Předmluva

Osmý workshop **ITAT'08 – Informační Technologie – Applikace a Teorie** se konal v Horském hotelu Hrebienok, (`http://www.sorea.sk/Default.aspx?CatID=24&hid=4/`) ve výšce 1280 metrů nad mořem ve Vysokých Tatrách (`http://www.tatry.sk/`), Slovensko, koncem září 2008.

Workshop ITAT (`http://ics.upjs.sk/itat`) je místem setkání lidí z bývalého Československa pracujících v informatice (oficiálním jazykem pro referáty jsou čeština, slovenština a polština. Letos z workshopu vydáváme dvě publikace – konferenční sborník s původními vědeckými pracemi v anglickém jazyce (8 příspěvků) a tento sborník.

Celkově bylo zasláno 41 příspěvků. Tento sborník obsahuje

- 3 původní vědecké práce
- 23 prací typu "Work in progress" a
- 6 posterů

Všechny práce byly recenzovány nejméně dvěma nezávislými recenzenty.

Tematicky je workshop široce zaměřen, zahrnuje všechny oblasti informatiky od základů a bezpečnosti, přes data a semantický web až po softwareové inženýrství. Důraz je kladen více na výměnu informací mezi účastníky než na selektivitu publikací. Workshop poskytuje první příležitost pro studenty k veřejné prezentaci a k diskuzi se "staršími". Velký prostor je vymezen pro neformální diskuze. Místo konání je tradičně alespoň 1000 m.n.m a mimo přímý dosah veřejné dopravy.

Workshop organizovali
Ústav Informatiky Univerzity P.J. Šafárika, Košice;
Ústav informatiky AV ČR vvi. Praha
Sekce Informatiky MFF UK Praha a
Slovenská Spoločnosť pre Umelú Inteligenciu

Částečná podpora byla poskytnuta z projektů programu Informační Společnost Tematického Programu II Národního Programu Výzkumu ČR 1ET100300419 a 1ET100300517.

Peter Vojtáš

Program Committee

Peter Vojtáš, (chair), *Charles University, Prague, CZ*
Gabriela Andrejková, *University of P.J. Šafárik, Košice, SK*
Mária Bieliková, *Slovak University of Technology in Bratislava, SK*
Leo Galamboš, *Charles University, Prague, CZ*
Ladislav Hluchý, *Slovak Academy of Sciences, Bratislava, SK*
Tomáš Horváth, *University of P.J. Šafárik, Košice, SK*
Karel Ježek, *The University of West Bohemia, Plzeň, CZ*
Jozef Jirásek, *University of P.J. Šafárik, Košice, SK*
Jana Kohoutková, *Masaryk University, Brno, CZ*
Stanislav Krajči, *University of P.J. Šafárik, Košice, SK*
Věra Kůrková, *Institute of Computer Science, AS CR, Prague, CZ*
Markéta Lopatková, *Charles University, Prague, CZ*
Ján Paralič, *Technical University in Košice, SK*
Dana Pardubská, *Comenius University, Bratislava, SK*
Martin Plátek, *Charles University, Prague, CZ*
Jaroslav Pokorný, *Charles University, Prague, CZ*
Karel Richta, *Czech Technical University, Prague, CZ*
Gabriel Semanišin, *University of P.J. Šafárik, Košice, SK*
Václav Snášel, *Technical University VŠB in Ostrava, CZ*
Vojtěch Svátek, *University of Economics in Prague, CZ*
Jiří Šíma, *Institute of Computer Science, AS CR, Prague, CZ*
Július Štuller, *Institute of Computer Science, AS CR, Prague, CZ*
Jakub Yaghob, *Charles University, Prague, CZ*
Filip Zavoral, *Charles University, Prague, CZ*
Stanislav Žák, *Institute of Computer Science, AS CR, Prague, CZ*
Filip Železný, *Czech Technical University, Prague, CZ*

Organizing Committee

Tomáš Horváth, (chair), *University of P.J. Šafárik, Košice, SK*
Hanka Bílková, *Institute of Computer Science, AS CR, Prague, CZ*
Peter Gurský, *University of P.J. Šafárik, Košice, SK*
Róbert Novotný, *University of P.J. Šafárik, Košice, SK*
Jana Pribolová, *University of P.J. Šafárik, Košice, SK*
Veronika Vaneková, *University of P.J. Šafárik, Košice, SK*

Organization

ITAT 2008 Information Technologies – Applications and Theory was organized by
University of P.J. Šafárik, Košice, SK
Institute of Computer Science, AS CR, Prague, CZ
Faculty of Mathematics and Physics, Charles University, Prague, CZ
Slovak Society for Artificial Intelligence, SK

Table of Contents

ITAT'08

Information Technology – Applications and Theory

SCIENTIFIC PAPERS

Evoluce chování agentů v 3D prostředí

Jakub Gemrot[1], Rudolf Kadlec[1], Cyril Brom[1], and Petra Vidnerová[2]

[1] Matematicko-Fyzikální fakulta, UK
jakub.gemrot@gmail.com, rudolf.kadlec@gmail.com, brom@ksvi.mff.cuni.cz
WWW : http://artemis.ms.mff.cuni.cz/pogamut
[2] Ústav Informatiky, AV ČR
petra@cs.cas.cz

Abstrakt *Článek se zabývá evolučním přístupem k optimalizaci pohybu a vysokoúrovňového rozhodování virtuální postavy v prostředí komerční hry Unreal Tournament 2004 (UT04). Představíme infrastrukturu pro optimalizaci chování vyhýbání se překážkám pomocí evolučních algoritmů (EA) a funkcionální architekturu, která je vhodná pro optimalizaci vysoko-úrovňového rozhodování pomocí genetického programování (GP). Možnosti obou přístupů jsou demonstrovány na dvou experimentech.*

1 Úvod

Svět počítačových her prochází v posledních letech rapidním vývojem. Hry jsou stále sofistikovanější, realističtější a odehrávají se zpravidla v dynamických a komplexních 3D světech. Na druhou stranu inteligence počítačem řízených postav, tzv. *botů*, se zlepšuje mnohem pomaleji. Tito boti splňují kritéria agentů tak, jak je popsal Wooldridge [13], i když jejich chování je často naskriptované a pevně spjaté s konkrétní prostředím. Takto naskriptovaný bot pak zpravidla nedokáže proaktivně sledovat cíl v jiných prostředích. Z tohoto pohledu představují moderní počítačové hry zajímavou výzvu pro výzkum umělé inteligence. Cílem tohoto článku je představit problém vývoje chování botů, jejich parametrizaci a námi zvolené řešení.

Vývoj a parametrizaci chování botů řešíme pomocí prostředků genetických algoritmů a genetického programování [6]. Pokud budeme na chování bota nahlížet jako na vrstevnatou architekturu, můžeme rozlišit dvě hlavní vrstvy. Vrstvu nižší úrovně, která se stará především o navigaci bota v prostředí a vrstvu vyšší úrovně, která plánuje provádění akcí. Navigace prostředím zajišťuje bezpečný pohyb bota ve světě a pracuje nad akcemi nízké abstrakce, např. udělej krok, otoč se doprava. Plánování akcí bota sleduje vždy určitý cíl a provádí rozhodnutí typu, kam jít a co tam udělat. Při plánování se již počítá s akcemi vyšší abstrakce, např. jdi do Fredova domu. Tyto dvě vrstvy chování vyvíjíme odděleně a postupně. V naší práci se zatím nezabýváme dalšími aspekty botů, jako jsou reprezentace znalostí či sociální chování.

Navigace prostředím je optimalizována pomocí klasických genetických algoritmů s fixní délkou chromozomu. Naproti tomu pro optimalizaci vysokoúrovňového rozhodování jsme použili genetické programování s proměnlivou délkou chromozomu.

V následující kapitole se seznámíme se specifiky zvoleného prostředí hry Unreal Tournament 2004, poté popíšeme příbuzné práce. Ve čtvrté kapitole budou představeny softwarové nástroje potřebné pro provádění našich experimentů. V páté kapitole představíme naše modely a výsledky jejich testování. Na závěr provedeme diskusi možného budoucího vývoje.

2 Boti a prostředí hry Unreal Tournament 2004

Zvoleným 3D prostředím je svět ze hry Unreal Tournament 2004 (UT04), který patří do kategorie first person shooters (FPS) her. Hráč vidí svět z pohledu očí svého avatara (first person), může jím procházet, sbírat předměty a pomocí střelných zbraní bojovat s ostatními avatary ve hře (shooter). Kromě avatarů hráčů se ve světě mohou nacházet ještě avataři kontrolováni počítačem (boti), kteří představují umělé protivníky. Hra UT04 pak nabízí několik typů her, které se liší způsobem, jakým hráči (i boti) dostávají vítězné body. V základním typu hry *Death match* je hráč odměňován pouze za zneškodňování nepřátel. Další typy her jako *Capture the flag* nebo *Domination point* pak odměňují hráče za kradení vlajek nepříteli, resp. za dobití a udržení určitého území ve světě. V tomto článku se zabýváme vývojem chování pouze pro první typ hry *Death match*.

Důležitým prvkem UT04 je způsob reprezentace prostředí pro potřeby botů. Vzhledem k výpočetní náročnosti algoritmů analýzy 3D prostředí musí být toto prostředí pro boty předzpracováno a reprezentováno tak, aby v něm byla možná rychlá navigace. Pro tento účel se v UT04 reprezentuje prostředí jako orientovaný graf navigačních bodů. Místa, která jsou pro bota dosažitelná chůzí či skákáním v prostředí, jsou rovnoměrně pokryta navigačními body. Dva navigační body jsou pak spojeny hranou tehdy, je-li možné se dostat přímým pohybem od jednoho k druhému. Během pohybu po této hraně je občas nutné vyhnout se drobným překážkám, např. rohu budovy, což řešíme odděleným vývojem nižší vrstvy chování bota, která se stará o navigaci prostředním. Součástí navigačního grafu jsou též veškeré předměty, které se ve světě nachází. UT04 navíc poskytuje algoritmus na hledání nejkratší cesty v tomto grafu.

Každý bot v UT04 je reprezentován: 1) svým humanoidním tělem, 2) množinou senzorů, kterými prostředí vnímá, 3) množinou akcí, které může vykonávat, 4) mechanismem pro výběr následující akce. Bot má také svůj cíl, sbírat vítězné body, jejichž význam je závislý na typu hry - v našem případě je získává za zneškodňování nepřátel.

Senzory dále můžeme rozdělit na interní a externí. Interní senzory poskytují botovi informace o jeho těle a inventáři. Jsou to: procento zdraví (při poklesu na 0 bot umírá), množství brnění (snižuje účinek nepřátelských zbraní), pozice v prostředí a aktuální rotace, aktuálně zvolená zbraň a seznam všech zbraní spolu s počtem jejich nábojů, které bot má.

Externí senzory poskytují botovi informace o konfiguraci okolního prostředí. Jsou to: seznam viditelných hráčů, navigační graf, seznam míst, kde se nachází předměty (zbraně, náboje, brnění a lékárny), informace z raycastingu [11] a množina asynchronních události (bot zemřel, bot uslyšel zvuk, přichází střela, atd.).

Akce jsou: jdi dopředu, zastav se, vyskoč, pliž se, najdi cestu v navigačním grafu, jdi do strany, otoč se o daný úhel, vystřel na určené místo, změn zbraň.

3 Příbuzné práce

Prakticky všechny příklady genetické optimalizace chování postav v prostředí komerčních her spadají do období posledních deseti let. Zaměříme-li se na doménu FPS her, můžeme najít dvě hlavní skupiny takovýchto modelů.

První dovolují genetickou optimalizaci téměř všech aspektů botova chování (pohyb, výběr zbraně, rozhodnutí zda zaútočit či ne, atd.) [3,9]. Tyto modely jsou většinou inspirovány evoluční robotikou a většinou nevyužívají veškerou informaci poskytovanou herním prostředím. Náš přístup naopak využívá všech dostupných informací, které zvolené herní prostředí UT04 nabízí (např. navigační graf).

Druhá skupina používá genetiku pro optimalizaci vybraného, člověkem většinou obtížně parametrizovatelného, chování v jinak z většiny předprogramovaném algoritmu botova rozhodování. Nejčastěji je optimalizováno chování pro bojové situace [12,2,4]. Výhodou těchto modelů je, že využívají symbolickou informaci poskytovanou herním prostředím.

4 Softwarové nástroje

I přesto, že hra UT04 poskytuje vlastní scriptovací jazyk UnrealScript, pro připojení našich botů k hernímu prostředí jsme použili platformu Pogamut [5]. Pogamut umožňuje mimo jiné psaní logiky botů v jazyce Java, pro který existuje široká nabídka volně šiřitelných knihoven. To je nesporná výhoda oproti použití UnrealScriptu, který takovouto podporu nemá. Platforma Pogamut je již několik let vyvíjena na našem pracovišti[3].

Pogamut zjednodušuje připojení botů do herního prostředí UT04 a poskytuje množství sensorických a motorických primitiv, manažer žurnálů a přídavný modul pro IDE Netbeans™. Platforma je založena na dobře známém protokolu GameBots [1], který rozšiřuje o mnoho nových možností a zpřístupňuje jej pro jazyk Java. Platforma Pogamut je volně šiřitelná pro nekomerční a nevojenské účely a může být stažena z našich webových stránek[4].

UT04 je realtimové prostředí, a proto samo o sobě není příliš vhodné jako simulátor pro použití genetických algoritmů. Rychlost běhu času sice může být zvýšena, ale hra neobsahuje možnost ”běž, jak nejrychleji je to možné”. Změna rychlosti simulace může navíc výrazně ovlivnit fyzikální simulátor a tím i výsledky experimentu.

Abychom obešli tuto nevýhodu prostředí UT04, vyvinuli jsme rozšíření platformy Pogamut, které nazýváme Pogamut GRID. Pogamut GRID umožňuje spouštění experimentů na clusteru počítačů a tím mnohonásobně zkracuje čas potřebný pro běh evoluce.

5 Modely pro vývoj a parametrizaci chování botů

Na chování botů nahlížíme jako na vrstevnatou architekturu. V této kapitole představujeme dvě vrstvy. Nejprve vyvíjíme nižší vrstvu, která je zodpovědná za navigaci v UT04, pomocí genetického algoritmu. Následně ve vyšší vrstvě vyvíjíme mechanismus pro výběr akcí pomocí genetického programování.

5.1 Nižší vrstva chování a vyhýbání se překážkám

Jak již bylo řečeno, nižší vrstva chování se stará o navigaci v prostředí. Tato navigace se opírá o navigační graf, který je poskytován UT04 a který také poskytuje implementaci algoritmu pro hledání cesty v tomto grafu. Jakmile je cesta v grafu navigačních bodů známá, zbývá jen vyřešit problém pohybu mezi jednotlivými body. Během tohoto pohybu je občas nutné se vyhnout menším překážkám (angl. obstacle avoidance) [10]. Následující kapitola představuje námi navržený model pro vyhýbání se překážkám, který je parametrizován pomocí genetického algoritmu [7].

Jediným způsobem v UT04, jak zjistit přítomnost překážek v některém ze směrů od aktuální pozice bota, je použití raycastingu. Raycasting zde má funkci podobnou infračerveným senzorům robota Khepery [8], které robot používá na detekci překážek kolem něj. Tedy náš model se sestává z několika takovýchto paprsků. Pokud paprsek narazí do překážky, tak vygeneruje sílu, která vychýlí bota

[3] AMIS, http://artemis.ms.mff.cuni.cz/

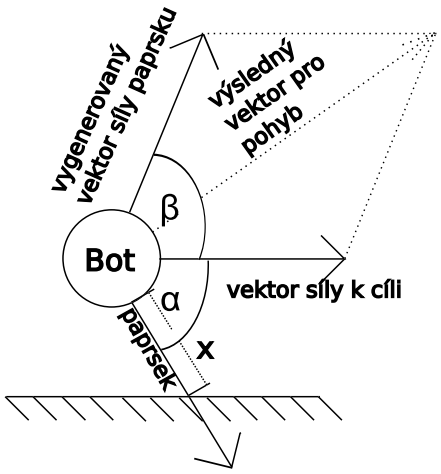
[4] Pogamut, http://artemis.ms.mff.cuni.cz/pogamut/

z jeho aktuálního pohybu. Problémem je, kolik těchto paprsků má model obsahovat, jakým směrem se mají tyto paprsky vrhat a do jaké vzdálenosti a jakou sílu (směr a velikost) mají generovat. Počet paprsků byl experimentálně stanoven na tři. Ostatní parametry byly stanoveny pomocí genetického algoritmu.

Model pracuje tak, že na bota vždy působí konstantní síla směrem k cíli. Bot s frekvencí 5hz používá raycastingu k detekci překážek. Narazí-li některý z paprsků na překážku, vygeneruje dodatečnou sílu, která na bota působí následujících 200ms a odchyluje ho od přímého pohybu k cíli. Směr této síly je stanoven pevně svým úhlem. Velikost vygenerované síly je kvadratickou funkcí $f(x) = Ax^2 + Bx$ vzdálenosti překážky od bota, viz obr. 1. I když je použit termín síly, tak bot není chápán jako hmotný bot a nepočítá se jeho setrvačnost. Bot běží vždy směrem výsledné síly maximální možnou rychlostí avatara v UT04.

Gen jedince tedy obsahuje těchto 5 parametrů (reálných čísel):

1. úhel paprsku α
2. vzdálenost, do které je paprsek vrhán
3. úhel síly β, kterou paprsek generuje
4. parametry funkce A a B



Obr. 1. Pohled na bota shora, na kterém je zobrazen jeden paprsek generující sílu, která bota odpuzuje od stěny, stejně se pak skládá výsledná síla pro více sil. Hodnota *x* je použita jako vstup funkce *f*, která udává velikost vektoru síly generovaného paprskem. Vektor síly k cíli a vektor síly paprsku se vektorově sečtou a určí výsledný směr botova pohybu.

Pro potřeby evoluce parametrů jsme vytvořili následující prostředí (obr. 2) obsahující dva navigační body, mezi kterými se nachází množství překážek, kterým se však dá jednoduše vyhnout úkrokem doleva či doprava. Bot se vždy objeví na bodu Start a má 15 sekund na dosažení bodu Cíl. Člověkem ovládaný avatar dokáže projít trasu asi za 5 sekund.

Obr. 2. Prostředí pro evoluci parametrů modelu vyhýbání se překážkám, pohled shora.

Fitness jedince (bota) je pak stanovena podle těchto kritérií:

1. vzdálenost, jak daleko se bot dostal na cestě od startu k cíli — bonus za vzdálenost s
2. jakou průměrnou rychlostí se bot pohyboval — bonus za plynulý pohyb v
3. dosažení cíle v časovém limitu —- zbylý čas připočten jako bonus t
4. zda bot narážel do zdí — postih $a(z)$, kdy z je počet naražení do zdí, $a(0) = 1, z \geq 1 : a(z) = 1 + \log_{10}(z)$

Označme maximální průměrnou rychlost bota v_m, vzdálenost mezi startem a cílem s_m a maximální čas t_m. Výsledná fitness (účelová funkce) je pak

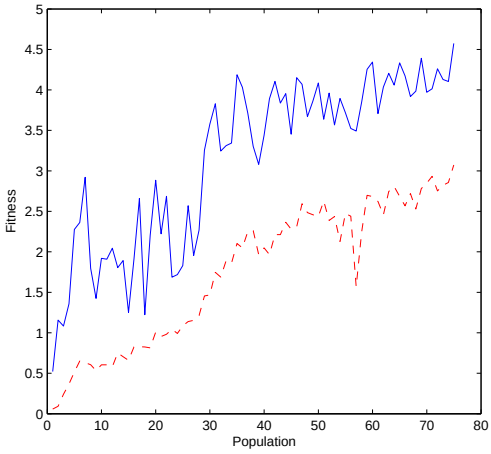
$$fitness = \frac{\frac{v}{v_m} + 3(sin(\frac{s}{s_m} * \frac{\pi}{2})) + 5\frac{t_m - t}{t_m}}{a(z)}.$$

Tento vzorec je výsledkem pozorování jednotlivých běhů genetického algoritmu. Vynecháme-li například bonus za plynulý pohyb, měli boti tendenci do překážky před sebou narazit a pak po ní klouzat dokud se nedostali za její roh. Naopak, když maximální bonus za dosaženou vzdálenost a průměrnou rychlost byl stejný, tak měl genetický algoritmus tendenci uváznout v lokálním maximu, kdy boti běhali co nejrychleji v rohu u stěny druhé překážky. Funkce sinus v bonusu za dosaženou vzdálenost dopomáhá tomu, aby evoluce nejdříve našla boty, kteří dosáhnou cíle a teprve potom aby byla optimalizována rychlost jejich pohybu.
Další parametry genetického algoritmu:

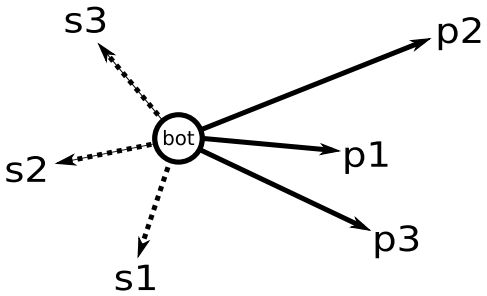
1. selekce: ruleta
2. pravděpodobnost mutace: 10% pro jedince, 20% že u mutovaného jedince zmutují parametry toho kterého paprsku
3. pravděpodobnost křížení: 70%, že se parametry dvou zvolených i-tých paprsků zprůměrují, 30% že se vezme celý i-tý paprsek jednoho z rodičů
4. počet jedinců v populaci: 50
5. počet generací: 75

Následují výsledky genetického algoritmu: výsledná fitness (obr. 3), rozmístění paprsků nejlepšího jedince 75. generace (obr. 4), trajektorie cesty nejlepšího jedince 75. generace v mapě (obr. 5).

Z grafu fitness evoluce (obr. 3) je patrné, že evoluce neprobíhala hladce a mezi jednotlivými populacemi byly značné rozdíly, nicméně je v něm zřejmá konvergentní tendence. Na obrázku trajektorie cesty nejlepšího jedince

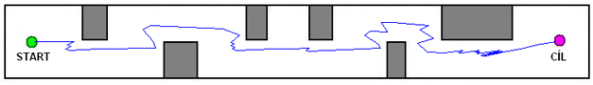


Obr. 3. Graf fitness evoluce. Plnou čarou je znázorněna fitness nejlepšího jedince v populaci, přerušovanou čarou je znázorněna průměrná fitness celé populace.



paprsek	α	délka paprsku	β	A	B
1	-11,8˚	100	107,4˚	-0,05	0,79
2	20,8˚	174	164,8˚	0,09	-0,31
3	-31,7˚	132	-136,9˚	0,14	1,11

Obr. 4. Rozmístění a parametry paprsků nejlepšího bota. P jsou vektory paprsků, S jsou příslušné vektory síly.



Obr. 5. Trajektorie cesty prostředím nejlepšího bota.

(obr. 5) vidíme, že zvládl úlohu vyhýbání se překážkám v cestě. Ze stejného obrázku lze také vidět, že pohyb není tak plynulý, jak bychom očekávali. Nabízí se dvě možnosti. Za prvé, buď úplně změnit model vyhýbání se překážkám - například bychom mohli uvažovat bota jako hmotný bot a počítat se setrvačností. Dále bychom mohli více přizpůsobit fitness funkci, která by penalizovala jakýkoli botův pohyb dozadu. Výsledného jedince jsme zatím neotestovali na komplexnějších mapách UT04. Ukázalo se však, že zvolený model lze parametrizovat pomoci genetických algoritmů a je schopen zvládnout problém vyhýbání se překážkám v cestě.

