assume value 1/2 for the patterns laying exactly on the margin.

Typically, *sigmoid* functions have many of these properties. In our research, we use several transformation functions; here a logistic function with one parameter a is recalled for illustration:

$$\mathcal{T}_a(x) = \frac{1}{1+e^{-ax}},\ a > 0 \tag{20}$$

The logistic function has the range $(0, 1)$ and the parameter a determines its steepness.

If there is no misclassification, membership degree should be very similar to the *linear margin function*.

$$f_{LM}(x) = \begin{cases} 1 & \text{for } x > 1 \\ \frac{1+x}{2} & \text{for } x \in [-1, 1] \\ 0 & \text{for } x < -1 \end{cases} \tag{21}$$

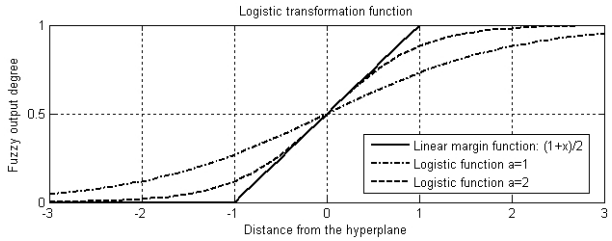


Fig. 2. Logistic transformation function.

When searching the optimal transformation function, we penalize all differences from function f_{LM} (21). According to suggested property 5. of transformation functions we will try to minimize membership degrees of misclassified patterns at the same time.

Let us now formulate the constrained optimization problem for logistic function with the parameter a:

$$\min_{a>0} W(a) = \sum_{i=1}^{m} |\mathcal{T}_a(d_i) - f_{LM}| + \sum_{i=1,\xi_i>1}^{m} y_i(\xi_i - 1)\mathcal{T}_a(d_i), \tag{22}$$

$$\text{subject to: } \mathcal{T}_a(1) \geq \alpha \text{ and } \mathcal{T}_a(-1) \leq 1 - \alpha, \tag{23}$$

where d_i is the normal coordinate of a pattern x_i and ξ_i is its slack variable. The constant $\alpha \in (0.5, 1]$ determines the minimal membership degree of the primary class support vectors. Ideally, it should be equal to 1, but this option would reduce the number of the suitable transformation functions significantly. Therefore, we choose a weaker condition. The logistic function in the optimization problem can be easily replaced by any other transformation function with one or more parameters.

When a new pattern is classified, its normal coordinate is computed, then the transformation function is applied and the membership degree of the pattern is acquired.

4 Conclusion

There are many other possible transformation functions. Presently, we are studying properties of a double sigmoid function. We intend to use existing *measures of quality of classification* [1] to compare our fuzzy C-SVM classifier with other fuzzy classifiers. We want to study how suitable transformation functions depend on a specific classification problem and which objective criteria can be used to compare transformations.

We will test our approach on benchmarks from University of California in Irvine machine learning repository and on real world application data from materials science. The materials science data was also the motivation for the reported research. The objective of the proposed tests is to check the usefulness of this approach.

In this paper, we discussed only fuzzy classification for two output classes. SVM classifiers usually work with 2 output classes too. There are several techniques for extending SVM classifiers for multi-class classification: one versus the rest, pairwise, multi-class objective function [5]. We want to attempt to adjust some of these techniques for multi-class fuzzy classification.

Our implementation of the fuzzy C-SVM classifier will be downloadable in spring 2008 from http://zdena.euweb.cz/fcsvm/index.php.

References

1. Hand D.J., Construction and Assessment of Classification Rules. John Wiley and Sons, Chichester, 1997
2. Ishibuchi H., Nakashima T., Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems. IEEE Trans. SMC-B 29, 601–618
3. Klir G.J., Bo J., Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall, Englewood Cliffs, 1995
4. Nauck D. and R. Kruse, Obtaining Interpretable Fuzzy Classification Rules from Medical Data. Artificial Intelligence in Medicine 16, 1999, 149–169
5. Schölkopf B. and Smola A.J., Learning with Kernels. The MIT Press, Cambridge, Massachusetts, London, England 2002