5.2 Vyšší vrstva chování - evoluce mechanismu výběru akcí

Druhou oblastí, na kterou jsme se zaměřili, je vysoko-úrovňová optimalizace botova chování. Vyhýbání se překážkám je jen jedním z mnoha úkolů, které musí bot v prostředí řešit. Tím nejzákladnějším problémem je takzvaný problém výběru akcí (action selection problem – ASP). Řešením problému výběru akcí je mechanismus výběru akcí (action selection mechanism – ASM). ASM na základě botova stavu a informací ze senzorů vybírá nejvhodnější možnou akci, t.j. je to funkce ze stavu bota a jeho okolí do množiny akcí.

Námi navržená funkcionální architektura pro popis algoritmu botova chování je inspirována konceptem stromů chování (behavior trees). Stromy chování jsou snadno pochopitelné a v posledních letech se společně s konečnými automaty staly jedním z nejčastěji používaných konceptuálních modelů pro popis chování herních postav. Listy stromu chování odpovídají atomickým chováním, která mohou být přímo vykonána v prostředí herního simulátoru. Odpovídají buď přímo jedné z atomických akcí (vystřel) nebo používají akcí nižší vrstvy chování (jdi k lékárničce). Vnitřní uzly odpovídají arbitrům, kteří rozhodují o vhodnosti akcí navrhovaných jim podřízenými uzly. Takovýto strom může být vyhodnocen dvěma způsoby: od kořene k listům nebo od listů ke kořeni. Naše implementace vyhodnotí nejdříve všechny listy. Výsledkem výpočtu v listu je dvojice [navrhovaná akce, vhodnost]. Výsledky jsou v dalším kroku postoupeny rodičovským uzlům, které na základě hodnot vhodnosti vyberou tu nejvhodnější akci. Tento postup se opakuje až ke kořeni, akce vybraná kořenem je nakonec provedena v prostředí. Architektura podobná této byla již s úspěchem testována v prostředí simulátoru Robocode [14].

Stromy chování mohou být přímo přeloženy do funkcionálního popisu. Na takto vzniklý program pak můžeme aplikovat metody genetického programování. Genem je v tomto případě samotný strom chování. Genetický operátor křížení zamění v rodičovských stromech dva podstromy stejného typu a operátor mutace nahradí podstrom náhodně vygenerovaným stromem požadovaného typu. Náhodné stromy jsou generovány rekurzivní procedurou jejíž vstupem je požadovaný návratový typ stromu a jeho maximální hloubka. Pro každý typ listu existuje zvláštní randomizační procedura (např. hodnota reálné konstanty je nastavena na náhodné číslo v intervalu $\langle 0, 1 \rangle$).

Program vzniklý překladem stromu chování může obsahovat tři typy funkcí:

- **Funkce chování**: funkce jejichž návratovým typem je [navrhovaná akce, její vhodnost], tomuto typu říkáme BehResult.
- **Sensorické funkce**: parametrizují funkce chování, jejich návratový typ je buď to reálné číslo normalizované na interval [0,1] (např. distanceTo(Location), heatlh() atd.) nebo typ specifický pro herní prostředí (např. funkce nearestEnemy() vrací ukazatel na objekt reprezentující nejbližšího nepřítele).
- **Matematické funkce**: $+$, $*$, $1 - x$, sin, min, max, konstanta.

Teď popíšeme jednotlivé skupiny funkcí podrobněji.

Funkce chování Primární funkce jsou předprogramovaná chování zaměřená na splnění základních úkolů, které může bot chtít v prostředí provést.

- ***stay****(double)*: Zůstaň stát na místě.
- ***wanderAround****(double)*: Procházej po předmětech v mapě.
- ***goTo****(Location, double)*: Jdi na danou lokaci.
- ***pickHealth****(Health, double)*, ***pickAmmo****(Ammo, double)*, ***pickWeapon****(Weapon, double)*, ***pickArmor****(Armor, double)*: Dojdi k nejbližšímu předmětu daného typu a seber jej.
- ***turnLeft****(double)*: Otoč se doleva.
- ***attackPlayer****(Player, double)*: Zaútoč na daného hráče. Toto chování je zodpovědné i za výběr nejvhodnější zbraně. S ohledem na vzdálenost nepřítele je vždy vybrána nejvhodnější zbraň s nenulovým počtem nábojů.

Všechny primární funkce mají alespoň jeden parametr typu double, hodnota předávaná v tomto parametru reprezentuje vhodnost akce navrhované chováním. Některé funkce mohou hodnotu vhodnosti ještě modifikovat. Pokud je například chování goTo zavoláno na neexistující lokaci, tak bude vhodnost chování nastavena na nula (goTo(null, 0.27) vrací dvojici [prázdná akce,0]).

Sekundární funkce umožňují vznik složených chování.

- ***sequenceArbiter****(BehResult, BehResult)*: Pokud je vhodnost prvního chování vyšší než 0.1, vrátí akci navrhovanou tímto chováním a její vhodnost, jinak vrací dvojici z druhého chování. Tento arbitr umožňuje vznik sekvenčních chování.
- ***highestActivation****(BehResult, BehResult)*: Porovná vhodnosti obou dvojic, jako výsledek vrátí tu dvojici, jejíž vhodnost je větší. Tento arbitr je vhodný pro vyjádření alternativních řešení daného problému.
- ***suitabilityWeighter****(BehResult, double)*: Vynásobí vhodnost v BehResult dvojici hodnotou druhého parametru.

Sensorické funkce Sensorické funkce slouží jako parametry pro funkce chování.

Interní senzory poskytují informace o botově vnitřním stavu.

- ***health()***: míra zdraví normalizovaná na interval $\langle 0, 1 \rangle$.
- ***ammo()***: počet nábojů pro aktuální zbraň normalizovaný na interval $\langle 0, 1 \rangle$.
- ***ammoForWeapon****(WeaponType)*: počet nábojů pro danou zbraň normalizovaný na interval $\langle 0, 1 \rangle$.
- ***hasWeapon****(WeaponType)*: 1 pokud bot má zbraň daného typu, 0 jinak.
- ***pain()***: změna zdraví za poslední iteraci / 30.

Externí senzory zprostředkovávají informace o okolním prostředí.

- ***time()***: Herní čas v sekundách.
- ***nearestAmmo()***, ***nearestArmor()***, ***nearestHealth()***, ***nearestWeapon()***: Nejbližší předmět daného typu.
- ***nearestEnemy()***: Nejbližší hráč z nepřátelského týmu.
- ***see****(Viewable)*: 1, pokud bot vidí daný Viewable objekt (Viewable objekty jsou hráči, místa na mapě atd.), 0 jinak.
- ***seeAnyEnemy()***: Zkratka za **see(nearestEnemy())**.
- ***distanceTo****(Location)*: Eukleidovská vzdálenost bota od dané lokace namapovaná na interval $\langle 0, 1 \rangle$ funkcí $\arctan$. Vrací -1, pokud zadaná lokace neexistuje.

Matematické funkce Na funkce max, min a $1 - x$ může být v některých kontextech nahlíženo jako na logické funkce and, or a not protože mnoho senzorů jejichž hodnota je pravda/nepravda jsou kódovány čísly $\{0, 1\}$.

Experiment Pro otestování navržené sady primárních a sekundárních funkcí bylo provedeno několik experimentů. Detaily těchto experimentů mohou být nalezeny na blogu *Genetic bots* [5]. Jeden z těchto experimentů teď bude popsán detailněji.

V tomto experimentu hrál evolvovaný bot hru typu *Death match* proti předprogramovanému botovi Hunterovi, který je řízen pomocí if-then pravidel napsaných v Javě (bot je součástí instalace platformy Pogamut). Podrobnosti nastavení genetického algoritmu následují.

1. Mapa: dm-TrainingDay
2. Generací: 80
3. Jedinců v každé generaci: 150
4. Iniciální generace: náhodné funkce s maximální hloubkou 5
5. Elita: 16 jedinců
6. Pravděpodobnost křížení: 0.2
7. Pravděpodobnost mutace: 0.2

[5] Genetic bots blog, http://artemis.ms.mff.cuni.cz/pogamut/tiki-view_blog.php?blogId=4

8. Selekční metoda: vážená proporcionální deterministická distribuce
9. Fitness: $dc + 50pk - (5d + \frac{ds}{10})$, kde dc je zranění způsobené evolvovaným botem, pk počet hráčů zabitých botem, d počet úmrtí bota a ds zranění, které bot utrpěl
10. Přesnost střelby: 0.5, tato hodnota dává napadenému botovi ještě šanci opětovat palbu
11. Výpočet fitness: každý bot bojoval po 90 sekund proti botovi Hunter

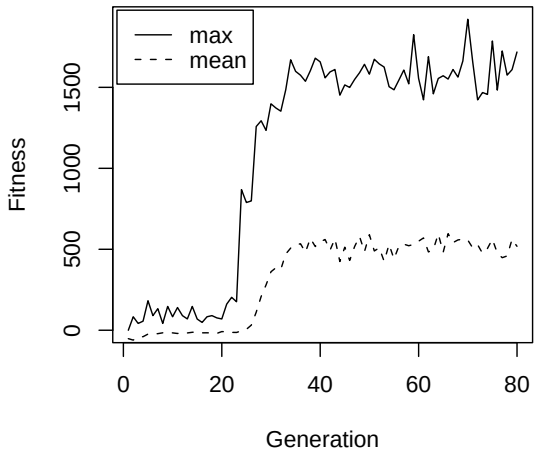
Mapa dm-TrainingDay je nejmenší z map oficiálně dodávaných se hrou UT04. Funkce fitness zvýhodňuje útočné chování oproti vyhýbání se střetům, protože to je požadované chování ve hře typu *Death match*. Odměna za zranění nepřítele je desetkrát větší než penalizace za vlastní zranění. Přesnost střelby nastavená na 0.5 dává botovi, na kterého byla zahájena střelba, ještě šanci utéct.

V prvních dvaceti generacích zkoušela evoluce náhodné strategie, mezi dvacátou a čtyřicátou generací se populace začala zlepšovat a bylo dosaženo lokálního optima. V dalších generacích se fitness již nezvyšovala. Obrázek 6 ukazuje, jak se fitness měnila v průběhu evoluce.

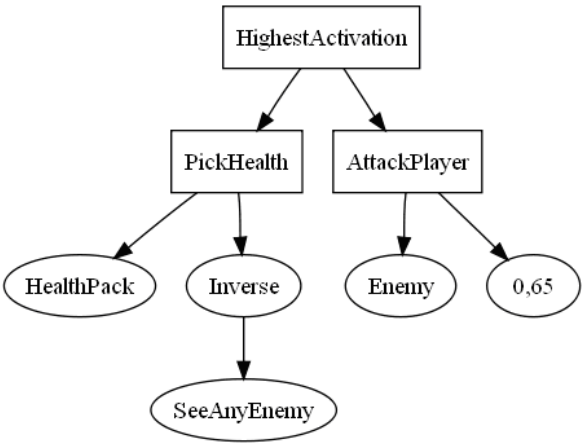
Obrázek 7 ukazuje strom chování jedince z poslední generace. Bot sbíral lékárničky, když neviděl nepřítele, a zaútočil na nejbližšího nepřítele, pokud nějakého uviděl.

Následující tabulka 1 ukazuje, jaká je výsledná akce toho stromu v závislosti na externím stimulu *SeeAnyEnemy*.

Výsledné chování není příliš komplexní, ale na takto malé mapě při hře typu *Death match* i lidští hráči používají podobnou vysokoúrovňovou strategii.



Obr. 6. Průměrná a maximální fitness v každé generaci.



Obr. 7. Strom chování nejlepšího jedince z osmdesáté generace.

6 Budoucí práce

V budoucí práci se zaměříme na detailnější rozbor obou vrstev chování a na přidání podpory pro nové herní módy. U nižší vrstvy plánujeme otestování navigace prostředím v jiných konfiguracích světa, případně vytvoření dalších modelů pro vyhýbání se překážkám a jejich porovnání. Vyšší vrstvu chování bychom chtěli rozšířit o podporu pro další typy her jako *Capture the flag* (CTF) a *Domination point*. Tyto herní módy nabízejí botovi více možných cest k dosažení úspěchu. Například v módu CTF se může bot specializovat na kradení nepřátelské vlajky, na zabíjení nepřátelských botů nebo může zvolit kompromisní strategii. Na základě strategie nalezené nejlepšími jedinci by se dalo zjišťovat, zda je daná mapa pro ten který herní mód vhodná a zda příliš nezvýhodňuje jednu strategii na úkor ostatních.

7 Závěr

V článku jsme představili funkcionální model pro popis vysokoúrovňového chování bota a model umožňující popis vyhýbání se překážek. Oba modely jsme implementovali a otestovali v prostředí hry Unreal Tournament 2004.

Experimenty ukázaly, že námi navrhované metody přinejmenším dokáží najít suboptimální smysluplná řešení zadaných problémů a potencionálně tím mohou ušetřit práci herních vývojářů.

Poděkování

Práce J. Gemrota, R. Kadlece and C. Broma byla podpořena grantem GA UK 1053/2007/AINF/MFF, Ministerstvem školství,mládeže a tělovýchovy (projekt MSM0021620838) a projektem 1ET100300517 programu Informační společnost. Práce P. Vidnerové byla podpořena projektem 1ET100300414 programu Informační společnost a záměrem AV0Z10300504 ”Computer Science for the Information Society: Models, Algorithms, Applications”.

Podmínka	Hodnota SeeAnyEnemy	Aktivace PickHealth	Aktivace Attack	Vítězné chování
Bot vidí nepřítele	1	0	0.65	Attack
Bot nevidí nepřítele	0	1	0	PickHealth

Tab. 1. Jak externí stimuly ovlivňují botovo rozhodování

Reference

1. R. Adobbati, A. N. Marshall, A. Scholer, and S. Tejada. 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: http://www.planetunreal.com/gamebots.
2. Alex J. Champandard. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors*. New Riders, Indianapolis, IN, USA, 2003.
3. N. Chapman. Neuralbot. URL: http://homepages.paradise.net.nz/nickamy/neuralbot/nb_about.htm, 1999.
4. J. Holm and J. D. Nielsen. Genetic programming - applied to a real time game domain. Master's thesis, Aalborg University, 2002.
5. R. Kadlec, J. Gemrot, O. Burkert, M. Bída, J. Havlíček, and C. Brom. Pogamut 2 - a platform for fast development of virtual agents' behaviour. In *Proceedings of CGAMES 07, La Rochelle, France*, 2007.
6. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
7. M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1996.
8. F. Mondada, E. Franzi, and A. Guignard. The Development of Khepera. In *First International Khepera Workshop*, HNI-Verlagsschriftenreihe, Heinz Nixdorf Institut, pages 7–14, 1999.
9. S. Priesterjahn, O. Kramer, A. Weimer, and A. Goebels. Evolution of human-competitive agents in modern computer games. pages 777–784, 2006.
10. C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference, San Jose, California.*, pages 763–782, 1999.
11. Scott D. Roth. Ray Casting for Modeling Solids. 18(2):109–144, February 1982.
12. J. Westra. Evolutionary neural networks applied in first person shooters. Master's thesis, University Utrecht, 2007.
13. M. Wooldridge and N. R. Jennings. Intelligent agents - theories, architectures and languages. In *Volume 890 of Lecture Notes in Artificial Intelligence. Springer-Verlag*, 1995.
14. B. G. Woolley and G. L. Peterson. Genetic evolution of hierarchical behavior structures. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation. London, England*, pages 1731–1738, 2007.

Automatické znovupoužitie údajov v kompozícii tokov práce gridových služieb

Ondrej Habala1[1], Branislav Šimo[2], Emil Gatial[2] a Ladislav Hluchý[2]
[1] Oddelenie distribuovaného spracovania údajov, Ústav informatiky, Slovenská akadémia vied
84507 Bratislava, Slovenská republika
[2]Ústav informatiky, Slovenská akadémia vied

Abstrakt. *V posledných rokoch sa množstvo výskumných projektov a softvérových produktov zaoberalo automatickou kompozíciou tokov práce aplikovanou na počítačové procesy. Takéto toky práce sú veľmi výhodnou metódou riadenia zložitých vedeckých a technických výpočtov, zložených z viacerých samostatných výpočtových modulov. Toky práce si takisto získali popularitu v gridovom počítaní, najmä v súvislosti s automatizáciou týchto procesov za použitia ich sémantickej anotácie. Avšak väčšina existujúcich prác sa zaoberá len problémom kompozície tokov procesov, pričom zanedbáva možnosť znovupoužitia existujúcej údajovej základne aplikačnej domény. V tomto článku popisujeme návrh systému, ktorý sa sústreďuje práve na tento nedostatok súčasného stavu riešení automatickej kompozície tokov práce webových služieb. Naše riešenie je založené na predchádzajúcej práci v projekte 7. RP K-Wf Grid a stavia na extenzívnom sémantickom popise webových služieb a dát cieľovej aplikačnej domény, uloženom vo forme ontológií. Cieľom je najmä umožniť skladanie takých tokov práce, ktoré využívajú už existujúce (i vypočítané) údaje a tým obchádzajú časti toku práce, ktoré by takéto údaje opakovane vytvárali. Riešenie je v súčasnosti vo fáze implementácie v projekte APVV SEMCO-WS.*

1 Úvod

Množstvo publikácií, výskumných projektov, i komerčných produktov [1–3] sa už zaoberalo problémom automatickej kompozície tokov práce počítačových procesov. Tento spôsob automatizácie výpočtov je atraktívny najmä pre softvérové inžinierstvo aplikované na vedecký výskum, kde komplikované vedecké výpočty často vyžadujú netriviálnu orchestráciu často desiatok až stoviek jednotlivých krokov. Už od vzniku gridového počítania bolo skladanie tokov práce gridových úloh intenzívne skúmané, najmä pre svoju využiteľnosť, dlhú históriu skúmania bez zamerania špecificky na gridové počítanie, a robustnú matematickú teóriu postavenú predovšetkým na orientovaných acyklických grafoch (DAG). V posledných rokoch sa pokroky v sémantickom webe a sémantických technológiách všeobecne zhmotnili i v podobe nástrojov pre gridové počítanie, a začal vznikať „sémantický grid [4]“, čiastočne i so zameraním na sémantické riadenie tokov práce gridových úloh. Avšak väčšina existujúcich riešení a publikácií sa zameriava len na úlohu komponovania tokov práce úloh – vykonateľných procesov – a zanedbáva existujúcu údajovú základňu aplikácie, ktorá je často veľmi rozsiahla a nie je nijako významne organizovaná. Tak vzniká situácia, keď aplikačné prostredie používateľovi ponúkne skonštruovaný tok práce, ktorý však on musí následne „naplniť“ údajmi, pričom tieto treba pracne vyhľadávať v databázach a katalógoch, čo v žiadnom prípade nie je práca pre človeka neznalého podrobností aplikačnej domény.

My sme navrhli a začali implementovať systém, ktorý sa zaoberá i výberom vhodných údajov pre konštruovaný tok práce. Naša práca je založená na výsledkoch projektu K-Wf Grid [8], a tieto rozširuje o nástroje schopné stanoviť aké údaje sú potrebné na dosiahnutie cieľa toku práce. Tento cieľ je jediné, čo musí používateľ popísať (sémanticky). Systém je založený na podrobných sémantických popisoch údajovej základne. Toky práce sú modelované Petriho sieťami, ktoré boli použité už i v projekte K-Wf Grid, a poskytujú veľmi pohodlnú a presnú metódu zápisu tokov práce, s možnosťami ktoré DAG systémy nemôžu dosiahnuť kvôli teoretickým obmedzeniam.

Zvyšok článku sa zaoberá najskôr projektom K-Wf Grid a jeho výsledkami, potom v jeho kontexte prezentuje náš príspevok. Systém je vyvíjaný v projekte SEMCO-WS [9], a jadro článku popisuje nami navrhnutý spôsob ako konštruovať toky práce ktoré využívajú nielen aplikačné komponenty, ale i dátovú základňu.

2 Výsledky projektu K-Wf Grid

Projekt Knowledge-based Workflow System for Grid Applications – K-Wf Grid – začal v septembri 2004, a skončil vo februári 2007. Bol veľmi úspešný v dosiahnutí svojich cieľov, a verejné predvedenie výsledkov na konferencii CGW'06 v Krakove, ako i záverečná recenzia projektu boli úspechmi. Konzorcium pozostávalo zo 6 partnerov, a na pozadí úspechu projektu bola i koncentrácia práce na jeden cieľ – automatickú kompozíciu tokov práce gridových služieb, s využitím sémantickej podpory a elegantných nástrojov pre integráciu používateľského rozhrania.

Vyvinuté nástroje boli otestované na 3 pilotných aplikáciách. My budeme popisovať iba tie časti výsledkov projektu, ktoré sú relevantné pre nami vyvíjaný systém.

Aplikácia projektu K-Wf Grid je sada webových a gridových služieb. Každá aplikácia začína svoj životný cyklus integrovaním do znalostnej bázy systému [10]. Proces integrácie je prevažne automatický [11], aplikačný expert musí len anotovať WSDL dokumenty služieb poznámkami označujúcimi vstupy a výstupy metód.

Po takejto integrácii môže byť aplikácia používaná. Používateľ zadá textový popis svojho cieľa, a nástroj užívateľského prostredia [12] nájde v znalostnej báze dostupné kontexty ktoré tomuto popisu zodpovedajú. Používateľ si jeden vyberie a kompozícia toku práce môže začať.

Keďže pracujeme s Petriho sieťami, v nasledujúcom texte používame termíny „aktivita“ pre procesy, „token“ pre inštancie vstupných a výstupných údajov, a „miesto“ pre vstupno-výstupné kanály akivít. Petriho sieť je bipartitný graf, kde sa aktivity a miesta vždy stridajú (t.j. hrana nemôže začínať i končiť len v aktivitách, alebo len v miestach).

Tok práce je formovaný a vykonaný vo viacerých krokoch. Úvodný tok práce je popis želaného výsledku, vychádzajúceho z jednej abstraktnej aktivity,

reprezentujúcej celý tok práce. Táto aktivita je následne rozvinutá [13] do Petriho siete pozostávajúcej z tried aktivít – v našom prípade zo zvolených rozhraní webových služieb. Tým sa zadefinuje funkcionalita jednotlivých aktivít a postu spracovania údajov. V poslednom ktoru sa pre rozhrania vyberú sady dostupných inštancií webových služieb [14] ktoré tieto rozhrania implementujú. Keď je tok práce takto skonštruovaný, je naplánovaný [5] a vykonaný. Pri plánovaní sa berie do úvahy predovšetkým kvalita inštancií, ohodnotená pri predchádzajúcich spúšťaniach tokov práce – systém sa učí.

Pri spúšťaní toku práce systém interaguje s používateľom za účelom získania údajov pre aktivity (tokenov). Napriek tomu že i tento krok je riešený pomerne komfortným spôsobom – vývojári aplikácie majú možnosť integrovať do systému doménovo špecifické webové formuláre pre zadávanie údajov – je evidentným nedostatkom automatickej kompozície tokov práce tak, ako bola implementovaná v projekte K-Wf Grid. A práve tento krok sa snažíme zautomatizovať spôsobom, popísaným v nasledujúcich kapitolách.

3 Sémantický riadená kompozícia tokov práce v SEMCO-WS

Projekt SEMCO-WS je projektom APVV, s konzorciom pozostávajúcim zo 4 členov – Ústav informatiky SAV, FEI TU Košice, FIIT STU Bratislava, a firmy MICROSTEP-MIS (ktorá je poskytovateľom pilotnej aplikácie a potenciálnym zákazníkom i komercializátorom výsledkov projektu). Projekt začal v roku 2007 a končí v roku 2010.

Projekt stavia na výsledkoch K-Wf Gridu, a snaží sa upraviť alebo doplniť niektoré jeho nástroje. Napríklad zatiaľ čo v K-Wf Gride bol proces kompozície a vykonania toku práce vizualizovaný grafickým nástrojom, užívateľ mohol do výsledného toku práce zasahovať len zmenou dát (tokenov). V SEMCO-WS bude možné tok práce i editovať podľa potrieb používateľa (zmeniť rozloženie a prepojenie aktivít a miest). Najdôležitejšou zmenou je však rozšírenie procesu kompozície o prácu s existujúcimi údajmi a ich metaúdajmi (sémantikou).

4 Automatický výber údajov pri kompozícii toku práce

Na to, aby bol vyvíjaný systém schopný vyberať nielen aktivity v toku práce, ale i úvodné údaje (tokeny) s ktorými budú aktivity spúšťané, musí byť schopný

1. identifikovať obsah údajov existujúcich v aplikácii,
2. vyvodiť parametre potrebných vstupných tokenov pre aktivitu na základe parametrov výstupných tokenov, ktoré má vyprodukovať,
3. dekomponovať výstupné štruktúry webových služieb modelovaných aktivitami toku práce do jednotlivých výstupných tokenov, a
4. vytvoriť vstupné štruktúry pre webové služby zo vstupných tokenov (aktivita môže mať viacero vstupných miest, ale ňou abstrahovaná webová služba prijme len jednu vstupnú štruktúru v SOAP obálke).

Riešenia navrhnuté pre tieto 4 problémy sú popísané v nasledujúcom texte.

4.1 Obsah údajov – metaúdaje

Akýkoľvek údaj v systéme je reprezentovaný tokenom. Ak je tým údajom číslo, zápis v tokene je jednoduchý – číslo v XML obálke. Ak je to súbor, môže byť jeho obsah priamo zapísaný v tokene (pre veľmi malé súbory), alebo môže token obsahovať URL súboru (v HTTP, FTP, GRIDFTP, LFN alebo podobnej schéme). Token môže obsahovať i adresu databázy a dopyt produkujúci požadovaný údaj, alebo akúkoľvek inú reprezentáciu informácie. Obsah tokenu je vždy aplikačne závislý, a systém nemusí a ani nemôže byť schopný ho interpretovať. Akýkoľvek token je buď zadaný a popísaný používateľom, alebo vytvorený a popísaný komponentom aplikácie. Systém musí mať k dispozícii len metaúdaje tokenu, aby bol schopný zhodnotiť jeho použiteľnosť v určitom kroku vytváraného toku práce.

Metaúdaje tokenu sú zaznamenané v jazyku OWL (OWL-S), a pozostávajú z dvoch častí – generickej, definovanej systémom, a aplikačne závislej časti, vytvorenej pre konkrétnu doménu. Toto delenie sa ukázalo byť užitočným v projekte K-Wf Grid, a SEMCO-WS ho prevzal. Dôvodom prečo bol použitý jazyk OWL a nie napr. WSMO [15] je fakt, že K-Wf Grid už vyvinul nástroje podporujúce OWL.

Keďže systém musí poznať údaje existujúce v aplikácii (bod 1. vyššie), a údaje sú reprezentované tokenmi, je súčasťou riešenia i XML databáza obsahujúce všetky existujúce tokeny. Metaúdaje tokenov sú uložené v OWL úložisku, kde je i identifikátor tokenu v XML databáze, pomocou ktorého je možné ho (po vyhľadaní podľa metaúdajov v OWL úložisku) sprístupniť.

Každý novovytvorený token musí byť popísaný metaúdajmi. Toto je možné urobiť dvoma spôsobmi – buď je popis vytvorený aplikačným komponentom (špeciálna metadáta generujúca webová služba, plug-in Java trieda pre systém, a pod.), alebo sú metaúdaje vytvorených výstupných tokenov inferované z metaúdajov vstupných tokenov. Na použitie druhého spôsobu je potrebné formálne popísať transformáciu, ktorú realizuje webová služba aktivity. Tento popis je samozrejme taktiež aplikačne závislý a musí byť vytvorený aplikačným vývojárom.

4.2 Spätné odvodzovanie parametrov vstupných tokenov na základe parametrov požadovaných výstupov

Proces konštrukcie toku práce v K-Wf Gride bol už dostatočne popísaný [13,14,6]. Tento proces nezohľadňoval existujúce údaje, a vždy predpokladal že celý tok práce treba prepočítať od začiatku, i ak boli k dispozícii niektoré medzivýsledky schopné nahradiť časť toku práce. Tento proces používal spätné odvodzovanie od finálnej aktivity až po tie úvodné. V SEMCO-WS sa tiež používa odvodzovanie, ale už i so zohľadnením existujúcich tokenov. Celý proces tvorby toku práce možno zapísať nasledujúcim algoritmom:

```
Program construct_workflow
     (token_metadata_list)
//The input is a list of metadata --
//descriptions of all tokens we wish to
//produce with the constructed workflow
```

```
Variables:
workflow    //list of components of the
            //constructed workflow
token
activity
token_metadata  //member of
                //token_metadata_list

1.  Foreach token_metadata in
    token_metadata_list
        a.  token ← find in token_db based on
            token_metadata
        b.  If token ≠ nil
            {
                workflow = workflow + token
            }
            Else
            {
                  Find activity able to produce
                  token;
                  workflow = workflow +
                  activity;
                  token_metadata_list =
                  token_metadata_list  +
                  token_metadata of all input
                  tokens of activity
            }
2.  If token_metadata_list is not empty, goto 1
3.  Output workflow
```

Z popisu algoritmu je zrejmé, že najskôr sa snažíme nájsť potrebné údaje, a ak tieto nie sú dostupné , nájdeme aktivitu ktorá ich dokáže vytvoriť. Samozrejme, že potom musíme nájsť údaje alebo ich producenta pre túto aktivitu, atď. až dospejeme do bodu, kedy už nemáme žiadnu aktivitu bez definovaných vstupných tokenov. Horeuvedený popis algoritmu je len schématický, samotný proces identifikovania aktivity je oveľa komplikovanejší, musí riešiť problém potenciálneho zacyklenia, optimalizácie produkcie tokenov (najkratším zreťazením aktivít a miest), a situáciu, kedy jednoducho nie sú k dispozícii potrebné údaje, ani žiaden ich producent (vtedy je možné kontaktovať používateľa a požadovať od neho definovanie vstupných tokenov, ako to bolo v K-Wf Gride).

4.3 Od tokenov a aktivít k dátovým štruktúram a službám

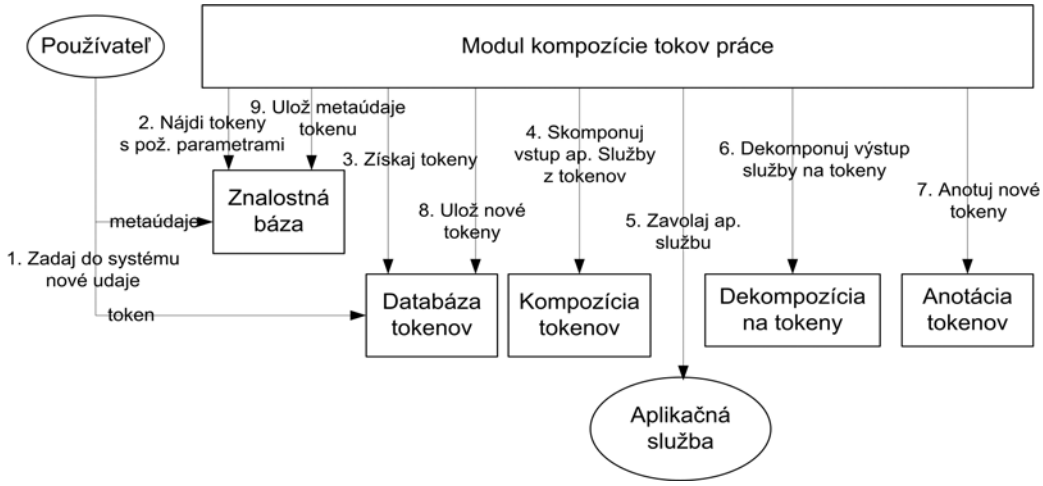
Naša Petri sieť operuje nad aktivitami, miestami a tokenmi. Aplikácie však pozostávajú z webových služieb, súborov a databáz. Na to aby sme vedeli z Petriho sieťe skonštruovať potrebné volania webových služieb, musíme byť schopní splniť podmienky bodov 3. a 4. z úvodu 4. kapitoly – vedieť skomponovať vstupnú štruktúru pre webovú služby zo vstupných tokenov aktivity, a vedieť z výstupnej štruktúry (odpovede) služby extrahovať výstupné tokeny aktivity.

Za týmto účelom sme navrhli rozšírenie WSDL dokumentov služieb o anotáciu (v `<documentation>` tagu), obsahujúcu XSLT transformácie. Tieto definujú spôsob ako z kompozitu vstupných tokenov vytvoriť vstupnú štruktúru služby, a naopak ako z odpovede služby vyextrahovať výstupné tokeny. Celý tento proces je pomerne jednoduchý, pri spustení („odpálení“ v terminológii Petriho sietí) aktivity systém spojí vstupné tokeny do jedného XML fragmentu, na tento aplikuje XSLT transformáciu z WSDL dokumentu služby, a výsledný XML použije ako náplň SOAP obálky pre volanie služby. Takisto po obdržaní odpovede od služby je táto transformovaná pomocou XSLT na XML fragment s definovaným formátom, z ktorého sú potom vybraté jednotlivé výstupné tokeny.

4.4 Proces kompozície toku práce

Pre názornú prezentáciu problému sme celý proces kompozície a vykonania toku práce rozdelili do niekoľkých krokov (viď Obr. 1).

1. Užívateľ zadal inicializačné údaje pre tok práce, čiže parametre želaného výstupu. Používateľ je doménový expert, takže je to pre neho jednoduchá úloha, a systém mu to umožňuje urobiť pomocou webového formulára.
2. Nástroj na kompozíciu toku práce používa pri hľadaní tokenov poskytujúcich potrebné údaje znalostnú bázu systému.
3. Ak sa nájde potrebný token, systém ho použije; inak je použitá aktivita ktorá taký token produkuje (nie je zobrazené na Obr. 1, keďže to nie je zameraním našej práce a dostatočný počet publikácií na túto tému existuje napr. i z projektu K-Wf Grid).
4. Počas vykonávania toku práce musí systém vytvoriť vstupné štruktúry pre webové služby, pomocou XSLT transformácie (Kapitola 4.3).
5. Vytvorená vstupná štruktúra je použitá pri volaní webovej služby.
6. Získaný výstup je dekomponovaný do jednotlivých tokenov, taktiež automaticky pomocou preddefinovanej XSLT transformácie.
7. Vytvorené tokenu musia byť anotované; toto je tiež aplikačne závislý krok, ale ako sme už povedali je možné vyhnúť sa pomocným službám a plug-in modulom ak použijeme formálny matematický opis transformácie implementovanej webovou službou. Tento popis je tiež možné pridať do WSDL dokumentu; alternatívne i URL pomocnej služby alebo identifikátor (napr. meno Java triedy) plug-in modulu je možné dať do WSDL.
8. Vytvorené tokeny sú uložené v databáze pre neskoršie použitie.
9. Metaúdaje tokenov sú uložené v znalostnej báze systému, a ďalší krok iterácie môže začať od bodu 2 (ak už nie je tok práce skonštruovaný).



Obr. 1. Proces automatickej kompozície – vyhľadávanie, spracovanie, vytváranie údajov.

Takisto ako v algoritme v Kap. 4.2 ani tu sa nezaoberáme špeciálnymi sutáciami, ako je nedostupnosť akéhokoľvek tokenu alebo producenta tokenov v niektorom kroku.

5 Záver

V tomto článku sme ukázali, že automatická kompozícia tokov práce v distribuovanom prostredí (v našom prípade v prostredí architektúry SOA a webových služieb) môže pracovať i so znovupoužitím dostupných údajov, ak sú tieto dostatočne sémanticky opísané. Tri kľúčové komponenty umožňujúce takúto automatizáciu sú popisy transformácie parametrov výstupných údajov na parametre vstupných údajov, popis inverznej transformácie, a popis schopností aplikačných služieb.

Prezentovaný návrh je v súčasnosti predmetom implementácie v projeke SEMCO-WS. Tým pokračuje práca započatá v projekte K-Wf Grid, aspoň v niektorých jeho oblastiach, ktoré je možné zlepšiť – automatické znovupoužitie údajov je určite jednou z týchto oblastí. Prototyp systému bude dostupný do konca roku 2008, a s ukončením projektu na začiatku roku 2010 bude k dispozícii i kompletný a overený systém.

Poděkování

Táto práca je podporovaná projektami SEMCO-WS APVV-0391-06, ADMIRE EU 7FP 215024, VEGA 2/7098/27.

Referencie

1. Bubak, M., Gubala, T., Kapalka, M., Malawski, M., Rycerz, K.: Grid Service Registry for Workflow Composition Framework. ICCS 2004, in LNCS vol. 3038, Springer, 2004. ISBN 978-3-540-22116-6, pp. 34-41.
2. VDS - The GriPhyN Virtual Data System. http://www.ci.uchicago.edu/wiki/bin/view/VDS/VDSWeb/WebMain (Accessed Jan. 2008)
3. Krishnan, S., Wagstrom, P., von Laszewski, G.: GSFL: A Workflow Framework for Grid Services. In Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, 1L 60439, U.S.A., 2002.
4. Semantic Grid Community Portal. http://www.semanticgrid.org/ (Accessed Jan. 2008)
5. Wieczorek, M., Prodan, R., Fahringer, T.: Comparison of Workflow Scheduling Strategies on the Grid. Proceedings of 6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005), LNCS 3911, Springer, pp. 792-800, ISBN 3-540-34141-2, Poznan, Poland, 2005.
6. Hoheisel, A., Der, U.: An XML-based Framework for Loosely Coupled Applications on Grid Environments. In: Sloot, P.M.A. et al. (eds.): ICCS 2003. Lecture Notes in Computer Science, Vol. 2657, Springer-Verlag, 2003, pp. 245–254.
7. Alt, M., Gorlatch, S., Hoheisel, A., Pohl, H.W.: A Grid Workflow Language Using High-Level Petri Nets. Proceeedings of Parallel Processing and Applied Mathematics, LNCS Vol. 3911, pp. 715-722 , Springer-Verlag, 2006.
8. Knowledge-based Workflow System for Grid Applications (K-Wf Grid). EU 6th FP Project, 2004-2007. http://www.kwfgrid.eu (Accessed Jan. 2008).
9. Semantic composition of Web and Grid Services (SEMCO-WS). Slovak APVV project, 2007-2009. http://semco-ws.ui.sav.sk/ (Accessed Jan. 2008).
10. Kryza, B., Pieczykolan, J., Babik, M., Majewska, M., Slota, R., Hluchy, L., Kitowski, J.: Managing Semantic Metadata in K-Wf Grid with Grid Organizational Memory. In M. Bubak, M. Turala, and K. Wiatr, editors, Proceedings of Cracow Grid Workshop - CGW'05, November 20-23 2005, pp. 66–73, Krakow, Poland, 2005.
11. Habala, O., Babik, M., Hluchy, L., Laclavik, M., Balogh, Z.: Semantic Tools for Workflow Construction. In Alexandrov, van Albada, Sloot, Dongarra, editors, Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III, volume 3993 of LNCS pp. 980–987. Springer, 2006.
12. Laclavik, M., Seleng, M., Hluchy, L.: User Assistant Agent (EMBET): Towards Collaboration and Knowledge Sharing in Grid Workflow Applications. In Cracow '06 Grid Workshop: K-Wf Grid. Cracow, Poland, 2007,ISBN 978-83-915141-8-4, pp. 122-130.
13. Gubala, T., Herezlak, D., Bubak, M., Malawski, M.,: Semantic Composition of Scientific Workflows Based on the Petri Nets Formalism. Proc. of e-Science 2006, 2006, Amsterdam, ISBN-0-7695-2734-5.
14. Dutka, L., Kitowski, J.: AAB - Automatic Service Selection Empowered by Knowledge. In M. Bubak, M. Tura³a, K. Wiatr (Eds.), Proceedings of Cracow Grid Workshop - CGW'05, November 20-23 2005, ACC-Cyfronet UST, 2006, Kraków, pp. 58.
15. Web Service Modeling Ontology. http://www.wsmo.org/ (Accessed Mar. 2008).

Vztahy mezi segmenty – segmentační schémata českých vět

Markéta Lopatková[1] and Tomáš Holan[2]

[1] ÚFAL MFF UK, Praha, lopatkova@ufal.mff.cuni.cz
[2] KSVI MFF UK, Praha, Tomas.Holan@mff.cuni.cz

Abstrakt *Syntaktická analýza vět přirozeného jazyka, základní předpoklad mnoha aplikovaných úkolů, je složitá úloha, a to zejména pro jazyky s volným slovosledem. Přirozeným krokem, který snižuje složitost vstupních vět, může být vytvoření modulu, ve kterém se určí struktura souvětí ještě před úplnou syntaktickou analýzou. Navrhujeme využít pojem segmentů, snadno automaticky rozpoznatelných úseků vět. Určujeme 'segmentační schémata' popisující vzájemné vztahy mezi segmenty – zejména souřadná a apoziční spojení, podřadná spojení, případně vsuvky. V tomto článku představujeme vývojový rámec, který umožňuje vyvíjet a testovat pravidla pro automatické určování segmentačních schémat. Popisujeme dva základní experimenty – experiment se získáváním segmentačních schémat ze stromů Pražského závislostního korpusu a experiment se segmentačními pravidly aplikovanými na prostý text. Dále navrhujeme míry pro vyhodnocování úspěšnosti segmentačních pravidel.*

1 Motivace

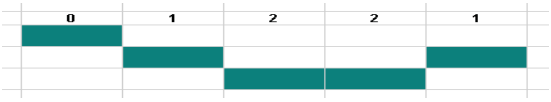
Syntaktická analýza vět přirozeného jazyka, základní předpoklad mnoha aplikovaných úkolů, je složitá úloha, a to zejména pro jazyky s volným slovosledem. Dlouhodobé úsilí přineslo řadu nástrojů pro parsing, které mají poměrně vysokou spolehlivost u krátkých a relativně jednoduchýh vět, bylo však ukázáno, že jejich úspěšnost pro složitější souvětí výrazně klesá (viz např. [1]).

Přirozeným krokem, který snižuje složitost vstupních vět a zejména souvětí, může být vytvoření modulu mezi morfologickou a syntaktickou analýzou, ve kterém by se určila struktura souvětí. Navrhujeme využít pojem segmentů, tedy lingvisticky motivovaných, přitom však snadno automaticky rozpoznatelných úseků vět, tak jak byl tento pojem navržen v [2] a modifikován např. v [3]. Tam byla popsána i výchozí sada pravidel, na jejichž základě lze (nedeterministicky) určit segmentační schéma zachycující vztahy jednotlivých segmentů v souvětí. Dalším krokem před samotnou syntaktickou analýzou by byl odhad jednotlivých klauzí, ze kterých se dané souvětí skládá.

Ukažme si základní myšlenku na příkladu konkrétní věty z tisku (1). Větu nejprve rozdělíme na jednotlivé segmenty; zde za hranice segmentů považujeme interpunkční čárku, souřadicí spojku *a*, závorky a koncovou tečku (ve větě podtrženy). Potom zachytíme jejich možné vzájemné vztahy – rozlišujeme souřadná a apoziční spojení, podřadná spojení, případně vsuvky. Takto získáme **segmentační schéma**, které umožňuje vymezit základní syntaktickou strukturu souvětí ještě před úplnou syntaktickou analýzou.

(1) *S tím byly trochu problémy , protože starosta v řeči rád zdůrazňoval své vzdělání (však studoval až v Klatovech a v Roudnici), a Víťa tedy občas nutně trochu tápal .*

První segment tvoří hlavní větu souvětí (neobsahuje žádný podřadicí výraz a je zde určité sloveso *byly*). Umístíme ho proto na nultou, základní úroveň v segmentačním schématu. Druhý segment je uvozen podřadicí spojkou *protože* a obsahuje určité sloveso *zdůrazňoval*, můžeme ho určit jako segment podřízený prvnímu segmentu a umístit na první nižší úroveň ve schématu. Následuje otevírací závorka, kterou interpretujeme jako začátek vsuvky, třetí segment tedy umístíme opět o úroveň níž. Čtvrtý segment je oddělen souřadnou spojkou *a*, umístíme ho proto na stejnou úroveň jako třetí segment (třetí segment obsahuje určité sloveso *studoval*, čtvrtý určité sloveso neobsahuje, půjde tedy zřejmě o koordinaci větně členskou). Uzavírací závorka uzavírá vsuvku tvořenou 3. a 4. segmentem. Vzhledem k tomu, že poslední segment obsahuje slovo *tedy*, chápeme trojici *, a tedy* jako příznak souřadnosti, pátý segment analyzujeme jako koordinaci buď k prvnímu, nebo k druhému segmentu. Segmentační schéma můžeme vyjádřit graficky – na obrázku 1 je jedno z možných schémat pro větu (1).



Obr. 1. Segmentační schema 01221.

Pokud bychom uměli před úplnou syntaktickou analýzou dostatečně bezpečně určit vztahy mezi jednotlivými segmenty, případně určit možnou strukturu klauzí, syntaktická analýza tohoto souvětí by se zjednodušila a mohla by vykazovat lepší výsledky.

Navíc se ukazuje, že při řadě důležitých aplikačních úloh – jako je např. vyhledávání informací, určování struktury dokumentů a jejich hlavních a vedlejších témat – není potřeba plná syntaktická analýza. Bylo by proto zajímavé zkoumat, nakolik se můžeme omezit pouze na zpracování segmentů na "vyšších" úrovních (a zanedbat hlouběji zanořené struktury).

O přínosech obdobných metod pro analýzu typologicky odlišných jazyků svědčí např. [4] či [5].

Hlavním cílem tohoto článku je představit vývojový rámec, který umožňuje dále testovat pravidla pro automatické určování segmentačních schémat. Po vymezení pojmu segmentu a segmentačního schématu (oddíl 2) popisujeme dva základní experimenty – experiment se získáváním segmentačních schémat ze stromů Pražského závislostního korpusu (oddíl 3.1) a experiment se segmentačními pravidly aplikovanými na prostý text (oddíl 3.2). Následuje oddíl 4, ve kterém představujeme vhodné míry pro vyhodnocování úspěšnosti segmentačních pravidel. Porovnáváme schémata získaná aplikací těchto dvou sad pravidel s ručně anotovaným vzorkem složitějších vět (i když jsme si vědomi pouze orientačního charakteru těchto výsledků).