A lower bound technique for restricted branching programs*

Stanislav Žák

Institute of Computer Science, Academy of Sciences of the Czech Republic
P. O. Box 5, 18207 Prague 8, Czech Republic stan@cs.cas.cz

Abstract. *We attempt to create a new lower bound technique based on a tight observation concerning separation of positive and negative input strings. As the first result this technique gives a superpolynomial lower bound for restricted branching programs computing a special Boolean function.*

1 Introduction

By a branching program (b.p.) P (over inputs of length n) we mean a finite, oriented, acyclic graph with one source (in-degree = 0) where all nodes have out-degree = 2 (so-called branching nodes) or out-degree = 0 (so-called sinks). The branching nodes are labelled by variables x_i, $i = 1, ..., n$, one out-going edge is labelled by 0 and the other by 1, the sinks are labelled by 0 or by 1.

For an input $a = a_1...a_n \in \{0,1\}^n$ by computation on a we mean the sequence $comp(a)$ of nodes starting at the source of P and ending in a sink. In each node with label x_i the next node is pointed by the edge with label a_i.

A special case of b.p. with in-degree = 1 in each node (with exception of the source) is called decision tree.

If $v \in comp(a)$ we say that a reaches v. If a and b reach v and immediately below v they reach different nodes we say that $comp(a)$ and $comp(b)$ diverge in v (or shortly a and b diverge in v).

P computes function f_P which on each $a \in \{0,1\}^n$ outputs the label of the sink reached by a.

We say that P computes in time $t(n)$ if each its computation on any input is of the length at most $t(n)$.

The well-known restriction of b.p.'s are so-called read-once branching programs in which along each computation each variable is tested at most once. Read-once b.p.'s compute in time n, of course.

If $comp(a)$ has a common part with a path in P we say that a follows this path.

By a distribution we mean any mapping D of $\{0,1\}^n$ to the nodes of P with the property that for each a $D(a) \in comp(a)$. The class of the distribution at node v is the set of all a's mapped to v.

Let $v \in P$. We say that T is a tree developed in v according to P iff T simply copies the paths starting at v till the sinks - different paths in P form different branches in T. Repeated tests on the same variable are omitted in T.

By the size of P we mean the number of its nodes. By the complexity of a Boolean function f we mean the size of the minimal b.p. computing f.

It is a well-known fact that superpolynomial lower bound on the size of b.p.'s implies superlogarithmic lower bound for space complexity of Turing machines.

There is a long history of proving superpolynomial lower bounds for restricted b.p.'s, especially for read-once b.p.'s (see [3]). The strike was done by [1] with proof of superpolynomial lower bound for b.p.'s computing in time $n.\log n$. We are searching for superpolynomial lower bound for less restricted b.p.'s computing in time $n.(\log n)^2$. In this text we present a technique based on a new principle as follows.

Let us have a b.p. P which computes a function f. Let $a = a_1...a_n$, $b = b_1...b_n$, $x = x_1...x_n$ be three inputs. Let for all $i = 1...n$ $x_i = a_i$ or $x_i = b_i$ and $f(a) = f(b) \neq f(x)$.

We see that there is at least one node in P reached by all three a, b, x - e.g. the source of P. On the other hand $f(a), f(b)$ differ from $f(x)$ - therefore in P there is a non-sink node v which is the last node in P reached by all three a, b, x.

Let us consider the situation in v. In v (according to definition of v) a, b, x must diverge. Let the i-th variable be tested in v. It must be $a_i \neq b_i$ otherwise $a_i = b_i = x_i$ and no divergence is possible.

Wlog we may assume that a follows 0-edge outgoing from v, b follows 1-edge and x follows 1-edge , too. Since v is "last", the computational paths starting in v and given by a and b must not meet until the moment when b, x diverge. This is our new principle.

For its application it remains to imagine that in one node many inputs with rich mutual relations can be distributed with the effects that many paths till some significant depths must not have any common nodes. As a consequence we can obtain lower bounds.

We develop a new proof technique based on this principle. In this text we prove a superpolynomial lower bound for branching programs computing in time $n.(\log n)^2$ using some additional assumptions

* Research partially supported by the "Information Society" project 1ET100300517 and the Institutional Research Plan AV0Z10300504.

taken from life. The next research will have to remove them.