2 Vymezení segmentů a jejich příznaky, segmentační schéma

Pod pojmem věta / souvětí zde rozumíme část textu, která je větou v typografickém smyslu (začíná velkým písmenem a končí koncovou interpunkcí, nejčastěji tečkou, otazníkem či vykřičníkem).[1] Jde tedy o posloupnost lexikálních jednotek (v anglicky psané literatuře označované jako 'tokens') $w_1 w_2 \dots w_n$, kde každá položka w_i reprezentuje buď jednu slovní formu daného přirozeného jazyka, nebo interpunkční znaménko (tečku, čárku, otazník, závorku, pomlčku, dvojtečku, středník, ...). Předpokládáme, že ke každé lexikální jednotce je k dispozici její morfologická analýza.

Na základě morfologické informace a slovní formy rozčleníme větu na jednotlivé úseky a těmto úsekům přiřadíme syntaktické příznaky.[2]

Hranice segmentu

Hranice jsou takové tokeny (slovní formy nebo interpunkce) či jejich posloupnosti, které rozdělují větu na jednotlivé dobře rozeznatelné části označované jako segmenty.

V následujících experimentech považujeme za **elementární hranice** následující symboly:

interpunkce: čárka, dvojtečka, středník, otazník, vykřičník, pomlčka (všech délek), otevírací a uzavírací závorka (všech druhů), svislítko, uvozovky (všech typů); tedy symboly *, : ; ? ! - ( )* [ ] | {} ‘ ’ “ ” , ‘ „ “
interpunkce na konci věty
souřadné spojky: morfologická značka začínající dvojicí `J^` (viz [6])

Pokud ve větě následuje několik elementárních hranic bezprostředně za sebou, označujeme maximální neprázdnou posloupnost těchto hranic jako **(složenou) hranici**.

Segmentem S potom rozumíme maximální neprázdnou posloupnost po sobě jdoucích slovních forem $w_1 w_2 \dots w_s$, která neobsahuje žádnou hranici.

Pro určování jednotlivých segmentů využíváme následující **doplňující pravidla:**

- Každá věta začíná a končí hranicí (pokud by nebyla hranice na začátku či na konci věty, doplňuje se prázdná hranice).
- Pracujeme se zjednoznačněnou koncovou tečkou (tj. uvnitř věty tečka není hranicí).

Poznamenejme zde, že hranice, určené na základě morfologické analýzy, nejsou nutně jednoznačné. Interpunkční znaménka jsou v užívaném morfologickém slovníku jednoznačná, to ale neplatí pro souřadné spojky (např. slovní forma *ale* je jednak souřadicí spojka, jednak tvar patřící ke třem substantivním lemmatům *ala*). Obecně proto připouštíme nejednoznačnou segmentaci věty. Pro češtinu ovšem existují velmi spolehlivé automatické nástroje, tzv. taggery, které vybírají pro každý token právě jednu morfologickou značku (nejvyšší publikovaná přesnost (accuracy) na prvních dvou pozicích morfologické značky je 99.36%, viz [7]). Proto budeme nadále pracovat s daty již předem morfologicky zpracovanými (správnou morfologickou analýzu přebíráme z PDT), a tedy s jednoznačně určenými hranicemi segmentů.

Příznaky segmentu

Morfologická analýza textu obsahuje ovšem řadu dalších (více či méně spolehlivých) informací, které lze využít pro určování vztahů mezi jednotlivými segmenty. Tyto informace zachycujeme pomocí **příznaků**, které přidělujeme jednotlivým segmentům – zatím pracujeme pouze s příznakem podřízenosti. Předpokládáme, že dalšími důležitými příznaky bude například příznak souřadnosti a příznak určitého slovesa.

Příznak podřízenosti (PP). Příznak podřízenosti se přiděluje takovému segmentu, který obsahuje (alespoň jednu) slovní formu, jejíž morfologická značka má první dvě pozice z následujícího výčtu (pro zájmena a číslovky), nebo je jedním z uvedených zájmenných příslovcí:

podřadicí spojka: `J,`
tázací/vztažné zájmeno: `P4`, `PE`, `PJ`, `PK`, `PQ`, `PY`
číslovka: `C?`, `Cu`, `Cz`
zájmenné příslovce: *jak, kam, kde, kdy, proč, kudy*

[1] Rozčleněním textu na věty se zde nezabýváme, přebíráme je z Pražského závislostního korpusu.

[2] Zde se odchylujeme od návrhu v [3], kde se uvažovaly tzv. separátory, které sloužily jednak jako hranice, jednak jako příznaky podřízenosti.

Segmentační schéma

Vztahy mezi segmenty popisujeme pomocí segmentačních schémat, která zachycují **úrovně vnoření jednotlivých segmentů**. Základní myšlenka úrovní segmentů je jednoduchá:

- Segmenty tvořící (všechny) hlavní klauze souvětí mají úroveň 0;
- Segmenty, tvořící klauze závislé na klauzích se segmenty na úrovni k, mají úroveň $k+1$ (úroveň vnoření je vyšší);
- Segmenty, tvořící koordinované, příp. aponované výrazy, mají stejnou úroveň;
- Segmenty, tvořící vsuvky (např. obsahy závorek), mají úroveň vnoření zvýšenu o 1 oproti segmentům, které je obklopují.

Segmentační schéma můžeme vyjádřit graficky (viz obrázek 1) nebo jako vektor čísel (pro větu (1) dostaneme dva vektory odpovídající dvěma segmentačním schématům (01220) a (01221)).

3 Experimenty s automatickým určováním segmentačních schémat

3.1 Jak získat segmenty ze syntaktických stromů?

Zde popíšeme možný postup, jak z analytické roviny Pražského závislostního korpusu[3] (PDT, [8]) určovat segmentační schémata pro jednotlivé věty. Analytická rovina PDT zachycuje povrchovou syntaxi, tedy v zásadě tytéž informace, které by měly umožnit segmentaci vět a vymezení možných úrovní jednotlivých segmentů.

Věta na analytické rovině je zachycena jako závislostní strom – uzly odpovídají slovním tvarům a interpunkci, hrany primárně závislostním vztahům (hrany považujeme za orientované od závislého k řídícímu uzlu). Mimo závislostní vztahy jsou zde zachyceny též vztahy koordinace a apozice – uzel odpovídající koordinační spojce (příp. jinému spojovacímu výrazu nebo interpunkci) je rodičem jednotlivých koordinovaných výrazů, podobně spojovací výraz uvádějící apozici je rodičem aponovaných výrazů. Typy vztahů reprezentované hranami jsou dále specifikovány tzv. analytickými funkcemi. V závislostní reprezentaci věty tedy nenajdeme uzly, které by odpovídaly konceptu segmentů, přesto by měla obsahovat informace, na jejichž základě lze segmentační schéma věty vymezit.

Pro popis pravidel potřebujeme zavést pojem cesty mezi segmenty a dále skupiny segmentů. Řekneme, že pro větu W **vede hrana ze segmentu** S_i **do segmentu** S_j ($S_i, S_j \subset W$), pokud pro nějaké slovo $u \in S_i$ existuje slovo $v \in S_j$ takové, že v závislostním stromě T věty W existuje cesta z u do v. Dále řekneme, že pro větu W **vede cesta ze segmentu** S_i **do segmentu** S_j ($S_i, S_j \subset W$), pokud existuje posloupnost segmentů $S_i = S_{p_1} .. S_{p_m} = S_j,\ S_{p_k} \subset W (k = 1 ..m)$ taková, že pro každé $k = 1 .. m-1$ vede hrana ze segmentu S_{p_k} do segmentu $S_{p_{k+1}}$. **Skupina segmentů** G je taková množina segmentů věty W, ve které pro každou dvojici segmentů $S_i, S_j \in G$ platí, že z S_i vede cesta do S_j (a tedy zároveň z S_j vede cesta do S_i).

Při získávání segmentačního schématu pro jednotlivé věty z PDT postupujeme následujícím způsobem.

Určení segmentů: Prvním krokem pro získání segmentačního schématu je určení hranic a segmentů.

Skupiny segmentů: Určíme jednotlivé skupiny segmentů.

Úroveň nula: Všechny segmenty ze skupin segmentů, ze kterých vede cesta do kořene závislostního stromu T buď přímá (tj. hrana) nebo pouze přes uzly reprezentující (elementární) hranice, dostanou úroveň nula.

Koordinace a apozice: Sousedí-li se segmentem S_i, pro který známe úroveň, segment S_j s neznámou úrovní a odpovídá-li hranice mezi nimi koordinačnímu či apozičnímu výrazu (např. uzel reprezentující elementární hranici má analytickou funkci Coord či Apos), dostane segment S_j stejnou úroveň jako segment S_i.

Hlouběji vnořené úrovně: Všechny segmenty ze skupin segmentů, které nemají stanovenu úroveň a ze kterých vede cesta buď přímá (tj. hrana) nebo pouze přes uzly reprezentující elementární hranice do segmentu s úrovní k, dostanou úroveň $k+1$.

Koordinace a apozice: Opět zkontrolujeme sousedy všech segmentů se známou úrovní (viz výš).

Tento postup opakujeme, dokud nejsou určeny úrovně vnoření všech segmentů.

Tato výchozí sada pravidel pro získání segmentů z PDT určí pro každou vstupní větu reprezentovanou analytickým stromem právě jedno segmentační schéma (ne nutně správné).

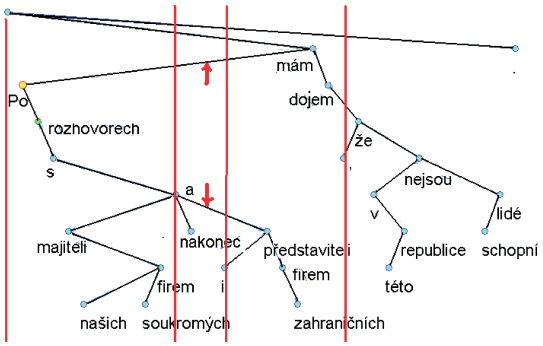
Ukažme si postup na konkrétní větě (2), jejíž analytický strom je na obrázku 2.

(2) *_ Po rozhovorech s majiteli našich soukromých firem a nakonec i představiteli firem zahraničních mám dojem _ že v této republice nejsou schopní lidé _.*

Věta (2) se skládá ze čtyř segmentů (ve větě hranice podtrženy, na obrázku odděleny svislými čarami). První a třetí segment tvoří skupinu (hrana z uzlu *po* do uzlu *mám* a zároveň cesta z uzlu *představiteli* do uzlu *s*, viz šipky). Tyto dva segmentu mají úroveň 0, neboť z uzlu *mám* vede hrana do kořene stromu. Druhý segment získá též úroveň 0, neboť jeho hranice s prvním segmentem odpovídá koordinačnímu výrazu. Čtvrtý segment získá úroveň vnoření 1,

[3] http://ufal.mff.cuni.cz/pdt2.0/

neboť z něj vede hrana do třetího segmentu s již stanovenou úrovní 0. Nalezené segmentační schéma věty tedy je (0001) (v tomto případě jde o správné segmentační schéma).



Obr. 2. Analytický strom věty (2) s vyznačenými segmenty.

3.2 Výchozí sada segmentačních pravidel pro text

Výchozí sada (heuristických) segmentačních pravidel byla publikována v [3]. Tato pravidla jsme zpřesnili a implementovali, abychom mohli porovnat jimi získaná segmentační schémata s výsledky segmentace na základě stromů PDT a s ruční anotací. Předpokládáme, že zpracovávaný text je již rozdělen na jednotlivé věty.

Při určování segmentačního schématu z prostého textu postupujeme vždy od začátku věty a chceme pro každý segment určit jeho úroveň vnoření.

Dále uvedená pravidla určují, jaká úroveň vnoření má být přiřazena prvnímu segmentu. Dále určují, jak se může měnit úroveň vnoření při překročení té které hranice.

Protože pravidla nedávají vždy jednoznačnou odpověď (například čárka může ukončovat jednu nebo více úrovní vnoření), namísto jedné úrovně vnoření každému segmentu přiřazujeme **interval** úrovní, ve kterých se může nacházet. Výsledné intervaly určují množinu segmentačních schémat.

Jednotlivé segmenty mohou být odděleny složenou hranicí, tedy posloupností několika elementárních hranic. V takovém případě jsou pravidla uplatňována postupně na jednotlivé elementární hranice. Proto pravidla neurčují, jaká bude úroveň vnoření následujícího segmentu, ale to, jak se změní globální průběžný ukazatel úrovně při zpracování elementární hranice. Segmentu je nakonec přiřazena taková úroveň, resp. takový interval úrovní, jaký byl nastaven po zpracování poslední elementární hranice před začátkem tohoto segmentu.

Pokud se podmínky dále uvedených pravidel odkazují na následující segment, odkazují se všechny na první segment ležící za složenou hranicí.

Začátek věty: Pokud první segment nemá přiřazen PP, bude umístěn na základní úrovni 0. Má-li PP, bude umístěn na úrovni 1.

Čárka: Pokud následující segment nemá PP, bude dolní mez intervalu u ukazatele úrovně zachována, horní mez bude nastavena na 0 (odpovídá konci libovolně mnoha vnořených klauzí). Má-li následující segment PP, bude ukazatel úrovně vnoření o 1 zvětšen (odpovídá začátku vnořené klauze či její části).

Otevírací závorka (libovolného druhu): Pokud následující segment nemá PP, bude úroveň vnoření o 1 zvýšena (začátek vsuvky). Má-li následující segment PP, bude úroveň zvýšena o 2 (vsuvka s hlouběji vnořenou strukturou).

Uzavírací závorka (libovolného druhu): Pokud jí předchází dosud neuzavřená otevírací závorka stejného druhu, bude úroveň vnoření (příp. interval možných úrovní) nastavena na úroveň platnou před zpracováním otevírací závorky, jinak se úroveň nezmění (toto omezení odpovídá třeba případům výčtů *a)... b)...*).

Koordinační spojka: Úroveň zůstává beze změny.

Dvojtečka: Pokud následující segment nemá PP, zůstane horní mez intervalu úrovně nezměněná (odpovídá koordinaci či apozici), dolní mez bude zvýšena o 1 (začátek vnořené klauze či její části). Má-li následující segment PP, bude horní mez zvýšena o 1 a dolní mez o 2 (hlouběji vnořená (část) klauze).

Otazník, vykřičník: Dolní mez u ukazatele úrovně bude snížena o 1, horní mez bude nastavena na 0 (konec libovolně mnoha vnořených klauzí).

Středník: Dolní mez u ukazatele úrovně bude zachována, horní mez bude nastavena na 0.

Svislítko, pomlčka, uvozovky: Úroveň zůstává beze změny.

Pokud v takto získaném schématu nemá žádný segment úroveň nulu, ‚posuneme' celé schéma o úroveň výše (posun -1).

Tato pravidla určí pro každou morfologicky analyzovanou vstupní větu množinu segmentačních schémat.

4 Vyhodnocování a rozbor výsledků

4.1 Testovací data a možné míry úspěšnosti

V předchozích oddílech jsme popsali základní experimenty s automatickým určováním segmentačních schémat z prostého textu a ze stromů PDT. Abychom mohli tato základní pravidla dále vyvíjet a zlepšovat, potřebovali jsme vytvořit testovací sadu vět se správně určenými segmentačními schématy.

Vybrali jsme proto z vývojovych dat PDT 2.0 (tzv. dtest data, 5 228 vět) věty, které obsahují alespoň pět segmentů (707 vět). Z těchto vět jsme pro každou desátou větu ručně určili segmentační schéma. Tím jsme získali 71 poměrně strukturně složitých vět s označenými segmentačními schématy.

Zdůrazněme zde, že tento výběr vět (průměrně 6,49 segmentu na větu) do značné míry zhoršuje naměřené výsledky oproti náhodnému výběru, neboť vylučuje věty z hlediska segmentace jednoduché (průměrný počet segmentů na větu v celých dtest datech činí 2,72).

Poznamenejme dále, že mnohé z testovacích vět jsou homonymní, mají více možných syntaktických stromů, v PDT je však zachycena vždy pouze jedna z možných struktur. Při určování segmentačního schématu jsme se drželi struktury, kterou zachytili anotátoři PDT. Každé větě tedy bylo přiřazeno jediné schéma (např. větě (1) bylo přiřazeno pouze segmentační schéma (01221)).

Chceme-li vyhodnotit úspěšnost navržených pravidel, nabízí se několik možných přístupů. Jako nejjednodušší vhodná míra se jeví míra počítající shodu úrovně vnoření jednotlivých segmentů, dále tuto základní míru značíme ρ.

Zkoumáme-li ovšem výsledky experimentů, zjišťujeme, že v řadě případů chyba při určování úrovně pro jeden segment má za následek chybně určené úrovně u dalších segmentů, přestože vztahy mezi jednotlivými segmenty jsou určeny správně. Například věta (3) má správné segmentační schéma (2233110), pomocí pravidel z PDT pak bylo navrženo schéma (1122000); přestože byly správně určeny skoro všechny vztahy mezi segmenty, úrovně se shodují u jediného segmentu (hranice opět podtrženy).

(3) *„ Když to odečtete od výplaty spolu se ztrátou při výměně slovenských korun za české a za pojištění , které se musí platit tam i u nás , nezbude manželovi z výplaty ani polovina ,“ zlobí se paní Krajčová .*

Tento nedostatek základní míry lze odstranit tím, že povolíme posun celého výsledného schématu tak, aby se shodovaly úrovně co největšího počtu segmentů (pro větu (3) je to posun hodnoceného schématu o +1, který nám dá shodu na šesti segmentech). Tuto míru značíme σ.

Protože nás primárně zajímají právě vztahy mezi segmenty, je vhodné uvažovat i míru (značíme δ), která měří správnost rozdílu úrovní dvou sousedních segmentů (např. schémata (101) a (211) mají stejné vztahy mezi prvním a druhým segmentem (rozdíl -1), ale různé vztahy mezi druhým a třetím segmentem).

Přestože hlavním cílem tohoto příspěvku není implementace konkrétní sady pravidel, ale vytvoření vhodného prostředí pro vývoj a testování těchto pravidel, podívejme se na úspěšnost dvou základních sad pravidel popsaných výše (jakkoliv mají tyto výsledky pouze informativní povahu danou nejen neúplností zatím navržených pravidel, ale i malým rozsahem testovacích dat).

4.2 Úspěšnost pravidel pro segmentační schémata ze syntaktických stromů

Základní sada pravidel pro získání segmentů z PDT určí pro každou vstupní větu právě jedno segmentační schéma. Při vyhodnocování úspěšnosti proto stačí *přesnost*, anglicky *accuracy*.[4] Úspěšnost těchto pravidel je shrnuta v tabulce 1.

accuracy	základní míra		míra s posunem	
# segmentů	# správně	ρ	# správně	σ
461	264	0,57	335	0,73

Tab. 1. Úspěšnost základní sady pravidel pro získání segmentů ze stromů PDT (při povoleném posunu ± 2).

Vyhodnocujeme-li rozdíl úrovní dvou sousedních segmentů, dosahují pravidla úspěšnost $\delta = 0,70$ (správně bylo určeno 274 ze 390 vztahů mezi segmenty).

Zmiňme zde tři hlavní problémy, které snižují úspěšnost výchozích pravidel pro automatické určování segmentačních schémat ze stromů PDT.

1. Větný člen, který tvoří samostatný segment, má přiřazenu úroveň vnoření o 1 vyšší než segment s jeho řídícím členem. Např. větám *Petr, který nikdy nelže, tentokrát zalhal.* či *Včera, kdy tak pršelo, přišli.* se správným segmentačním schématem (010) bude přiřazeno nesprávné segmentační schéma (120).
2. Zatím neřešena je speciální (ale poměrně častá) česká konstrukce, kdy za sebou následují dva podřadící výrazy (podtržené), jako např. *Nevěděl, že když jsem se probral k vědomí, zavolal jsem policii.*
3. Dalším rozšířeným jevem, na který je potřeba se soustředit, jsou koordinační (příp. apoziční) spojení více než dvou členů.

4.3 Úspěšnost pravidel pro segmentační schémata z textu

Vyhodnocování úspěšnosti při určování segmentačního schématu z prostého textu spočívá v počítání, zda pro jednotlivé segmenty výsledný interval úrovní obsahuje správnou úroveň vnoření toho kterého segmentu (zatím tedy měříme pouze tzv. *recall*), viz tabulku 4.3. (Vzhledem k nejednoznačně určeným vztahům mezi segmenty zde neuvádíme míru δ.)

recall	základní míra		míra s posunem	
# segmentů	# správně	ρ	# správně	σ
461	297	0,64	349	0,76

Tab. 2. Úspěšnost základní sady pravidel pro získání segmentů z prostého textu (povolené posouvání ± 2).

Průměrný počet segmentačních schémat na jednu větu pro naše testovací data je 2,17 (průměrná míra víceznačnosti na celém dtestu je 1,32).

[4] Míry pokrytí (*recall*) a přesnost (*precision*) se rovnají.

Zmiňme zde opět alespoň několik jevů, které výchozí sada pravidel nepostihuje adekvátně a které musí být předmětem bližší specifikace (ta však vyžaduje další podrobné lingvistické zkoumání).

1. Nejsou stanovena pravidla pro přímou a polopřímou řeč, např. ve všech nalezených segmentačních schématech věty (3) nebudou úrovně prvních čtyř segmentů dostatečně hluboko (nalezené úrovně (1122) místo správných (2233)).
2. Zatím není řešena koordinace několika klauzí s opakujícím se příznakem podřízenosti (např. věta *Jak účelně větrat, jak nepřetápět, jak spotřebu měnit a podle toho účtovat.* bude mít přiřazeno chybné schéma (0122) místo správného schématu (0000).)

Shrnutí

Segmentačního schéma zachycuje vztahy mezi jednotlivými segmenty věty, tedy lingvisticky motivovanými, přitom však snadno automaticky rozpoznatelnými úseky vět. Určuje tedy základní strukturu souvětí ještě před úplnou syntaktickou analýzou.

Příspěvek se soustřeďuje na popis rámce pro vývoj pravidel pro automatický odhad segmentačních schémat vět. Tento rámec umožňuje přesnou formulaci a zjemňování segmentačních pravidel. Dále jsme představili vhodné míry pro vyhodnocování úspěšnosti segmentační analýzy.

V této fázi vývoje byly implementovány dvě sady pravidel – pravidla pracující se syntaktickou strukturou Pražského závislostního korpusu a pravidla pro zpracování prostého textu. Porovnali jsme výsledky těchto pravidel s ručně anotovaným vzorkem vět z PDT, i když je zřejmé, že tyto výsledky mají pouze informativní charakter.

Provedené experimenty umožnily bližší specifikaci segmentačního schématu a pravidel pro ruční anotaci. Pro další výzkum segmentační analýzy a zpřesňování pravidel je potřeba vytvořit řádově větší testovací vzorek vět. Ukazuje se, že bez manuální anotace velké sady vět se v další fázi výzkumu neobejdeme.

Poděkování

Práce na tomto projektu je podporována Grantovou agenturou ČR, GAČR 405/08/0681 a částečně též programem Informatická společnost č. 1ET100300517.