2 Multisyms

For appropriate n's we understand the binary inputs of length n as a matrices $m \times k$ where $m.k = n$. We say that such a matrix is a t-multisym if for each choice of t columns there is a row r monochromatic on them (we say that r covers this choice of columns) . For the purposes of this text we shall use only 2-multisyms, simply multisyms. This function was used to prove superpolynomial lower bound for so-called read-once branching programs (see [2]).

We often use notation $m = \epsilon(n).\log n$ and $k = \frac{n}{\epsilon(n).\log n}$. It is easy to see that for $\epsilon(n) \geq 3$ the number of multisyms is at least 2^{n-1}. Indeed the number of multisyms is at least

$2^n - \binom{k}{2}.2^m.2^{n-2m} \geq$
$2^n - n^2.2^{n-m} \geq$
$2^n - 2^{n+2.\log n - \epsilon(n).\log n} \geq$
$2^n - 2^{n-\log n} \geq 2^{n-1}$.

By a canonical branching programs computing multisyms we mean any branching program P consisting from a chain of subprograms $(P_{i,j})$ for $i, j = 1...k$, $i \neq j$. Each program $P_{i,j}$ is responsible for verifying the covering of the pair of columns (C_i, C_j). Each $P_{i,j}$ has two sinks - simply to separate the matrices with covered (C_1, C_2) from the others. The first sink of $P_{i,j}$ is the source of the next subprogram $P_{i',j'}$, the second sink of $P_{i,j}$ is one of sinks of P. Such a $P_{i,j}$ is a chain of microprograms M_r for each row r. M_r is responsible for covering of (C_1, C_2) on row r. (M_r tests equality of two bits.)

Let P be a branching program computing multisyms. Let v be a node of P, a, b be multisyms, x be a nonmultisym, C_1, C_2 be columns, all a, b, x reach v. By $R(v, a, C_1, C_2, b, x)$ we mean that

i) v is the last node reached by all three a, b, x,

ii) in v x follows a,

iii) the test in v is in C_1, the test in which x leaves a for the first time (below v) is in C_2.

We say that P is a reasonable branching program iff P satisfies the next restrictions for all a, C_1, C_2:

R1 For each a and for each C_1, C_2 there is at least one v such that there is b, x satisfying $R(v, a, C_1, C_2, b, x)$ or $R(v, a, C_2, C_1, b, x)$.

R2 Let $R(v, a, C_1, C_2, b, x)$ and $b' \in v$ and $test_v b' \neq test_v a$ (= b' follows the other edge). Then $R(v, a, C_1, C_2, b', x)$.

The next lemma demonstrates that R1, R2 are taken from life.

Lemma 1. *Each canonical branching program (computing multisyms) is reasonable.*

Proof. Let P be a canonical branching program computing multisyms. Let us verify R1.

Let a be a multisym, let C_1, C_2 be a pair of columns. Let us take the subprogram $P_{1,2}$ of P corresponding to the pair C_1, C_2 and within it the microprogram M_r (responsible for a row r) in which a leaves $P_{1,2}$. Let v be the input node of M_r. We want to prove $R(v, a, C_1, C_2, b, x)$ for some b, x. Let x be a nonmultisym which at v follows a and which ends at sink of $P_{1,2}$. (Such an x exists: e.g. x with the following properties - x does not cover C_1, C_2, on M_r x follows and then diverges a, outside of C_1, C_2 x equals 1, on both C_1, C_2 x has at least one 1).

Let b be a multisym which at v follows the opposite edge. Clearly $R(v, a, C_1, C_2, b, x)$.

R1 is verified.

R2 is satisfied clearly. □

The restriction "reasonable b.p." may seem very special since it is formulated in terms of multisyms and therefore it can be reasonably used only for b.p.'s computing this function. On the other hand multisyms seem to be a very appropriate candidate for a function superpolynomially difficult for general b.p.'s computing in time $n(\log n)^2$. We plan in the next development of our ideas to remove both R1, R2. Hence our approach to start with R1,R2 is legitimate. As the main goal in this paper we demonstrate the first effective use of the principle for lower bounds described in Introduction.

3 Combinatorics

The following lemma is formulated purely combinatorially without any connection with branching programs. We will use it in the next section.

Lemma 2. *Let T be a sequence of places, let $|T|$ be its length. Let k be a natural number.*

Let $\binom{k}{2}$ pebbles be distributed on places of T, at most $k-1$ pebbles on one place.

Let u, α be numbers, $u < \frac{\binom{k}{2}}{|T|}$ and $u < k-1$.

Then in T there is a subsequence S of α places such that

(i) on each of them more than u pebbles are distributed,

(ii) between each two places neighboring in S there is at most $d = \alpha.\frac{|T|-o}{o-1}$ places in T where $o = \frac{\binom{k}{2}-|T|.u}{k-1-u}$.

Proof. By a marked place we mean any place with at least $u+1$ pebbles. In T there is at least $o = \frac{\binom{k}{2}-|T|.u}{k-1-u}$

marked places. The average distance between marked places is at most p; $p = \frac{|T|-o}{o-1}$.

The number of intervals of length at least $\alpha.p+1$ of non-marked places is at most $\frac{|T|-o}{\alpha.p+1} = \frac{|T|-o}{\alpha.\frac{|T|-o}{o-1}+1} < \frac{o}{\alpha}$.

Hence there are at most $\frac{o}{\alpha}$ subsequences of marked places in which each two neighbors have distance at most $\alpha.p$ and moreover there is at least one such subsequence which contains at least α places. □

Corollary 1. *For* $k = \frac{n}{\epsilon(n).\log n}$, $\epsilon(n) \leq \log n$, $T(n) = n.(\log n)^2$, $\alpha = (\log n)^2$ *and* $u = \frac{\sqrt{(n)}}{(\log n)^2}$, $d = d(n)$ *is at most* $6.(\log n)^6$.

Proof. $d(n) = \alpha(n).\frac{T(n)-o}{o-1} =$
$\frac{\alpha(n).T(n)}{o-1} - \frac{\alpha(n).o}{o-1} \leq$
$\frac{n.(\log n)^4}{o-1}$ (since $o > 1$)
$= \frac{n.(\log n)^4}{\frac{\binom{k}{2}-T(n).u}{k-1-u}-1} \leq$
$\frac{n.(\log n)^4.(k-1-u)}{\binom{k}{2}-T(n).u-k+1+u} \leq$
$\frac{n.(\log n)^4.k}{\frac{1}{2}.\binom{k}{2}} \leq 6.(\log n)^6$. □

4 Lower bound

Definition 1. *Let B be a full binary decision tree. Let M be a set of inputs, $M \subseteq \{0,1\}^n$. Let T be a subtree of B. Let L_T be a number of all leaves of T and $L_T^{M,B}$ be the number of leaves of T reached (in B) by inputs from M. By (M,B)-ratio of T $p_T^{M,B}$ we mean the number $\frac{L_T^{M,B}}{L_T}$.*

Theorem 1. *The polynomially sized reasonable branching programs cannot compute multisyms with $3 \leq \epsilon(n) \leq \log n$ in time $n.(\log n)^2$.*

Proof. By contradiction. Let P be a reasonable branching program computing multisyms in time $n.(\log n)^2$ and with $|P| \leq n^q$ for some q.

For each multisym a we understand the nodes of $comp(a)$ as a sequence of places. Let $v \in comp(a)$ be a node (place) and let C_1 be the column of the variable tested in v. The number of pebbles on v is given by the number of columns C such that $R(v,a,C_1,C,b,x)$ for some b's, x's. Hence on each v there is at most $k-1$ pebbles where k is the number of columns in the matrices (multisyms).

From R1 it follows that on $comp(a)$ there must be at least $\binom{k}{2}$ pebbles. Therefore according to Lemma, for each a , in $comp(a)$ there is a relatively long $((\log n)^2)$ subsequence S_a of nodes with relatively many $(\frac{\sqrt{(n)}}{(\log n)^2})$ pebbles and with the distances between their nodes relatively small (at most $6.(\log n)^6$ - see Corollary 1).