Reference

1. Holan, T.: O složitosti Vesmíru. In Obdržálek, D., Štanclová, J., Plátek, M. (eds.) Malý informatický seminář MIS 2007. MatFyzPress, 2007. pp. 44-47
2. Kuboň, V.: Problems of Robust Parsing of Czech. Ph.D. Thesis, MFF UK, Prague, 2001
3. Kuboň, V., Lopatková, M., Plátek, M., Pognan, P.: A Linguistically-Based Segmentation of Complex Sentences. In Wilson, D.C., Sutcliffe, G.C.J. (eds.) Proceedings of FLAIRS Conference. AAAI Press, 2007. pp. 368-374
4. Jones B.E.M.: Exploiting the Role of Punctuation in Parsing Natural Text. In Proceedings of the COLING'94. Kyoto, 1994. pp. 421-425
5. Ohno, T., Matsubara S., Kashioka, H., Maruyama, T., Inagaki, Y.: Dependency Parsing of Japanese Spoken Monologue Based on Clause Boundaries In Proceedings of COLING and ACL. ACL, 2006. pp. 169-176
6. Hajič, J.: Disambiguation of Rich Inflection (Computational Morphology of Czech). UK, Nakladatelství Karolinum, Praha, 2004
7. Spoustová, D., Hajič, J., Votrubec, J., Krbec, P., Květoň, P.: The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In Proceedings of Balto-Slavonic NLP Workshop. ACL, Prague, 2007. pp. 67–74
8. Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M.: Prague Dependency Treebank 2.0. LDC, Philadelphia. 2006

ITAT'08

Information Technology – Applications and Theory

WORK IN PROGRESS

Využití moderních přístupů pro detekci plagiátů

Zdeněk Češka

Katedra informatiky a výpočetní techniky, Fakulta aplikovaných věd, Západočeská univerzita v Plzni
Univerzitní 22, 306 14 Plzeň, Česká republika
zceska@kiv.zcu.cz

Abstrakt. *Plagiátorství je v současnosti nejvíce skloňovaným pojmem, se kterým se můžeme setkat v každé oblasti lidské tvůrčí práce. Školství je jednou z důležitých oblastí, kde je nutné tomuto zamezit. V tomto článku se zabýváme moderními přístupy pro detekci plagiátů textových dokumentů. Naše metoda využívá normalizaci textu a latentní sémantickou analýzu pro nalezení skrytých vztahů mezi dokumenty. Dále uvádíme předběžné experimenty provedené na testovacím korpusu, který obsahuje 950 textových dokumentů o politice. Předběžné experimenty naznačují výhodnost naší metody a zlepšení výsledků oproti ostatním přístupům. V závěru článku diskutujeme využití WordNet tezauru pro zlepšení přesnosti současných metod a možnosti identifikace plagiátů, které byly přeloženy do jiných jazyků.*

1 Úvod

Plagiátorství je současným problémem, se kterým se potýkáme v každé oblasti tvůrčí lidské práce. Jedním z možných řešení, které se často prosazuje, jsou nejrůznější ochrany zabraňující kopírování digitálních médií. Ačkoli takovýchto ochran existuje nepřeberné množství, vždy se podaří nalézt nějakou slabinu a patřičnou ochranu deaktivovat. Internet můžeme považovat za zvláštní případ média, kde jsou jakékoli informace volně dostupné. Na Internetu lze bez problemů nalézt obsah většiny CD, DVD a dalších médií v nechráněné podobě, díky čemuž tyto ochrany pozbývají smysl.

Cílem není chránit informace, ale vyhledávat plagiátory a patřičně je trestat. Hlavní výhoda tohoto přístupu spočívá v psychologii, kdy každý plagiátor může být odhalen, porovná-li se jeho práce s databází již existujících děl. Školství je jednou z oblastí, kde kopírování cizích prací velmi škodí a brání tak přirozené tvořivosti studentů.

Clough [2] a Maurer [7] provedli malé srovnání aktuálního stavu metod pro detekci plagiátů mezi textovými dokumenty. V tomto článku jdeme hlouběji a popisujeme moderní metodu pro detekci plagiátů s využitím latentní sémantické analýzy (Latent Semantic Analysis – LSA) spolu s normalizací textu pro odhalování skrytých sémantických asociací mezi frázemi. Zvláštním rysem naší metody je zpracování celého korpusu najednou. Při tomto zpracování se využívá globální statistika všech obsažených dokumentů a zlepšuje se přesnost detekce plagiátů.

Další text v článku je organizován tímto způsobem. Sekce 2 popisuje současný stav v oblasti detekce plagiátů. Sekce 3 navrhuje metodu založenou na LSA. Porovnání naší metody s ostatními je uvedeno v Sekci 4. Sekce 5 popisuje budoucí práce na naší metodě a konečně Sekce 6 je souhrnem našich dosažených výsledků.

2 Současné metody

Metody pro detekci plagiátů lze rozdělit na metody pro zpracování psaného textu a metody pro zpracování zdrojových kódů. Detekce plagiátů zdrojových kódů je v současné době již poměrně vyřešená, což způsobuje především pevná struktura kódu. V tomto článku se budeme nadále zabývat psaným textem, a to z důvodu jeho uplatnění ve školství na nejrůznější semestrální, bakalářské a diplomové práce. Tab. 1 pak prezentuje rozdělení metod pro detekci textových plagiátů dle složitosti použitého algoritmu a počtu dokumentů, které daná metoda zpracovává najednou. Toto rozdělení bylo původně publikováno Lancasterem [3].

Typ rozdělení		Popis
Složitost použité metody	Povrchní	Metrika je počítána bez jakékoli znalosti lingvistických pravidel nebo struktury dokumentů
	Strukturní	Metrika je počítána s částečným porozuměním dokumentů
Počet dokumentů které, se zpracovávají u dané metody	Jednotlivá	Pro výpočet této metriky se zpracovává pouze jeden dokument, tj. dvě jednotlivé metriky mohou být využity pro výpočet párové podobnosti
	Párová	Dva dokumenty se zpracovávají současně pro výpočet metriky
	Multidimenzionální	*M* dokumentů se zpracovává společně pro výpočet metriky
	Korpální	Všechny dokumenty obsažené v korpusu se zpracovávají společně pro výpočet metriky

Tab. 1. Rozdělení metod pro detekci plagiátů textových dokumentů.

Jedním z nejpopulárnějších systémů je SCAM [10] založený na modelu relativních frekvencí slov, tzv. RFM modelu. Tato metoda může být klasifikována jako Povrchní, Párová. Podobně lze zařadit systém „Detection of Duplicate Defect Reports" [8], který pracuje na principu vektorového modelu, tzv. VSM. Ačkoli systém Ferret [5] využívá slovních trigramů pro nalezení překryvu textu mezi dvěma dokumenty, jedná se stále o Povrchní, Párovou metodu. Důvodem tohoto zařazení je porovnávání trigramů bez hlubšího porozumení souvislostí uvnitř textu.

V následující textu se zabýváme metodou, která je založana na LSA. Tuto metodu lze klasifikovat jako Strukturální a Korpální z důvodu sofistikovaného předzpracování textu a jeho následné hlubší analýzy.

3 Detekce plagiátů s využitím LSA

Námi navrhovaná metoda využívá LSA [4] pro odvození skrytých sémantických asociací mezi frázemi obsaženými v textu. Každá fráze je v našem případě reprezentována slovním N-gramem, který si lze představit jako posloupnost n slov následujících bezprostředně za sebou. V dalším textu popisujeme jednotlivé kroky procesu zpracovávající textové dokumenty.

3.1 Předzpracování textu

Předzpracování je jedním z klíčových kroků pro dosažení kvalitních výsledků u úloh zabývajících se zpracováním přirozeného jazyka (Natural Language Processing – NLP). V našem případě využíváme techniky pro mazání stop-slov a lematizaci. Mazání stop-slov je základní NLP technika, která odstraňuje všechna bezvýznamná slova v závislosti na definovaném slovníku. Lematizace [11] je následný proces pro získání základního tvaru slova, takzvaného lemmatu.

3.2 Extrakce frází

V dalším kroce extrahujeme fráze (ve smyslu N-gramů) předem zvolené délky z předzpracovaného textu. V našich předběžných experimentech jsme se zaměřili na N-gramy délky 1 až 5. N-gramy délky 1 jsou ve skutečnosti pouze jednotlivá slova a používáme je pro srovnání s metodami VSM a RFM.

3.3 Analýza a redukce frází

Čím delší fráze extrahujeme, tím vzniká větší množství unikátních frází, které musí být porovnávány napříč všemi dokumenty. Z tohoto důvodu velké množství frazí neuměrně zvyšuje časové požadavky na výpočet při aplikaci LSA. Pro redukci frází na přijatelnou úroveň jsme vytvořili filter založený na počtu dokumentů, ve kterých se daná fráze vyskytuje, tzv. DF filter. V závislosti na tomto filtru určujeme, zda je daná fráze důležitá či ne. Fráze, které se nacházejí pouze v jednom dokumentu jsou odstraněny okamžitě, protože nemohou být plagiovány v ostatních dokumentech. V dalším kroce odstraňujeme fráze, které se vyskytují ve více než $\mu + \sigma$ dokumentech, kde μ je střední hodnota počtu dokumentů, ve kterých se daná fráze nachází a σ je směrodatná odchylka od střední hodnoty. Tento krok odstraňuje všechny velmi často se opakující fráze, které lze považovat za bezvýznamné.

Délka fráze	Počet původních frází	Počet frází po redukci	Průměrný výskyt stejné fráze
1	30550	15343	7.45
2	128449	28206	1.76
3	169093	23337	1.34
5	189621	18281	1.18
7	195999	15549	1.13
9	199421	13536	1.10

Tab. 2. Počet frází před a po aplikaci DF filtru. Experiment byl proveden na vzorku 1000 zpráv standardního ČTK korpusu.

Tab. 2 zobrazuje počty fází před a po aplikaci DF filtru. DF filter významně ovlivňuje delší fráze, kde se s rostoucí délkou výrazně zvyšuje redukční poměr. Důvodem jsou především dlouhé fráze, které se vyskytují pouze v jednom dokumentu. V případě frází délky 5 lze dosáhnout až 10-ti násobného redukčního poměru.

3.4 Vytvoření zjednodušeného modelu dokumentů

Dále vytvoříme zjednodušený model vztahů mezi frázemi a dokumenty, který může být popsán maticí A. Nechť A je n x m obdélníková matice složená z n vektorů $[A_1, A_2, \ldots, A_n]$, kde vektor A_i představuje fráze obsažené v dokumentu i. Vektor A_i se skládá z m prvků $a_{i,j}$, kde každý prvek představuje váženou frekvenci výskytu fráze j v dokumentu i, jak naznačuje rovnice (1). Tato rovnice je modifikací standardního TF-IDF váhování [9].

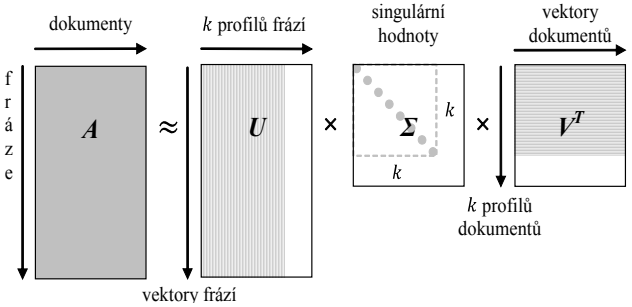
$$a_{i,j} = \begin{cases} \frac{1}{2} + \frac{PF_{i,j} \cdot \log\left(\frac{|n|}{DF_j}\right)}{2 \cdot \max_i\left(PF_{i,j}\right) \cdot \log\left(|n|\right)} & \text{jestliže se fráze } j \text{ nachází v dokumentu } i \\ 0 & \text{jinak} \end{cases} \quad (1)$$

$PF_{i,j}$ představuje frekvenci výskytu fráze j v dokument i, DF_j označuje počet dokumentů, ve kterých se nachází fráze j a $|n|$ je celkový počet zkoumaných dokumentů. Rozdíl oproti TF-IDF spočívá v IDF normalizaci tak, aby $a_{i,j} \in <0.5,1>$. V případě, že fráze j se nenachází v dokumentu i, $a_{i,j} = 0$. Tento způsob váhování dosahuje nejlepších možných výsledků v dalším kroce, který se zabývá dekompozicí matic.

3.5 Latentní sémantická analýza

V tomto kroce se odvozují skryté sémantické asociace mezi frázemi, které jsou obsaženy ve zkoumaných dokumentech. Pro odhalení těchto vztahů využíváme metodu singulární dekompozice (Singular Value Decomposition – SVD), která rozkládá matici A na tři nezávislé matice U, Σ a V^T. Všechny tyto matice mohou být dekomponovány s redukovaným skrytým prostorem k pro získání nejlepší k-té aproximace A, viz [1]. Toho docílíme přepsáním singulární hodnot $\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_m$ číslem 0, kde $1 \leq k \leq m$. V našem případě matice U je n x k sloupcově ortonormální, jejíž sloupečky představují singulární vektory frází. Σ je k x k diagonální matice bez záporných a nulových hodnot, které představují singulární hodnoty. A konečně matice V^T je k x m řádkově ortonormální, jejíž řádky představují singulární vektory dokumentů.

Obr. 1 zobrazuje dekompozici matice A mnohem detailněji. Ve výsledku matice V^T obsahuje jednotlivé profily dokumentů a je základním stavebním prvkem pro výpočet podobností mezi dokumenty.



Obr. 1. Dekompozice matice frází zastoupených v dokumentech prostřednictvím metody SVD.

3.6 Normalizace podobností mezi dokumenty

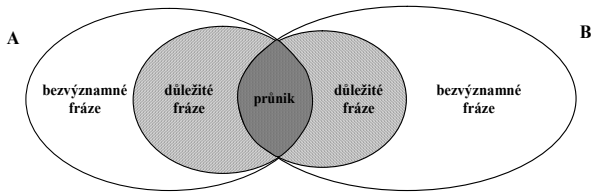
V posledním kroce počítáme podobnosti mezi jednotlivými páry dokumentů. Nejdříve je nutné přenánásobit matici V^T singulárními hodnotami pro získání správného rozměru jednotlivých prvků v profilech dokumentů, což popisuje rovnice (2).

$$B = \Sigma \times V^T \tag{2}$$

Korelační matice podobností mezi dokumenty se vypočte dle rovnice (3), kde sloupečky matice B musí být normalizovány. Výsledná matice sim_{SVD} je symetrická, kde pro každý pár dokumentů je obsažena jejich procentuální podobnost.

$$sim_{SVD} = \left\|B\right\|^T \times \left\|B\right\| \tag{3}$$

Ačkoli se může zdát, že výpočet je v současnosti hotov, je nutné se zamyslet nad vlivem DF filtru pro redukci fází. Obr. 2 zachycuje situaci, kde část frází je označena jako bezvýznamná a nejsou tím pádem uvažovány během výpočtu. Následný výpočet probíhá nad menší množinou, tudíž sim_{SVD} dosahuje nižšího procentuelního ohodnocení, které nemůže odpovídat realitě.



Obr. 2. Průnik dvou množin frází.

Rovnice (4) modifikuje sim_{SVD} pro získání správného ohodnocení podobnosti mezi dokumenty R a S. Podobnosti jsou váženy poměrem mezi počtem původních frází $|ph_{orig}|$ a počtem frází po redukci $|ph_{red}|$.

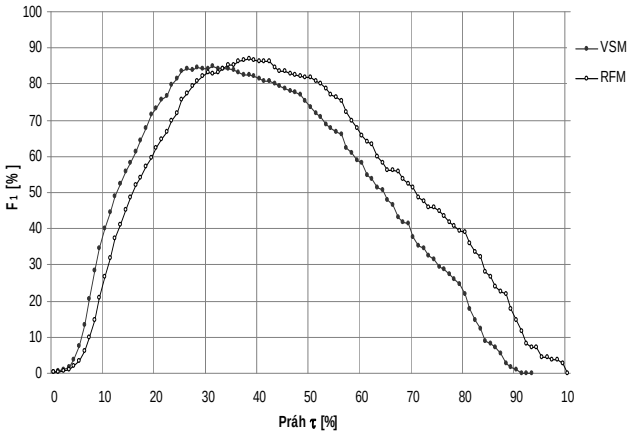
$$sim(R,S) = sim_{SVD}(R,S) \cdot \sqrt{\frac{|ph_{orig}(R)|}{|ph_{red}(R)|} \cdot \frac{|ph_{orig}(S)|}{|ph_{red}(S)|}} \tag{4}$$

4 Experimenty

Pro naše počáteční experimenty jsme shromáždili kolekci 150 plagiovaných dokumentů v českém jazyce. Tato kolekce byla vytvořena manuálně studenty. Ze standardního ČTK korpusu jsme náhodně vybrali 300 článků o politice a použili je jako základ pro vytvoření plagiovaných dokumentů. Výsledný korpus čítající 950 dokumentů jsme namíchali ze 150 plagiovaných dokumentů, 300 původních článků a 500 dalších náhodně vybraných článků o politice.

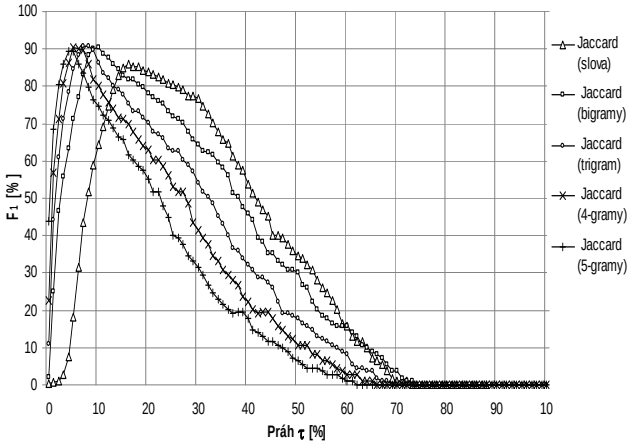
Obr. 3 zobrazuje závislost míry F_1 na prahu τ pro metodu VSM [8] a RFM [10]. Obě křivky jsou poměrně široké, tudíž není problém stanovit správný práh τ, kterým rozhodujeme, zda je daný dokument plagiát či ne. Nicméně dosahované skóre pro F_1 je nižší než u ostatních metod.

Následující metoda využívá Jaccard-Tanimoto koeficient [6], který byl použit v systému Ferret [5]. Obr. 4 zachycuje rozličné závislostní křivky při použití jednotlivých slov, bigramů, trigramů, 4-gramů a 5-gramů. Při porovnání s předchozím grafem jsou všechny křivky výrazně užší než VSM a RFM. Jak lze vidět z grafu, zvětšující se délka N-gramu (v našem případě fráze) snižuje práh τ potřebný pro dosažení nejlepších možných výsledků.



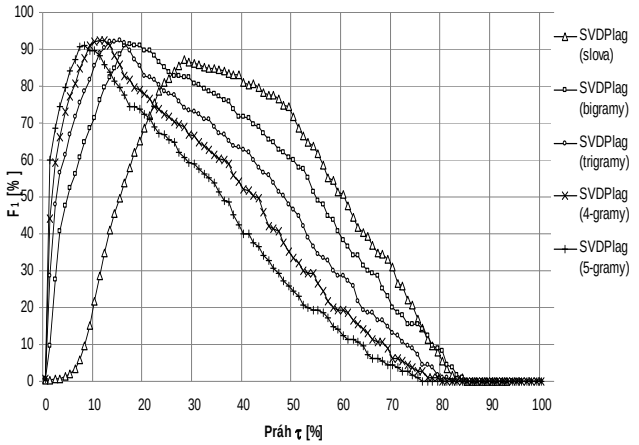
Obr. 3. Závislost míry F_1 na prahu τ při detekci plagiátů metodami VSM a RFM.

Obr. 5 zobrazuje závislostní křivky míry F_1 na prahu τ pro naší metodu SVDPlag, která využívá LSA. Oproti systému Ferret jsou křivky širší, a tudíž je snazší určit správný práh. Naše experimentální metoda SVDPlag dosahuje mnohem lepších hodnot F_1 oproti ostatním metodám.



Obr. 4. Závislost míry F_1 na prahu τ při detekci plagiátů Jaccard-Tanimoto koeficientem s využitím slov, bigramů, trigramů, 4-gramů a 5-gramů.

Tab. 3 je sumarizací nejlepších získaných hodnot pro F_1. Jak je vidět, naše experimentální metoda dosahuje nejlepších výsledků pro fráze skládající se ze čtyř následujících slov (4-gram). Systém Ferret dosahuje nejlepších výsledků pro fráze o třech slovech, kdy míra F_1 je 90,82% oproti 92,57% v porovnání s naším systémem. V případě trigramů a 4-gramů dosahuje naše metoda velmi obdobných výsledků. Podobně se chová i metoda založená na Jaccard-Tanimoto koeficientu pro bigramy, trigramy a 4-gramy. Přestože jsou výsledky okolo sekvence tří slov velmi podobné, doporučujeme raději volit 4-gramy. Delší fráze lépe separují nesouvisející dokumenty a redukují šum nacházející se v hlavičkách a patičkách dokumentů.



Obr. 5. Závislost míry F_1 na prahu τ při detekci plagiátů metodou SVDPlag s využitím slov, bigramů, trigramů, 4-gramů a 5-gramů.

Metoda	Práh τ	F_1
VSM	30%	84,97%
RFM	37%	87,03%
Jaccard (slova)	16%	85,84%
Jaccard (bigramy)	10%	90,56%
Jaccard (trigramy)	8%	90,82%
Jaccard (4-gramy)	6%	90,53%
Jaccard (5-gramy)	4%	89,36%
SVDPlag (slova)	28%	87,31%
SVDPlag (bigramy)	17%	91,36%
SVDPlag (trigramy)	15%	92,48%
SVDPlag (4-gramy)	12%	92,57%
SVDPlag (5-gramy)	8%	91,03%

Tab. 3. Nejlepší dosažené výsledky pro míru F_1.

5 Budoucí práce

Náš další výzkum se ubírá směrem k využití WordNet tezauru. V nejjednodušší variantě plánujeme nahrazovat slova jejich multijazykovými indexy (InterLingual Index - ILI), kde každý index označuje skupinu stejných synonym (tzv. synset). Daný index je společný pro různé jazyky, tudíž bude možné v budoucnu zahrnout i vícejazykovou podporu.

WordNet tezaurus má nicméně daleko širší uplatnění. Jednotlivé synsety jsou provázány nejrůznějšími odkazy, představující například hyperonyma, hyponyma, antonyma, odvozeniny atd. Z našeho pohledu jsou nejzajímavější hyperonymické odkazy, díky kterým lze nalézt obecnější tvary slov. Tyto odkazy vlastně tvoří stromovou strukturu slova od nejkonkrétnějšího významu po nejobecnější. Představme si slova „kočka“ a „pes“, jejichž společné hyperonymum je slovo „zvíře“. Struktura WordNetu je samozřejmě mnohem detailnější a ke slovu „zvíře“ se dostaneme až po několika posunech v hierarchii, kdy jdeme například přes slova „savec“ a „obratlovec“.

Zmíněným postupem můžeme velice zjednodušit slovní zásobu a zobecnit veškerá slova na libovolnou požadovanou úroveň. Jediným problémem, se kterým se potýkáme, je volba vhodné úrovně. Podstatná slova mívají obvykle hloubku stromu 5 až 8. Naproti tomu slovesa 1 až 3. Z tohoto důvodu je nutné zacházet s každým slovním druhem zvlášť, což je otázkou budoucnosti.

6 Závěr

V tomto článku jsme představili metodu pro detekci plagiátů využívající LSA. Tato metoda na základě asociací mezi frázemi odhaluje podobnost mezi dokumenty. Naší metodu lze klasifikovat jako Strukturální a Korpální, viz Tab. 1.