Let us distribute each multisym a to the first node of S_a. Let v be the node with a maximal class M of this distribution. It is easy to see that (for $\epsilon(n) \geq 3$) the number of multisyms is at least 2^{n-1}. Hence $|M| \geq \frac{2^{n-1}}{n^q}$.

From v we develop the syntactic tree B' according to P ("syntactic" means that we take into account also the branches which are not followed by any input e.g. in case of repeated tests on the same variable) till the depth $(\log n)^2.6.(\log n)^6$ in P. Below this depth we continue the developing of the tree ignoring the repeated tests. On each branch b at the node corresponding to the related sink of P we add some subtree S_b which on each their branches tests all variables not tested below v till now. We know that the length of branches (constructed till now) is at least n and that each such branch is followed by at most one input (of length n). The branches (constructed till now) can be of different length (due to the possible different number of repeated tests along different branches). To obtain a full tree B' of certain length we add some full tree T_b of appropriate length to each branch b shorter that the longest one of branches constructed till now. As a result we obtain a full binary decision tree B' of depth $n+\delta$. δ is at most $(\log n)^2.6.(\log n)^6$.

We modify B' as follows. In the middle of each edge below v till the depth $(\log n)^2.6.(\log n)^6$ we insert a new node with the same test as in the node from which the edge in question out-goes. On the dead edge of this test we add a dummy full binary tree D of depth $\log((\log n)^2.6.(\log n)^6) + (\log n)^2 + \delta$. Below the depth $(\log n)^2.6.(\log n)^6$ we add dummy subtrees of depth $\delta - l$ where l is level of B' below $(\log n)^2.6.(\log n)^6$. Maximal l for this operation is δ.

Let B be the resulting tree. The number of its leaves is at most
$2^{n+\delta} +$
$+2^{6.(\log n)^8}.2^{\log((\log n)^2.6.(\log n)^6)+(\log n)^2+\delta} +$
$+\sum_{l=0}^{\delta} 2^l.2^{\delta-l} \leq$
$\leq 2^{n+\delta+1}$.
It follows that M-ratio of B
$p_B^M \geq \frac{|M|}{2^{n+\delta+1}}$.

Lemma 3. *There is a full binary decision tree T of depth $(\log n)^2 + \delta$ rooted in v such that*

a) Each its branch (considered as a sequence of nodes) is a subsequence of a branch of B.

b) $p_T^{M,B} \geq p_B^M$.

c) The nodes of T reached by inputs from M in depth $(\log n)^2$ are pairwise different in P.

d) The number of leaves of T reached by inputs from M is the same as the number of nodes of T reached by inputs from M on level $(\log n)^2$.

We continue in the proof of Theorem 1. Since P is small ($\leq n^q$) from c)and d) it follows that the number of leaves of T reached by inputs from M is small ($\leq n^q$) which implies that $p_T^{M,B} \leq \frac{n^q}{2^{(\log n)^2+\delta}}$. According to b) it follows that $\frac{n^q}{2^{(\log n)^2+\delta}} \geq p_T^{M,B} \geq p_B^M \geq \frac{|M|}{2^{n+\delta+1}}$. A contradiction. Q.E.D. □

Proof of Lemma 3.

The main point of construction of the desired tree T is recursive use of procedure *Proc*.

In the first step of *Proc* we take into account the both subtrees of B rooted by the immediate successors v_0, v_1 of v. If one of these subtrees is reached by no input from M we add it to the constructed tree T'. We know that in the other case in P the branches starting in v_0 and in v_1 and followed by inputs from M don't meet till the depth $\frac{\sqrt{n}}{(\log n)^2}$. This follows from the restriction R2. (Cf. c) of Lemma.)

For $i = 1, 2$ below v_i we distribute each input a from M to the node of B where a has the next (second) node of its special subsequence S_a. First let us take into account the easy case when on each branch there is at most one node of this distribution. Among the full subtrees with roots in the nodes of the distribution we chose that one which has maximal M-ratio. We notice that this ratio is at least the same or larger than the ratio of the tree rooted by corresponding v_i. In T' there will be an edge from v to the root of the chosen tree. On each tree chosen in the present iteration of *Proc* we apply the next iteration.

The difficult case is when some nodes of this distribution are on the same branch of B. Let leaders be nodes (of this distribution) which have no such nodes as predecessors.

We take into account a partition of the set of nodes of the distribution according to the equivalence "to be below (or equal to) the same leader".

Let w_i's be a class of nodes (a class of the partition) where sets of multisyms M_i's are distributed. We construct the corresponding trees R_i's rooted in w_i's. Each R_i contains all branches followed by inputs from M_i. The sets of leaves of R_i's are pairwise disjoint (since in each leaf is at most one input - from the construction of B). In general the union of R_i's do not cover the whole subtree rooted in the leader of w_i's because in general there are branches not followed by any input from M. To each R_i we potentially add some other branches to save the possibility to continue the construction of full tree of depth $(\log n)^2 + \delta$ in case when the modified R_i is chosen as the tree with maximal M-ratio. We proceed according to the following rules:

Let us take R one of R_i's. In B let us follow its branches from its root to its leaves. Let u be a node on some branch b such that only one outgoing edge is followed by inputs from M_i. In case when u is in B in depth at least $d = (\log n)^2.6.(\log n)^6 + \delta$ we add the whole subtree of B rooted by the out-going edge in question to R or we do nothing if u plays its role for another R which consumes this subtree in question.

In case when u is in the depth at most d the most difficult case is such that the out-going edge in question is followed by inputs from some other M_i's.

We saturate the need of subtree in the direction of the out-going edge in question using the added subtree rooted in the middle of this edge. The added subtree is large enough to yield subtrees of desired depth $(\log n)^2 + \delta$ for at most $6.(\log n)^6$ R_i's in at most $(\log n)^2$ iterations of *Proc*.

From T' we construct the desired full binary tree T of depth $(\log n)^2 + \delta$ as follows:

From each node of T' in depth $(\log n)^2$ in T' we follow the branches in B. If the node in question (on level $(\log n)^2$ and below) of T' is not reached in B by any input from M we simply add a subtree of an appropriate depth to gain the desired depth $(\log n)^2+\delta$. (In case of back reconstruction of B from T b) of Lemma is not corrupted.)

If a node has only one out-going edge followed by inputs from M we prolong the dead edge by an subtree of appropriate depth to gain the desired depth $(log\ n)^2+\delta$ and we follow the edge with inputs from M.

If in a node both outgoing edges are followed by inputs from M we consider two subtrees rooted in this node. Each of these two subtrees contains the subtree rooted by one outgoing edge and the dummy tree rooted by the middle of the opposite out-going edge.

We chose that one with maximal number of inputs from M. In both cases the back reconstruction of B does not corrupt b) of Lemma.

(From construction of B it follows that the operation above is ensured till the depth $(\log n)^2.6.(\log n)^6 + \delta$. This is sufficient for construction of T.)

The conditions a), b), c), d) of Lemma are satisfied.

It remains to verify that T is a full tree of depth $(\log n)^2 + \delta$.

From the description of *Proc* it is easy to see that T is a full tree till the level (in T) at least $(\log n)^2$. On this level the nodes not reached (in B) by any input from M have an appropriate prolongation by a subtree till the desired depth $(\log n)^2+\delta$ - this follows from the construction of B and from the description of *Proc*. □

The proof above is the first application of the principle mentioned in Introduction. The result is modest but reasonable. The challenge is to remove restrictions R1, R2 and in this way to achieve a considerable improvement of Ajtai's result [1].

References

1. Ajtai M., A Non-Linear Time Lower Bound for Boolean Branching Programs. Proc. of 40th IEEE Ann. Symp. on Foundations of Computer Science, 1999, 60–70
2. Jukna S., Žák S., Some Notes on the Information Flow in Read-Once Branching Programs. In Proc. of 27th Ann. Conf. on Current trends in Theory and Practice of Informatics, LNCS 1963, Springer, Berlin, 2000, 356–364
3. Wegener I., Branching Programs and Binary Decision Diagrams. SIAM, 2000
4. Žák S., A Lower Bound Technique for Restricted Branching Programs (version G). Institute of Computer Science, Academy of Sciences, Prague, Technical report No. 991, March 2007