Z našich experimentů provedených na korpusu čítajícím 950 dokumentů o politice je zřejmé, že SVDPlag překonává ostatní metody pro detekci plagiátů. Nejlepší výsledky jsme získali pro 4-gramy a 12% práh, kdy za těchto podmínek míra F_1 dosahuje 92,57%.

Jedním z klíčových faktorů naší budoucí práce bude využití WordNet tezauru pro pokročilou normalizaci slov. Dále plánujeme návrhnout složitější model textových dokumentů, který by podchytil vztahy slov obsažených uvnitř frází. Posledním cílem je rozšíření stávajícího korpusu o více plagiovaných dokumentů a přidání dalších témat.

Poděkování

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

Reference

[1] M. Berry, S. Dumais, G. O'Brein, „Using Linear Algebra for Intelligent Information Retrieval“, *SIAM Review*, vol. 37 issue 4, pp. 573-595, Society for Industrial and Applied Mathematics, Philadelphia, USA, 1995. ISSN 0036-1445.

[2] P. Clough, „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.

[3] T. Lancaster, F. Culwin, „Classification of plagiarism detection engines“, *E-journal ITALICS*, vol. 4 issue 2, 2005. ISSN 1473-7507.

[4] T. Landauer, P. Foltz, D. Laham, „An introduction to Latent Semantic Analysis“, *Discourse Processes*, vol. 25, pp. 259-284, 1998.

[5] P. Lane, C. Lyon, J. Malcolm, „Demonstration of the ferret plagiarism detektor“, *Proceedings of the 2nd International Plagiarism Conference*, Newcastle, 2006.

[6] C. Manning, H. Schutze, „Foundation of statistical natural language processing“, *The MIT Press*, Massachusetts Institute of Technology, Cambridge MA, 1999.

[7] H. Maurer, F. Kappe, B. Zaka, „Plagiarism – A survey“, *Journal of Universal Computer Science*, vol. 12 issue 8, pp. 1050-1084, 2006.

[8] P. Runeson, M. Alexanderson, O. Nyholm, „Detection of duplicate defect reports using natural language processing“, *Proceedings of the IEEE 29th International Conference on Software Engineering*, pp. 499-510, 2007.

[9] G. Salton, C. Buckley, „Term-Weighting Approaches in Automatic Retrieval“, *Journal of Information Processing and Management*, vol. 24 issue 5, pp. 513-523, 1988.

[10] N. Shivakumar, H. Garcia-Molina, „SCAM: A copy detection mechanism for digital documents“, *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries*, Austin, 1995.

[11] M. Toman, R. Tesar, K. Jezek, „Influence of word normalization on text classification“, *Proceedings of the 1st International Conference on Multidisciplinary Information Sciences & Technologies*, vol. 2, pp. 354-358, Merida, Spain, 2006. ISBN 84-611-3105-3.

Knowledge-based programming environments

Peter Drahoš and Peter Kapec

Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava 4
drahos@fiit.stuba.sk, kapec@fiit.stuba.sk

Abstract. *Programming environments are usually closely tied to software representation and visualization. Most development environments depend on specific languages and use file-based project organization. Using knowledge-based representation of software artifacts and visual programming techniques we are able to provide relevant information and help in an interactive way. This approach offers many advantages over the traditional programming environments which are more machine and language oriented. Instead a more natural system utilizing relations between software artifacts is presented.*

1 Introduction

Software, opposed to other engineering products, is intangible. Software visualization aims to help us with the intangible software to make it "more" tangible. It seems that software comprehension is easier when the software is graphically represented. For illustration control-flow diagrams are popular help for program understanding. Usage of software visualization has long history and we can find early attempts in the beginnings of computer science. Early in the 1940-ties Goldstein and Neumann presented possible advantages of using flow-charts [5]. Later in the 1970-ties Nassi-Schneiderman Diagrams were often used as an alternative to flow-charts. When graphical workstations became affordable in the 1980-ties, new and more robust visualization systems could be developed. Utilizing the third dimension, with the hope of "gaining" more space for visualization, was the aim of many projects in 1990-ties. In past years more sophisticated navigation and viewing methods were developed. Although there was a big progress in the software visualization field, open problems still exist and have to be solved.

1.1 Software visualization

In the past two decades many software visualization prototypes and visual programming environments were developed. However only few managed to move from research projects to systems usable in practice. New ideas, especially those improving productivity, should become part of daily practice. This step is a major problem and is directly connected to the evaluation of such visualization prototypes. When evaluation of visualization systems is not done in practice and their efficiency is not shown, the question "Why such visualization prototypes are actually developed?" may appear. This question and results of Koschke's study lead to following challenges in the software visualization field[8]:

- visualization of large datasets and dynamic aspects.
- navigation between multiple views.
- automatic selection of visualization techniques.
- **semantic visualization.**
- integration with other tools.

Our work concentrates primary on semantic visualization utilizing existing approaches in graph visualization and selection.

1.2 Hybrid environments

Modern software products often consist of many more or less independent parts such as libraries, plug-in modules, scripts, documentation etc. Many of these are implemented by third party developers in different programming languages or formats. With a vast variety of components being reused and combined in bigger projects we face new challenges in project management systems. Modern projects are developed in various programming IDEs and designed using various methods but all of these are often designed for homogeneous programming in certain language. A heterogeneous, visual and semantic approach to project management is needed.

In our work we aim to create an Visual Programming Environment (VPE) based on semantic representation of software and existing languages. Having a semantic representation of various software artifacts we are able to preserve relations between various aspects of the developed code such as documentation, versioning or even software management. Combining all these aspects of programming into one graph-based visualization system should improve productivity and comprehension of the development process.

We call this approach hybrid programming environment because it does not distinguish between the languages, methods and artifacts used in the software project. Such programming environment would be very beneficial in the age of software archeology that we are slowly moving forward to.

2 Software representation

Under knowledge we often understand information placed into a meaningful context, combined with experience, interpreted etc. Ontologies allow representing and storing

knowledge about a certain domain in a declarative way. One of benefits of ontologies is that they can be interpreted by humans and also computers. Of course presenting ontologies in a navigable form is important and often visualizations based on graphs are used.

Knowledge discovery has many application areas. One of them, software mining, deals with understanding software artifacts. The term software covers many different things ranging from source code, documentation, source code revisions, diagrams, graphical user interfaces, data and algorithms and many others. It is difficult to deal with such different software artifacts, especially in large projects. Software mining and software visualization aim to simplify extraction of previously unknown and potentially useful information from software.

2.1 Topic maps

An interesting idea is to store knowledge about different kinds of software artifacts using one representation. Topic maps, a standard for ontology representation, were developed to describe knowledge structures and associations between information resources. Topic maps were inspired by several domains – they borrow ideas from traditional indexing, library science, knowledge representation and artificial intelligence. Topic maps are based on three main concepts: topics, associations and occurrences. Topics represent subjects of our world. They can be anything we can talk about. Associations define relations between two or more topics. Occurrences link topics to one or more information resources that are somehow related to the topic. Features of topic maps allow representing knowledge about software artifacts.

For example topics can be used to represent information about all kinds of software artifacts and associations can represent different relations between them.

3 Artifact storage

Persistent storage and query mechanism are important issues for ontologies to be useful. Storages should be considered as repositories that allow dynamic updating through appropriate APIs or query languages. Currently three approaches exist for persistent storage.

Data centric persistency approach uses concepts represented in ontology to define structures in a relational database system. The drawback of this approach is that it is not well suited for dynamic changes in class hierarchies in ontologies.

Structure centric approach maps the data model concepts to structures in a relational database system. Thanks to the well-defined and static number of model concepts, e.g. in Topic Maps, structures in RDBMS can be created straightforwardly and with scalability optimizations [12], [9].

Finally the XML-based approach offers serialized versions of ontologies. The XML serialization imposes strict constraints on the structure, thus native XML databases also seem to be suitable for storage [3]. XML databases focus primarily on XML documents, which of course can contain ontologies. However this approach has the same shortcomings as the data centric approach and XML databases offer no real advantages against relational databases when we consider the underlying ontology data model.

The GXL format[7], although not originating in knowledge representation field, is a XLM-based storage developed for exchanging graph-based data. The GXL format is capable of storing typed, attributed, directed, ordered, hierarchical graphs and hypergraphs. It would be an interesting research direction to define ontology serialization formats based on GXL.

Appropriate querying methods are at least as important as knowledge storage. A simple solution is to utilize SQL or Xquery languages, but they are only suitable for data with known schema, which is in contrary to the dynamic nature of ontologies. Semantic query languages are needed as they are capable to deal with graph representation of ontologies

3.1 Hypergraphs

Graphs, or even hypergraphs, can be used to formally define two popular standards for ontology representation, Topic Maps and RDF/OWL. A formal definition of a hypergraph follows:

Definition 1. *Let $V = \{v_1, ..., v_n\}$ be a finite set, whose members are called nodes. A hypergraph on V is pair $H = (V, \epsilon)$, where ϵ is a family $(E_i)_{i \in I}$ of subsets of V. The members of ϵ are called edges.*

An alternative hypergraph definition was defined, specially developed for the formal definition of Topic maps based on this hypergraph defnition, and is shown in definition 2, taken from[1].

Definition 2. *A hypergraph is a five-tuple $H = (V, \lambda_V, E, \lambda_E, I)$ where V, E, I are disjoint finite sets and we call V the vertex set of H, E the edge set of H, I the incidence set of H and $\lambda_V : V \to P(I)$ is a mapping that satisfies following conditions:*

$$\forall v \neq v' \;\; \lambda_V(v) \cap \lambda_V(v') = 0 \;\; \cup_{v \in V} \lambda_V(v) = I \quad (1)$$

and $\lambda_E : E \to P(I)$ is mapping that satisfies the following conditions:

$$\forall e \neq e' \;\; \lambda_E(e) \cap \lambda_E(e') = 0 \;\; \cup_{e \in E} \lambda_E(e) = I \quad (2)$$

This hypergraph definition allows easy mapping to Topic maps: the hypergraph vertices map to Topic map's

topics, edges map to associations and incidences to roles. Other Topic map properties can be achieved by defining an additional mapping, see definition 3, taken from [1].

Definition 3. *A topic map τ is a couple $\tau = (H, \Theta)$ where $H = (T, \lambda_T, A, \lambda_A, I)$ is homogeneous hypergraph of τ and $\Theta : T \rightarrow P(T \cup A \cup I)$ is a mapping from topic set of τ to the set of subsets of elements of τ, that we call the covering of the topic map, with the constraint that:*

$$\forall t \in T \;\; H[t] \cap \Theta(t) = 0 \tag{3}$$

From the definition mentioned above it can be seen, that the hypergraph formalism is powerful enough to describe features of Topic Maps. This hypergraph approach can be also applied to RDF Graphs.

Hypergraph representation of software seems to be more natural than topic maps as it is closer to visualization based on graphs. Currenty we are studiing methods of storing hypergraph representations in conventional databases based on previous works in the field [4], [6].

Figure 1 shows how a simple function could be represented as a hypergraph. Obtaining the hypergraph representation can be achieved by parsing existing source code. It is possible for the syntactic and semantic rules of the source language to be stored in the same hypergraph [2].

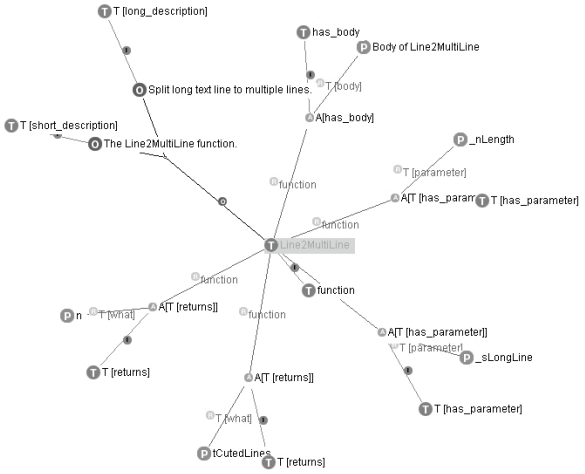


Fig. 1. Example hypergraph representation of the simple function and relation to its documentation.

Additionally, hyper-graph representation can cover other code artifacts such as classes, objects, abstract data types etc. Using relations we could describe the whole software development process from UML design through implementation, project management up to task such as debugging, deployment and even collaborative development.

4 Visual programming environment

Software visualization frequently uses graph visualizations and often utilizes 3D visualizations. Such visualizations are well suitable for software comprehension and may be used for quality assessment of large software projects [10]. However, for programming tasks like modifying or writing new source code parts, we need access to concrete language constructs. Therefore it is essential that the programming environment provides a text editor that is integrated directly into graph visualization. This increases the familiarity of the environment for most programmers not used to purely visual programming languages.

As hypergraphs can be easily transformed into bipartite graphs using the following definitions, many well known graph visualization techniques can be directly applied. Using this transformation the hyperedges of a hypergraph are transformed to nodes of a bipartite graph, thus the visualization technique does not know it is rendering a hypergraph. Rendering hypergraphs directly would however require some modifications to layout algorithms due the nature of hyperedges, which allow connecting multiple vertices.

Following definitions illustrate the mathematical transformation of hypergraph into a bipartite graph.

Definition 4. *Let $H = (V, \epsilon)$ be a hypergraph with $m = |\epsilon|$ edges and $n = |V|$ nodes. The edge-node incidence matrix of H is:*

$$M_H \in M_{m \times n}(\{0, 1\}) \tag{4}$$

and defined as:

$$m_{i,j} = \begin{cases} 1 \; if v_j \in E_i \\ 0 \; else \end{cases} \tag{5}$$

Using the hypergraph's incidence matrix we can construct a bipartite graph by following definition:

Definition 5. *For a hypergraph $H = (V, \epsilon)$ with an incidence matrix M_H the bipartite incidence graph*

$$B_H = (N_V \cup N_\epsilon, E) \tag{6}$$

is defined as follows:

$$E = \{\{m_i, n_j\} : m_i \in N_\epsilon, n_j \in N_v, and \, m_{i,j} = 1\}$$
$$N_\epsilon = \{m_i : E_i \in \epsilon\} \;\; N_V = \{n_j : v_j \in V\} \tag{7}$$

This transformation is the core idea behind using hypergraphs for software prepresentation.

4.1 Context visualization

The most cited principle in information visualization, known as visual information seeking mantra is "Over-view first, zoom and filter, details on demand" [11]. This principle can be directly applied to hypergraphs where zoom and

filter operations are graph projections, details are stored in nodes and edges.

A common problem of visual programming environments is the problem of small screen space. Using hypergraph projections allows us to filter-out unnecessary parts and to display only relevant information. We plan to implement a simple method for displaying contextual information about focused code fragments with dynamic weighting of spring forces, based on relation relevancy in hypergraph representation. Hyperedge relevancy below certain treshold will not be displayed in order to conserve screen space and reduce the complexity of the displayed graph.

Figure 2 shows a concept of the described visual programming environment that displays simple software example from Figure 1. The central billboard contains generic text editor while relevant contextual information and its relations are displayed using darker billboards.

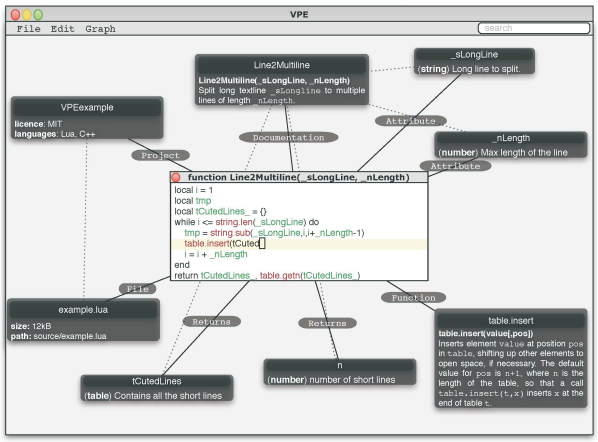


Fig. 2. Concept of the VPE interface showing the simple function from fig.1

5 Conclusions

The presented work in this paper is not necessarily new, many of the presented concepts have been around in visual programming approaches for some time. However we try to apply the benefits of hypergraph representation of software and visual aid to heterogenous programming environments based on existing textual languages rather than develop new language.

The presented work shows some of the storage and software representation ideas we are currently investigating. Our goal is however to create an Visual Programming Environment, which was originally a computer visualization problem. By combining existing approaches in knowledge representation with graph-visualization techniques from computer graphics we are working towards an interesting hybrid combination.

Acknowledgements

This work was partially supported by the (VEGA) grant 1/3103/06: Information infrastructure for information processing in distributed environments.

References

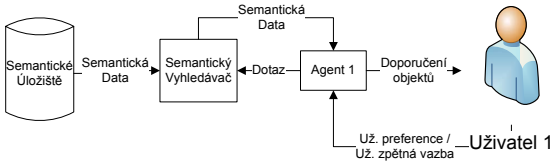
1. Auillans, et al., A formal model for topic maps. In ISWC ’02: Proceedings of the First International Semantic Web Conference on The Semantic Web, Springer-Verlag, pp. 69-83, 2002
2. R. Bardohl, M. Minas, A. Schurr, G. Taentzer., Application of graph transformation to visual languages, 1999.
3. R. Bourret., Xml and databases, 2003. http://rpbourret.com/xml/XMLAndDatabases.htm.
4. A. Gutiérrez, P. Pucheral, H. Steffen, and J.M. Thévenin., Database Graph Views: A Practical Model to Manage Persistent Graphs. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, Proceedings of VLDB’94, pages 391–402. Morgan Kaufmann, 1994
5. H. H. Goldstein, J. von Neumann., Planning and Coding Problems of an Electronic Computing Instrument. In: Taub, A.H., von Neumann, J., Collected Works, pp. 80-151, McMillan, New York, 1947.
6. R.H. Guting., GraphDB: Modeling and Querying Graphs in Databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, VLDB’94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile, pages 297–308.
7. R. C. Holt, A. Winter, and A. Schürr, GXL: Towards a Standard Exchange Format Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2000.
8. R. Koschke., Software Visualization in Software Maintenance, Reverse Engineering, and Reengineering: A Research Survey. In Journal on Software Maintenance and Evolution, John Wiley & Sons, Ltd., Vol. 15, No. 2, pp. 87-109, 2003.
9. J. Kung, T. Luckeneder, K. Steiner, R. Wagner, and W. Wos, Persistent topic maps for knowledge and web content management. In Proceedings of WISE (2), pages 151-158, 2001.
10. C. Lewerentz and F. Simon, Metrics-based 3D Visualization of Large ObjectOriented Programs. In: Proceedings of the First International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT02), 2002.
11. B. Shneiderman and C. Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition). Pearson Addison Wesley, 2004.
12. R. Widhalm and T. Muck, Topic Maps: Semantische Suche im Internet. Springer-Verlag, 2001.

Návrh agenta řízeného uživatelskými preferencemi

Alan Eckhardt

Karlova Univerzita, Katedra softwarového inženýrství,
Akademie věd ČR, Ústav informatiky,
alan.eckhardt@mff.cuni.cz

Abstract. *Vize sémantického webu vykresluje web tak, že bude pochopitelný pro stroje. Tento článek přistupuje k sémantickému webu z opačného konce - od uživatele. Navrhneme softwarového agenta využívajícího sémantická data získaná anotací, který je bude prezentovat uživateli podle jeho preferencí. Tento agent usnadní uživateli hledání a výběr ideálního objektu, podle jeho preferencí.*



Obr. 1. Schéma komunikace mezi agentem a uživatelem.

1 Úvod

1.1 Sémantický web jako proces a co z toho bude mít uživatel

Velká vize sémantického webu jako prostoru dat srozumitelných pro stroje se ukázala jako nerealizovatelná. Naopak se začaly objevovat malé zárodečné systémy, které umějí získávat sémantická data ze stávajícího webu a zpřístupňovat je k všeobecnému použití.

Tyto systémy se většinou již nezajímají, co se se sémantickými daty stane. Základem těchto systémů je extrakce dat ze standardních HTML stránek a jejich uložení jako dat sémantických, tedy s přidanou informací o tom, co tato data vlastně znamenají.

V zásadě lze rozlišit dva přístupy - extrakce strukturovaných dat a extrakce lingvistických dat.

V prvním případě [2], [17], [19] se vychází z toho, že data jsou rozložena po stránce v podobných útvarech (datový záznam) tak, jak to známe např. z internetových obchodů. Navíc v každém datovém záznamu je většinou odkaz na stránku s detaily daného produktu. Tyto detailní stránky jsou také velmi podobné. Z těchto podobností lze zjistit, jaké atributy a hodnoty atributů daný objekt má.

Druhý přístup je získávání sémantiky z lingvistických dat [5]. Zde se aplikují lingvistické metody pro rozklad věty na ”stromeček”, který reprezentuje roli jednotlivých slov ve větě. Navíc v každém uzlu jsou informace o daném slově, jeho slovní druh, pád, apod. Z těchto informací lze také získat sémantická data.

Většina prací se zabývá buď tím, jak sémantická data získat, jak je efektivně ukládat do úložiště, manipulovat s nimi a dotazovat se na ně. Nicméně návrhů reálných systémů, které by se daných úložišť dotazovaly, je zatím poskrovnu.

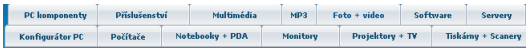
Jakým způsobem využít pracně získaná sémantická data, to je předmětem tohoto článku. Naší motivací bude uživatel, který si chce koupit notebook. V dnešní situaci musí sám procházet nabídky internetových obchodů, hledat si specifikace na stránkách výrobce, pročítat diskuze a názory vlastníků, zjišťovat informace o dodavatelích atd. Ne každý uživatel je toho schopný. Chtěli bychom tedy navrhnout uživatelsky příjemný prostředek pro nalezení nějakého objektu, v našem případě notebooku.

2 Agent a jeho interakce s uživatelem

Náš navrhovaný agent umožňuje uživateli vyhledávat objekty podle jeho preference (Obrázek 1). Nejprve je nutné, aby si uživatel vybral, o jaké objekty má zájem. To lze udělat dvěma způsoby

1. Uživatel najde agenta, který již dané objekty vyhledávat umí.
2. Agent nabídne uživateli seznam možných druhů objektů, které umí hledat.

Druhý případ bude pro běžného, neškoleného uživatele příjemnější. Toto již umí z běžných internetových obchodů, kde většinou je nějaká forma rozcestníku podle druhu objektů. Typický příklad výběru druhu objektů je na Obrázku 2 (převzato z www.mironet.cz).



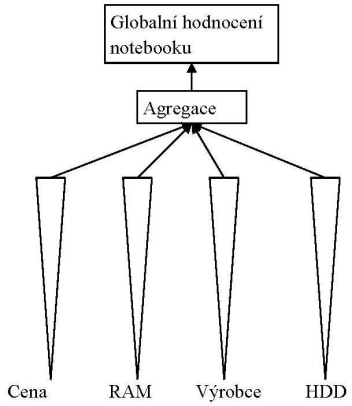
Obr. 2. Typický výběr druhu objektu v obchodě.

První případ je zase lepší pro již zkušeného uživatele - umožňuje mu najít agenta, který bude dobře odpovídat tomu, co přesně uživatel hledá. Navíc každý agent se může trochu lišit ve způsobu zadávání preferencí či hledání. Tyto vlastnosti může uživatel zohlednit při výběru agenta.

Nyní již uživatel agentovi zadal, co za objekty má hledat. Teď agent musí nějakým způsobem získat uživatelovy preference, aby věděl, jaké vlastnosti mají dané objekty mít. Uživatel může vyplnit nějakou formu dotazníku [10], což ovšem vyžaduje určitou zkušenost a také čas od uživatele. Když se mu dotazník nebude chtít vyplňovat, agent může začít práci i bez preferencí. V tom případě se použije předdefinovaná sada preferencí.

Agent nyní získá informace o objektech a o jejich atributech. Pokud již má nějaké uživatelské preference, setřídí objekty podle těchto preferencí a vybere několik nejvhodnějších. Pokud ne, použije se předdefinovaná sada - ta bude navržena tak, že se vybere několik reprezentativních objektů z celého prostoru objektů. Tuto sadu lze získat např. aplikací shlukování a vybráním centroidů nalezených shluků.

Vybrané objekty poté ukáže uživateli. Ten s nimi může být, ale také nemusí být (a typicky není), spokojen. Proto je důležité, aby měl uživatel možnost zpětné vazby pro agenta - označí mu ty objekty, u kterých se agent spletl a které se vlastně uživateli moc nelíbí.



Obr. 3. Kombinace preferencí.

3 Návrh agenta

3.1 Multikriteriální rozhodování

Jedna z hlavních součástí agenta je systém pro vyhodnocování uživatelských preferencí. Předpokládáme, že preference jsou založeny na vlastnostech notebooku. Těchto vlastností je typicky více a agent musí všechny vlastnosti zkombinovat do celkového hodnocení notebooku.

Uživatelské preference rozdělíme na dvě kategorie - lokální, které jsou definovány na jednotlivých atributech, a globální, která je definována na preferencích atributů. Lokální preference tedy slouží k normalizaci hodnot atributů, jsou to vlastně funkce $f_i : D_{A_i} \rightarrow [0,1]$. Globální pak kombinuje jednotlivé lokální preference $@ : [0,1]^N \rightarrow [0,1]$.

Celé vyhodnocování preference je tedy dvoustupňové. Když hledáme k nejlepších objektů, pak podle [9] nejprve získáme N seznamů s notebooky. V každém seznamu jsou notebooky seřazeny podle preference jednoho atributu - pokud uživatel preferuje levné notebooky, pak v prvním seznamu budou na vrchu seznamu levné notebooky a vespodu drahé. Pak se v seznamech postupuje odshora dolů, takže klesá preference v jednotlivých atributech. V určitém okamžiku se hledání zastaví a agent uživateli předloží k nejvhodnějších objektů. Na Obrázku 3 je znázorněno kombinování jednotlivých preferencí atributů. Jsou zde znázorněny jednotlivé seznamy, setříděné podle preference jednoho atributu a poté agregace těchto preferencí.

3.2 Získávání uživatelských preferencí

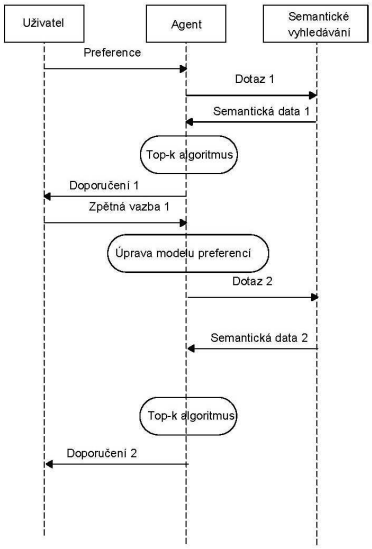
V [10] byl navrhnut systém pro přímé zadávání uživatelských preferencí. Tyto preference jsou zadávány explicitně, takže je agent může hned použít pro nalezení vhodných notebooků.

Nicméně uživatel - laik se v takovém formuláři zřejmě nebude příliš orientovat. Pro takové uživatele je určeno agentovo učení ze zpětné vazby od uživatele. Agent nejprve ukáže uživateli nějakou sadu notebooků S_0. Tato sada bude předem připravená, aby pokrývala co nejvíce prostor atributů. Nejvhodnější se jeví ji vygenerovat pomocí shlukování. Sadu S_0 uživatel ohodnotí, označí objekty, které se mu líbí a ty, které se mu nelíbí.

Tím agent dostane od uživatele první informace o jeho preferencích. Tyto informace zpracuje, vytvoří si model jeho preferencí a podle těchto preferencí mu nabídne sadu S_1, která bude již odvislá od uživatelových preferencí. Uživatel opět tyto notebooky ohodnotí, agent zreviduje model a nabídne další sadu S_2. Takto proces pokračuje až do okamžiku, kdy uživatel nalezne jeho vysněný notebook.

Na Obrázku 4 je znázorněna posloupnost akcí agenta a uživatele. První akce od uživatele, který předává své preference, není povinná - to je případ, kdy agent předá uživateli připravenou sadu S_0.

V našem návrhu jsme implementovali metodu Statistical pro získávání modelu uživatelských preferencí založeném na fuzzy logice. Statistical jsme navrhli v [6] a dále rozšířili v [8] a [7]. Tato metoda je otestována v části 4. Metoda je zaměřena na nominální atributy (nečíselné, např. barva, výrobce apod) a využívá rozložení hodnocení objektů s danou hodnotou atributu. V našem případě tedy např. sleduje, jaké mají hodnocení notebooky od výrobce Toshiba. Pokud mají spíše dobré, Toshiba bude mít spíše pozitivní vliv, a naopak. Pokud je rozložení rovnoměrné, pak výrobce Toshiba nijak výrazně neovlivňuje hodnocení notebooku.



Obr. 4. Komunikace mezi agentem, uživatelem a sémantickým úložištěm.

3.3 Dotazování se dat

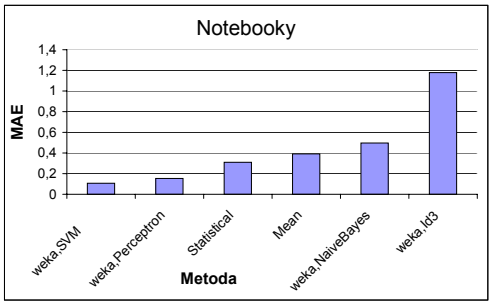
Agent musí být schopen si říct o data od nějakého sémantického vyhledávače. V této době je normou dotazovací jazyk SPARQL (http://www.w3.org/TR/rdf-sparql-query/), ovšem existuje velké množství jiných jazyků. Proto tato část bude značně odvislá od toho, jaké rozhraní poskytuje ten který vyhledávač. Nyní, ve fázi návrhu, se tedy omezíme na tvrzení, že bude stačit podpora SPARQL. V implementační fázi pak bude nutné dodělat moduly pro jednotlivé vyhledávače.

4 Experimenty

Provedli jsme experimenty na učení se uživatelského modelu. Testování bylo provedeno na umělých datech - vytvořili jsme sadu funkcí, které zastupovaly uživatele. Tyto funkce vytvořily hodnocení objektů, v našem případě notebooků. V úvahu se braly atributy cena, velikost harddisku, velikost paměti RAM a výrobce.

Sada dat obsahuje 203 notebooků. Testování jsme provedli 5-ti násobnou křížovou validací. K porovnání jsme použili čtyři tradiční dataminingovské metody z frameworku Weka [21] a funkci Mean. Mean spočte průměrné hodnocení na trénovacích datech, které uživatel zadal. Toto hodnocení pak vrací na každý objekt z testovací sady. Jde tedy o velmi jednoduchou funkci, sloužící pro poměření výkonu v absolutním měřítku. Metody z Weky jsme použili v základním nastavení.

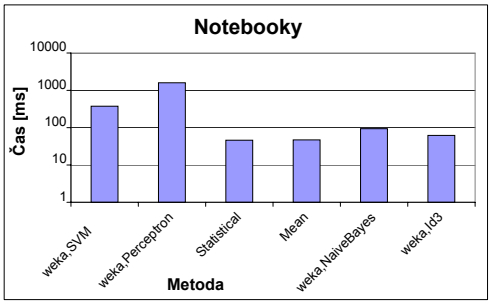
Chybu jsme měřili pomocí MAE (Mean Average Error), což je průměrná odchylka od skutečného hodnocení. Z Obrázku 5 je vidět, že neuronové sítě a metoda



Obr. 5. Chyba na testovacích datech.

založená na podpůrných vektorech (SVM) jsou řádově lepší než námi implementovaná metoda Statistical. Naproti tomu jsme překonali známou metodu naivního bayesovského klasifikátoru a rozhodovacích stromů generovaných algoritmem ID3.

Pozitivní je také fakt, že metoda Mean je znatelně horší než naše Statistical. Z toho lze vyvodit, že Statistical má smysl používat.



Obr. 6. Čas vytváření modelu.

Dále jsme měřili čas, který metody potřebují na vytváření modelu (Obrázek 6). Zde Statistical vychází velmi dobře. Poráží všechny ostatní metody, jedině Mean je rychlejší. To jsme ovšem předpokládali, vzhledem k jednoduchosti metody Mean.

5 Související výzkum

V článku jsme citovali pouze přímo využívané reference. Naproti tomu v každé oblasti je obsáhlá literatura.

V multikriteriálním rozhodování nesmíme opomenout [12], kde autor navrhuje způsob vyhodnocování top-k dotazů. To je ovšem pro lokální databázi - v našem případě využijeme distribuovaného pojetí v [9]. Touto cestou se vydává také např. [18].

Další velká oblast se týká získávání modelu preferencí. Kromě prací rozsáhlé komunity zabývající se data miningem zmíníme alespoň [3], [11], [14], [16], [20], kteří se vydali cestou zohlednění indukce pro uživatelské preference.

V tomto textu nezmiňujeme problém interpretace uživatelova chování. Akce, které provádí na stránce, lze interpretovat ve smyslu preferencí. Typický je čas strávený na stránce. Pokud stránka pojednává o nějakém produktu, můžeme říci, že se o produkt zajímá (a nebo si zašel na kafe). V této oblasti můžeme citovat [1], [4], [13], [15].

6 Závěr

V tomto článku jsme navrhli agenta, který by umožňoval doménově specifické hledání v podobě doporučení. Popsali jsme interakci s uživatelem a navrhli způsob, jakým zpracovávat jeho odezvu. Otestovali jsme námi navrženou metodu na umělých datech.

Do budoucna bychom chtěli doimplementovat celý systém interakce a ten otestovat na reálných datech. Podoba budoucích experimentů by měla zahrnovat živé osoby, které budou s agentem komunikovat.

Poděkování

Tento výzkum byl zčásti financován výzkumnými projekty 1ET 100300517 a MSM 0021620838.

References

1. M. Barla. Estimation of user characteristics from logs of user activity. In P. Návrat, P. Barto‘, M. Bieliková, L. Hluchý, and P. Vojtáš, editors, *Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, pages 175–181. Slovak University of Technology, Bratislava, 2006.
2. R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction. In *International Conference on Very Large Data Bases (VLDB)*, 2001.
3. K. Cao-Van. *Supervised Ranking, from semantics to algorithms*. Ph.D. dissertation, Ghent University, 2003.
4. L. Deng, W. Ng, X. Chai, and D.-L. Lee. Spying out accurate user preferences for search engine adaptation. In *Advances in Web Mining and Web Usage Analysis*, volume Volume 3932/2006, pages 87–103. Springer Berlin / Heidelberg, 2006.
5. J. Dědek and P. Vojtáš. Extrakce informací z textově orientovaných zdrojů webu. In *Konference Znalosti*, pages 331–334, 2008.
6. A. Eckhardt. Inductive models of user preferences for semantic web. In K. R. Jaroslav Pokorny, Vaclav Snasel, editor, *In proceedings of Dateso 2007*, pages 103–114. Desná, Czech Republic, ISBN 80-7378-002-X, 2007.
7. A. Eckhardt, T. Horváth, and P. Vojtáš. Learning different user profile annotated rules for fuzzy preference top-k querying. In H. P. V. Subrahmanian, editor, *In proceedings of Scalable Uncertainty Management (SUM07)*, pages 116–130. Springer-Verlag 2007 ISSN 0302-9743, 2007.
8. A. Eckhardt, T. Horváth, and P. Vojtáš. Phases: A user profile learning approach for web search. In T. L. et al Eds., editor, *In proceedings of 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2007)*, pages 780–783. IEEE computer society Los Alamitos, California, Washington, Tokyo 2007 ISBN 0-7695-3026-5/07, 2007.
9. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *In proceedings of Twentieth ACM Symposium on Principles of Database Systems, 2001 (PODS 2001)*, pages 102–113. ACM, 2001.
10. P. Gurský, T. Horváth, J. Jirásek, R. Novotný, J. Pribolová, V. Vaneková, and P. Vojtáš. Knowledge processing for web search—an integrated model and experiments. *Scientific international journal for parallel and distributed computing, ISSN 1097-2803*, 9:51–59, 2008.
11. S. Holland, M. Ester, and W. Kießling. Preference mining: A novel approach on mining user preferences for personalized applications. In *Knowledge Discovery in Databases: PKDD 2003*, pages 204–216. Springer Berlin / Heidelberg, 2003.
12. I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In *VLDB*, pages 754–765, 2003.
13. T. Joachims. Optimizing search engines using clickthrough data. In *KDD ’02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.
14. S. Y. Jung, J.-H. Hong, and T.-S. Kim. A statistical model for user preference. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):834– 843, June 2005.
15. D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
16. S. Ko and J. Lee. User preference mining through collaborative filtering and content based filtering in recommender system. In *E-Commerce and Web Technologies: Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002. Proceedings*, page 244. Springer Berlin / Heidelberg, 2002.
17. N. Kushmerick. Wrapper induction: efficiency and expressiveness. In *Artificial Intelligence*, pages 118:15–68, 2000.
18. S. Michel, P. Triantafillou, and G. Weikum. Klee: A framework for distributed top-k query algorithms. In *International Conference on Very Large Data Bases (VLDB)*, pages 34–43, 2008.
19. I. Muslea, S.Minton, and C.Knoblock. A hierarchical approach to wrapper induction. In *Conf. on Autonomous Agents*, 1999.
20. P. Vojtáš and M. Vomlelová. On models of comparison of multiple monotone classifications. In B. Bouchon-Meunier and R. R. Yager, editors, *In proceedings of IPMU’2006*, pages 1236–1243. Éditions EDK, Paris, 2006 ISBN : 2-84254-112-X, 2006.
21. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.

Fuzzy logic and piecewise-linear regression

Ján Fröhlich[1] and Martin Holeňa[2]

[1] Student of Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering
Břehová 7, Prague, Czech Republic
jfrohlich@gmail.com
[2] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, Prague, Czech Republic
martin@cs.cas.cz

Abstract. *McNaughton theorem of fuzzy logic states that the functions on the unit cube which admit representation by formulas of the Łukasiewicz propositional logic are exactly all continuous piecewise-linear functions (CPLFs) with integer coefficients. The paper discusses an approach based on the linear regression in this context. A CPLF and corresponding polyhedral partition of the input domain that best approximate given sample data are sought, considering the trade-off between the accuracy of the approximation and the complexity of the resulting formula. The proposed method is scalable and combines a kd-trie space-partitioning data structure, a linear regression and a clustering supported by hypothesis testing.*

1 Introduction

Fuzzy logic has been frequently used for the representation of knowledge extracted from data in data mining. Nevertheless, the existing methods typically rely on general properties of fuzzy logic [5]. Nearly no attention has been paid to specific properties of particular kinds of fuzzy logic. In this article, we deal with Łukasiewicz propositional fuzzy logic. There are several reasons for our interest in Łukasiewicz logic: piecewise-linear functions it is related to have good approximation capabilities, the linear relationship is easy to understand and for finding the piecewise-linear approximation of given data, efficient methods such as the linear regression and the linear discrimination are available.

In the next section, function representation in Łukasiewicz propositional fuzzy logic will be recalled. The concepts related to the piecewise-linear regression (PLR) are discussed in Section 3. Finally, the proposed method is described in Section 4.

2 Łukasiewicz propositional logic and piewcewise-linear functions

We recall the essentials necessary to understand the role of fuzzy logic in a function representation. A detailed treatment can be found in the monographs [3] and [9]. Fuzzy logic extends the classical logic to deal with the concept of *partial truth*. Truth value can assume a continuum between 0, absolute falsity, and 1, absolute truth. The truth function $\mathbf{t}$ of *conjuction* $\&$ is a commutative and associative mapping $\mathbf{t} : [0,1]^2 \to [0,1]$, non-decreasing in both arguments, and satisfying the conditions $1\mathbf{t}x = x, 0\mathbf{t}x = 0$ for all $x \in [0,1]$. A binary operation having properties just listed is called *t-norm*. The truth function $\Rightarrow_{\mathbf{t}}$ of *implication* is defined as the *residuum of the t-norm* $\mathbf{t}$: $x \Rightarrow_{\mathbf{t}} y = \sup\{z \mid x\ \mathbf{t}\ z \leq y\}$. In Łukasiewicz logic, these definitions result in Łukasiewicz conjuction $*$, $x * y = \sup(0, x + y - 1)$, and Łukasiewicz implication, $x \Rightarrow_* y = \min(1, 1 - x + y)$. The algebra underlying Łukasiewicz logic is called *MV-algebra*; in this paper the *standard MV-algebra* on $[0,1]$ is considered. *Rational truth constants* $\bar{r}$ representing *rational numbers* $r \in [0,1]$ are introduced to a fuzzy logic language if considered desirable. To enable representing the non-continuous piecewise-linear functions too, Łukasiewicz logic may be enriched by the *Baaz delta* connective $\triangle$, $\triangle(1) = 1$ and $\triangle(x) = 0$, $0 \leq x < 1$; the resulting logic is denoted $\text{Ł}_\triangle$.

Definition 1. *Let $f : [0,1]^k \to [0,1]$ be a function, and Φ be a formula of Łukasiewicz propositional logic. The Φ is called* representation *of f (more precisely, representation of f with respect to the standard MV-algebra on $[0,1]$) if two conditions are satisfied: the variables of Φ are a subset of a set of k variables $\{x_1, \ldots, x_k\}$, and for each evaluation $v_1, \ldots, v_k$ of the variables $x_1, \ldots, x_k$, the corresponding evaluation of Φ is equal to the value $f(v_1, \ldots, v_k)$.*

Definition 2. *A continuous function $f : [0,1]^k \to [0,1]$ of k real variables is called* continuous piecewise-linear function (CPLF) *with integer coefficients if there exist k-dimensional convex polyhedra $P_i, i \in \{1, \ldots, p\}$, such that $\bigcup_{i=1}^{p} P_i = [0,1]^k$ and $f|P_i$ are linear functions with integer coefficients. The polyhedra $P_i, i \in \{1, \ldots, p\}$, are called the* linearity domains *of f and the set $\{P_1, \ldots, P_p\}$ is termed a* polyhedral partition of the unit cube $[0,1]^k$.

Theorem 1 (McNaughton [7], 1951). *The set of all functions on $[0,1]^k$ that admit representation by some formula of Łukasiewicz propositional logic is equal to the set of all continuous piecewise-linear mappings of $[0,1]^k$ to $[0,1]$ with integer coefficients.*

Given a formula, the function it represents can by easily obtained using the definition of evaluation. On the other

hand, because the original proof of McNaughton theorem was not constructive, an algorithm how to find the representation of a given function was unknown until 1990s when two constructive proofs of the opposite inclusion were published, by Mundici [8] and by Perfilieva and Tonis [10]. Let us recall that as it is in general not assured that the subtotal of the linear combination $a_{i,1}c_1 + \cdots + a_{i,k}c_k$ never leaves $[0,1]$, the restriction of operations on standard MV-algebra to $[0,1]$ causes both the length of resulting formula and time complexity of the algorithm to be $O(2^L)$, where $L = |a_{i,1}| + \cdots + |a_{i,k}|$, as discussed in [4].

From the knowledge representation point of view, expressing the structure of the dependency that a function describes may be considered more important to concern about than to preserve a mapping to properly scaled function values. Given a function $f : [0,1]^k \to [0,1], f(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x} + b$, consider the function $\tilde{f}(\mathbf{x}) = \frac{f(\mathbf{x})}{L}$, where $L = 2(\max\{\sum_{a_i>0} a_i, \sum_{a_i<0} |a_i|\} + |b|)$. Note that the partial sum of the expression $\tilde{f}(\mathbf{x}) = (\cdots((\frac{1}{2} + \frac{b}{L}) + a_1\frac{x_1}{L}) + \cdots + a_k\frac{x_k}{L}) - \frac{1}{2}$ never leaves $[0,1]$. Hence, in Łukasiewicz propositional logic enriched by rational constants, the representation $\tilde{\Phi}$ of the function $\tilde{f}$ can be found in a polynomial number of steps producing a formula of a polynomial length, provided an evaluation assigning values $\frac{x_i}{L}$ to propositional variables representing x_i is applied. The original function value can be retrieved by an inverse linear transformation, $f(\mathbf{x}) = L\tilde{f}(\mathbf{x})$.

A method partitiong the space of explanatory variables to convex polyhedra is desirable, that allows to control the accuracy of the approximation by a CPLF. The number of the linearity domains is unknown beforehand.

3 Piecewise linear regression

Consider n sample points: a $n \times k$ dimensional matrix $\mathbf{X}$ stores their coordinates $x_{i,1}, \ldots, x_{i,k}, i \in \{1, \ldots, n\}$, in the space of the independent explanatory variables $X_1, \ldots X_k$, and a $n \times 1$ dimensional vector $\mathbf{y}$ contains corresponding values y_i of a dependant variable. With respect to the context discussed in the preceding section, the points are assumed to lie in the unit hypercube. The objective is to cluster together points that belong to a common hyperplane in $(k+1)$-dimensional space: location of clusters and coefficients of linear function are to be found. Now, the underlying concepts will be recalled.

Linear regression The dependent variable Y is modelled as a linear combination $\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$ of independent variables $X_i, i \in \{1, \ldots, k\}$, superposed with random error $E \sim \mathrm{N}(0, \sigma^2)$; values of independent variables are assumed to be error free. If *residuals* $e_i = y_i - (\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k), i \in \{1, \ldots, n\}$ are realizations of independent random variables $E_i \sim \mathrm{N}(0, \sigma^2)$, a maximum likelihood estimation of parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_k)'$ is $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, where the matrix $\mathbf{X}$ has been extended by a column of ones corresponding to the constant term β_0.

Test for linear hypothesis Let $\mathbf{T}$ is a $t \times k$ matrix of rank $t < k$, H_0 denotes the null hypothesis that $\mathbf{T}\beta = 0$, and R_0 and R are sums of squared residuals of model when H_0 is valid and of model without any restrictions, respectively. Then the random variable $F = \frac{(n-k)(R_0-R)}{tR}$ has a $\mathrm{F}_{t,n-k}$ distribution [1]. The version of the test for testing the null hypothesis that model parameters of two disjoint groups of observations are equal, is known as *Chow test*.

Measure of quality of fit Once the parameters' estimate $\hat{\beta}$ is obtained, the fitted values $\hat{\mathbf{y}}$ can be calculated from the model: $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$. The *total sum of squares* (TSS) $\sum_{i=1}^n y_i^2$ is the sum of *regression sum of squares* (RSS) $\sum_{i=1}^n \hat{y}_i^2$ and *error sum of squares* $\sum_{i=1}^n (y_i - \hat{y}_i)^2$. The non-centered *coefficient of determination* R'^2 is defined as the proportion of TSS that is explained by the model; for non-constant linear models the averages are substracted from the values and *centered coefficient of determination* R^2 is introduced.

Linear classifier For the purpose of the method proposed, any technique that allows to separate two sets of samples by a hyperplane is competent. We chose the *logistic regression* for its simplicity: the log likelihood (probability of a class, given the data) ratio is modelled as a linear decision boundary [1].

4 Proposed method

First, a *kd-trie* data structure [11] from the coordinates of samples in the space of explanatory variables is constructed, so that the leaves contain sufficiently small number of samples. We call a part of the corresponding space partition *a cell*. The cells are then analyzed: a linear regression of samples in the cell is performed. The cells with a good linear fit, measured by the coefficient of determination, will be referred to as *cluster cells*, the remaining cells are termed *border cells*. Neighbouring cluster cells containing the samples of the same linear function, according to Chow test, are then gradually merged. The convexity of the merged cells is examined. Border cells are subsequently merged too, in order to explore the topology of cluster cells. Finally, the linearity domains are found by intersecting function hyperplanes of each cluster and its neighbours. If the resulting function is not required to be continuous, hyperplanes delimiting each polyhedron are found by a linear classification method. In clustering tasks, a class dominating each cell is found and the quality of the cell classification is measured by the relative presence of the leading class. The algorithm might be summarized as follows.

1. Build a kd-trie from the samples.
2. Perform a linear regression of samples in each leaf.
3. Discriminate the cluster and border cells.
4. Merge the cluster cells contaning the samples of the same function.
5. Merge the border cells, and for each cluster cell, find the neighbouring cluster cells.
6. Find a polyhedral partition.

A similar PLR method based on neural network and classical clustering and classificaton algorithms has been described in [2].

A kd-trie is constructed from the set of samples of k-dimensional data such that each cell (a leaf node of the kd-trie) contains less than twice the *minimal number of samples per cell*. The list of neighbours of each cell is built. The following four steps are repeated, each cell containing at least twice the minimal number of samples is split in two cells:

1. The splitting dimension $d \in \{1, \ldots, k\}$ is selected in which the variance of d-th coordinates of the samples in the cell is maximal.
2. The median m of the d-th coordinates of the samples in the cell is calculated.
3. If $m > 0$ the cell is split by the hyperplane $x_d = m$.
4. The lists of neighbours of all cells adjacent to the original cell are updated and the neighbour lists for both new cells are created.

If no cell is divided during a round, the process is terminated.

Cells where the value of coefficient of determination is greater or equal to the *minimal value of coefficient of determination* are assigned to the set of cluster cells; the remaining cells are considered to be the border cells. A centered or non-centered coefficient of determination is applied, depending on whether the hypothesis that the function values of observations in a cell are constant is rejected on the significance level of constant model test or not.

To measure a convexity of an union of adjacent cells, a *convexity ratio* γ is defined as a ratio of the sum of volumes of individual cells united, and the volume of the convex hull of the vertices constituting the cells considered. Clearly, $\gamma \in (0, 1]$.

Subsequently, the cluster cells are merged. All cluster cells are browsed:

1. Merging a cluster cell c with each neighbouring cell t is considered.
   - A cell t is merged to the cell c if two conditions are satisfied: *the null hypothesis of Chow test is not rejected* on the specified significance level, and the convexity ratio of the united cells exceeds the *minimal convexity ratio* required.
   - A link between the cells is created if the observations they contain are considered to belong to the same hyperplane, but merging is not allowed due to convexity reasons.
2. If a cell t is merged to c, neighbour lists of adjacent cells are updated, t and cells merged to t are appended to the list of cells merged to c and the neighbour list of c is updated.
3. Cell t is removed from the list of cluster cells. If any cell was merged to c or a link was created, the process is repeated with the updated list of neighbours.

To reduce the number of border cells, each cluster cell c is visited once again:

1. For each border cell b adjacent to c, the nearest cluster(s) c' different than c are found; the distance is measured by the number of border cells constituting the shortest connection.
2. The border cells connecting c and c' are merged and neighbour lists of affected cells are updated.

Then, the border cells that connect the same set of cluster cells are merged, and, for each cluster, the cluster cells behind its border are appended to the list of its neighbours.

Finally, the polyhedral partition is constructed. For each cluster cell c, the intersections of hyperplanes in $(k+1)$-dimensional space representing the function on c, functions on cluster cells neighbouring to c, and hyperplanes delimiting the root cell are constructed. Vertices of the linearity domain corresponding to the cluster cell c are the intersections projected to the k-dimensional space of explanatory variables. If the continuity of a piecewise-linear function is not required (the representation in $Ł_{\triangle}$ is sought), the desired polyhedron is found as the intersection of the halfspaces delimited by the hyperplanes discriminating the samples of c from the samples of each cluster cell neighbouring to c (the hyperplanes are found by linear classification of the samples). Each sample point in an adjacent bordering cell is assigned to the cluster cell the function hyperplane of which is closest to that point. The resulting polyhedra might be merged (similarly like the cluster cells were) and the discriminating step repeated.

4.1 Scalability

The proposed method is scalable. We suppose any restriction on resources available can be transformed to a constraint in the number $m \in \mathbb{N}$ of cells it is possible to handle at once. The space of explanatory variables can be explored step by step: at a particular level of building the kd-trie, the splitting of all but one cell is abandoned, so that enough resources are left to construct the full-depth kd-trie for the cell c chosen. The algorithm then merges only the cells that belong to the full-depth kd-trie. Once the cells originating from the cell c are merged to the level where the splitting of

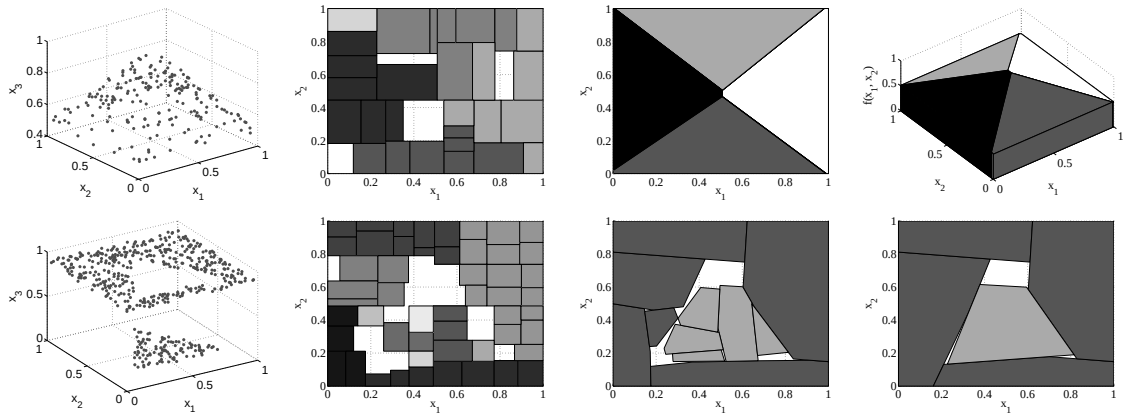


Fig. 1. The top row: PLR of sample points in a three-dimensional space, the continuity is required. The sample data, the merged cluster cells, polyhedral partition and resulting function hyperplanes are depicted, respectively. Individual cluster cells are differentiated by colour shades. The bottom row: PLR of sample points in a three-dimensional space, the continuity is not required. The sample data, the merged cluster cells and resulting linearity domains before and after merging are depicted, respectively. Note the way the non-convex area surrounding the triangular hyperplane in the centre is treated.

other cells was abandoned, all but the resulting cells may be removed and the resources freed. All cells, one by one, can be explored in this way. Cluster and border cells originating from individual depth explorations are then processed as if they were at the leaf level.

Needless to say, there must be enough resources available to store the problem solution on a specified approximation level.

4.2 Implementation

We have implemented the method in Matlab using the Multi-parametric toolbox [6]. The application of the method is illustrated on two sample sets (Figure 1).

5 Conclusion

The suitability of fuzzy logic for knowledge representation, good approximation capabilities of piecewise-linear functions, and the efficiency of linear regression motivated us to consider an application of the piecewise-linear regression for finding the formulas of Łukasiewicz logic. We recalled the concepts of PLR and proposed a scalable PLR method that allows to regulate the level of approximation and recognizes natural obligue borders of linearity domains. We are going to test the method on real world data. Then we will address the subsequent step of finding a formula representing given data in Łukasiewicz propositional logic: we will implement a polynomial algorithm for finding the formula that represents a given function. Finally, we intend to obtain even more comprehensible knowledge representation — by describing the found CPLF using the formulas of Łukasiewicz predicate logic.

Acknowledgment

The research reported in this paper has been supported by the grants 201/08/0802 and ICC/08/E018 of the Grant Agency of the Czech Republic.

References

1. Anděl J., Statistické metody, Matfyzpress, Praha, 1998
2. Ferrari-Trecate G., Muselli M., A New Learning Method for Piecewise Linear Regression, J.R. Dorronsoro (Ed.): ICANN 2002, Springer-Verlag, Berlin, 2002, 444–449
3. Hájek P., Metamathematics of Fuzzy Logic, Kluwer, Dordrecht, 1998
4. Holeňa M., McNaughton theorem of fuzzy logic from a data-mining point of view, Neural Network World 3, 2007, 180–212
5. Holeňa M., Statistical, logic based, and neural networks based methods for mining rules from data, in Hyder, A.K. and Shahbazian, E. and Waltz, E. Multisensor Fusion, Kluwer, Dordrecht, 2002, p. 511–532
6. Kvasnica M., Grieder P., Baotic M., Morari M., Multi-parametric toolbox (MPT), Hybrid Systems: Computation and Control, Springer Verlag, Berlin, 449-462
7. McNaughton R., A theorem about infinite-valued sentential logic, Journal of Symbolic Logic, 16, pp. 1-13
8. Mundici D., A constructive proof of McNaughton's theorem in infinite-valued logic, Journal of Symbolic Logic, 59, 1994, pp. 596–602
9. Novák V., Perfilieva I., Močkoř J., Mathematical Principles of Fuzzy Logic, Kluwer, Boston, 1999
10. Perfilieva I., Tonis A., Functional system in fuzzy logic formal theory, Bulletin for Studies and Exchanges on Fuzziness and its Applications, 64, 1995, pp. 42–50
11. Teorey T. J., Fry J. P., Design of Database Structures, Prentice Hall Professional Technical Reference, 1982

Optimalizácia parametrov pri sťahovaní dynamicky generovaných stránok

E. Gatial, Z. Balogh a L. Hluchý

[1]Institute of Informatics, Slovak Academy of Sciences
Dubravska cesta 9, 845 07 Bratislava, Slovakia
emil.gatial@savba.sk

Abstract. *V súčasnosti sa rapídne zvyšuje množstvo informácií obsiahnutých v Internete, čo kladie čoraz väčšie nároky na vyhľadávacie a indexačné služby ako Google, Yahoo a podobné. Tieto služby sa čoraz viac potykajú s problémom spracovania dynamicky generovaných stránok. Ich spracovanie je problematické najmä z dôvodu zahltenia množstvom podobných stránok. Tento článok sa zaoberá vyhodnocovaním a optimalizáciou URI parametrov podľa podobnosti obsahu stiahnutých stránok. Tento spôsob vyhodnocovania predpokladá, že dynamické stránky sú skladané a generované podľa parametrov URI dotazu, pričom každý parameter do istej miery ovplyvňuje obsah a formu stránky. Popísaná metóda porovnáva odlišnosti v obsahu stránky a vyhodnocuje koeficient významnosti URI parametrov a generuje redukovanú URI adresu HTML stránky, tak aby zamedzilo duplicitnému spracovaniu stránok s podobným obsahom. Záver článku obsahuje stručné vyhodnotenie popísanej metódy.*

1 Úvod

V súčasnosti sa rapídne zvyšuje množstvo informácií obsiahnutých v Internete, čo kladie čoraz väčšie nároky na vyhľadávacie a indexačné služby ako Google, Yahoo a podobné. Tieto služby sa čoraz viac potykajú s problémom spracovania dynamicky generovaných stránok. Ich spracovanie je problematické najmä z dôvodu zahltenia množstvom podobných stránok – teda stránok s podobným alebo totožným obsahom. Na základe štúdie [1] spracovanej v roku 1997 bol odhadnutý počet dynamicky spracovaných stránok na 80% s rastúcou prognózou. Aj keď existujú technológie, ktoré zjednodušujú prácu s dynamicky generovaným obsahom stránok (ako napríklad ASP[1], JSP[2]), trend stúpania počtu dynamických stránok stále rastie. Metódam na identifikáciu zmien obsahu sa venuje práca [2], ktorá sa zameriava hlavne na odhaľovanie zmien obsahu stránku pre atribúty identifikátorov *sessions*, ktoré sú založené na vyhľadávaní reťazcov rovnakej dĺžky zakódovaných do URI parametra. Príbuzná problematika sa týka metód sťahovania skrytého webu, kde dynamicky vytvárané stránky nemusia byť nutne prístupné pomocou hypertextového prepojenia. Navyše pri dynamicky generovaných stránkach nemusí nutne platiť podmienka ukončenia na stiahnutie všetkých stránok zo spracovávaného webového sídla. Metódami na sťahovanie dynamických stránok sa venuje článok [3], v ktorom autori navrhli metódy pomocou ktorých je možné spracovať približne 90% stránok len na 3 až 5 úrovní prehľadávania.

Tento článok sa zaoberá optimalizáciou URI adresy pri sťahovaní webového sídla, tak aby boli vylúčené stránky s podobným obsahom. Úvodná časť je venovaná motivácii optimalizácie a vysvetleniu problematiky dynamických stránok. Hlavná časť článku sa venuje navrhovanej metóde optimalizácie. V záverečnej časti je navrhovaná metóda zbežne vyhodnotená na vzorkách domén spracovaných nástrojom WebCrawler [4].

1.1 Úloha dynamických stránok

V kontexte tohto článku budeme dynamickú stránku chápať ako HTML stránku, ktorej obsah bol vytvorený pomocou programu počas požiadavky na jej zobrazenie. Naopak, obsah statickej stránky existoval ešte pred zaslaním požiadavky na zobrazenie.

Dynamické stránky sú bežne používané tam, kde je nutné prepojenie alebo sprostredkovanie údajov dynamických údajov. Spravidla najbežnejším použitím dynamických stránok je agregovanie údajov z databáz alebo iných distribuovaných informačných zdrojov.

1.2 Motivácia

Navrhovaná metóda optimalizácie URI parametrov vychádzala z potreby zjednodušenia spracovania dynamických stránok pri sťahovaní stránok nástrojom WebCrawler. Hlavný cieľ optimalizácie URI parametrov je vylúčiť so spracovania stránky, ktoré majú totožný informačný obsah, ale líšia sa len v detailoch, ktoré sú spôsobené dynamickým generovaním stránky. Najčastejšie sa jedná o odlišnosti ako aktuálny datum a čas, reklamné banery, aktuálne správy, a pod. Návrh metódy bol spočiatku určený pre identifikáciu parametrov, ktoré vôbec neovplyvňujú generovanie obsahu a neskôr bol zameraný na analýzy vplyvu parametrov na obsah stránky. Pri návrhu a implementácii metódy boli brané do úvahy dva prípady na platnosť identifikovaného URI parametra:

- Globálna platnosť parametra vzhľadom na URI cestu, kde identifikovaný parameter je uložený pre spracovávané Webové sídlo. Tento prípad platí za predpokladu, že všetky parametre majú jednoznačný význam vzhľadom na spracovávané sídlo.
- Lokálna platnosť parametra vzhľadom na URI cestu, kde identifikovaný parameter je uložený pre konkrétnu cestu. V tomto prípade sú vyhodnotené všetky parametre, ktoré sa vyskytujú v rôznych cestách. Toto vyžaduje väčšiu časovú aj pamäťovú náročnosť.

Pri analýze obsahu sťahovaných stránok boli identifikované tri druhy dynamickosti stránok v závislosti od použitých parametrov:

- Časová dynamickosť – obsah stránky je závislý od času, v ktorom bola stránka vygenerovaná (napríklad

[1] Active Server Pages Technology. http://msdn.microsoft.com/workshop/server/asp/aspfeat.asp

[2] JavaServer Pages (JSPTM) Technology. http://java.sun.com/products/jsp/.

čas, dátum, odkazy na reklamu, počítadlo návštevnosti, novinky, a pod.).

- Používateľská dynamickosť – obsah generovaný v závislosti od používateľových nastavení (napríklad rozoženie stránky, štýly a pod.).
- Dynamickosť závislá od vstupu – obsah stránky je generovaný v závislosti od vstupu zadaného používateľom (napríklad požiadavka na zobrazenie žiadaného obsahu, filtrovanie údajov,).

Vo všeobecnosti stránky môžu obsahovať ľubovoľnú kombináciu všetkých druhov dynamických čŕt.

1.3 Problémy pri spracovaní dynamických stránok

Jedna z najväčších prekážok automatického spracovania – indexovateľnosti - stránok je požiadavka registrácie alebo iný druh autorizácie (ako napríklad meno/heslo) potrebnej na vstup do systému alebo vstup do systému povolený len z lokálnej podsiete (intranet). Problematické je taktiež spracovať dynamické stránky, ktoré vyžadujú konkrétny vstup od používateľa, ktorý nie je v čase automatického spracovania známy (spracovanie pomocou crawler/spider metód) ako napríklad dotaz na vyhľadanie konkrétneho druhu tovaru.

Stránky ktoré nie sú automaticky spracovateľné sa nazývajú neindexovateľné a pokrývajú časť Internetového informačného priestoru, ktorý sa nazýva skrytý Web (ang. hidden Web). Keďže sa tento článok venuje spracovaniu parametrov, preto pojem „stránka“ sa bude v texte tohto článku ďalej spomínať len v súvislosti spracovania dynamických spracovateľných stránok. Na statické stránky je možné pozerať ako na špeciálny prípad dynamických stránok prístupných z URI s nulovým počtom parametrov.

Obr. 1. Rozloženie spracovateľnej a nespracovateľnej časti Webu.

Cieľom metódy na optimalizáciu parametrov URI adresy je redukovať počet spracovaných stránok tým, že niektoré parametre budú vylúčené zo spracovania. Táto metóda predpokladá, že parameter ovplyvňuje jednu vlastnosť stránky a parametre sú navzájom nezávislé.

2 Metóda redukcie parametrov URI adries dynamicky generovaných stránok

Automaticky spracovateľné stránky sú väčšinou prístupné pomocou zaslania požiadavky na URI adresu *u* s jedným alebo viacerými parametrami. Každá URI adresa sa pre HTTP protokol (podľa RFC 3986[3]) definuje podľa schémy:

```
scheme ":" hier-part [ "?" query ]
       ["#" fragment ]
```

kde

- `scheme` vyjadruje prenosový protokol (v prípade HTML stránok je to http alebo https),
- `hier-part` jednoznačne identifikuje webové sídlo a skladá sa z Internetovej adresy (angl. authority) a cesty k informačnému zdroju (angl. path),
- `query` zahŕňa nehierarchicky usporiadanú množinu parametrov, ktorá jednoznačne identifikuje zdroj v rámci webového sídla,
- `fragment` je URI komponent, ktorý umožňuje identifikovať sekundárny zdroj vzhľadom na primárny zdroj.

V kontexte tohto článku bude ďalej rozoberaný len atribút *query*, ktorý obsahuje množinu parametrov oddelenú značkou „&“. Každý parameter môže obsahovať meno parametra a hodnotu parametra oddelenú značkou „=“. Atribút *query* potom môže obsahovať parametre $parameter_i$, zapísaný ako

$$name_i \ [\ „=“ \ value_i \ ]$$

pričom obsahuje meno parametra $name_i$ a môže obsahovať hodnotu parametra $value_i$ oddelenú značkou „=“. A samotný atribút *query*, ktorý obsahuje *n* parametrov je potom zapísaný ako

$$parameter_1 \ „\&“ \ parameter_2 \ „\&“ \ ... \ „\&“ \ parameter_n$$

Pri spracovávaní webového sídla *S* sú potom spracovávané stránky definované ako množina URI adries *u*, ktoré boli získané počas spracovania nástrojom WebCrawler.

Metóda redukcie URI parametrov je založená na vyhodnocovaní vplyvu neznámeho parametra $parameter_i$ vzhľadom na to, či je skúmaný parameter je ponechaný v požiadavke na spracovanie URI adresy *u(query)* alebo je z tejto URI adresy vynechaný *u(query - parameter$_i$)*. Pri analýze vplyvu parametra $parameter_i$ sa vychádza z predpokladu: „pokiaľ sú si stránky podobné alebo sú totožné, tak parameteri nemá vplyv na informačný obsah požadovanej stránky a môže byť pri ďalšom spracovaní vyradený zo spracovania“.

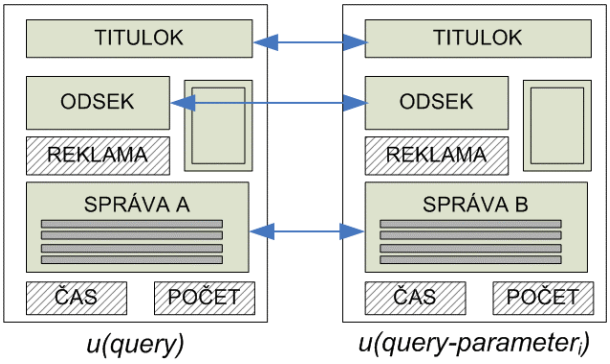
3 Podobnosť stránok

Podobnosť stránok je vyhodnocovaná na základe porovnávania HTML značiek a ich obsahu. V nástroji WebCrawler je použitý algoritmus implementovaný pomocou knižnice HTMLParser[4], ktorý iteratívnym spôsobom prechádza všetkými HTML značkami v časti <BODY> súčasne v oboch porovnávaných stránkach. Výsledkom algoritmu je počet rozdielnych HTML značiek *diff(D1,D2)* a celkový počet porovnávaných značiek *tags(D1,D2)*. Algoritmus porovnávania je naznačený v nasledujúcom pseudokóde:

[3] Uniform Resource Identifier (URI): Generic Syntax, http://www.ietf.org/rfc/rfc3986.txt

[4] HTML Parser projekt, http://htmlparser.sourceforge.net/

```
D1 = Načítaj(u(query));
D2 = Načítaj(u(query-parameteri));
diff = 0; tags = 0;
Pokiaľ (D1 alebo D2 má nespracovanú značku)
{
  Zn1 = D1.značka; Zn2 = D2.značka;
  Ak (Zn1 == Zn2 a Zn1.obsah == Zn2.obsah)
  Potom D1 a D2 posuň na ďalšiu HTML značku;
  Inak {
    Nájdi spoločnú značku s rovnakým obsahom
pre D1 a D2;
    Zvýš(diff);
  }
  Zvýš(tags);
}
```



Obr. 2. Analýza vplyvu parametra „parameter$_i$“ na obsah stránky pomocou porovnania obsahu stránok. Šípky znázorňujú časti obsahu, ktoré majú byť porovnávané. Vyšrafované časti obsahu predstavujú „časovo dynamické“ elementy, ktoré sa nezarátavajú do výpočtu vplyvu parametra.

Pri každom vyhodnotení parametra je nutné vyjadriť mieru časovej dynamickosti stránky preto, aby sme vylúčili vplyv generovaného informačného obsahu stránky nezávislého od parametrov. Táto miera je vyhodnotená porovnaním stránok stiahnutých z rovnakej adresy *u(query)*, ale v rôznych časových okamihoch. Koeficient vplyvu parametra $parameter_i$ možno vypočítať podľa nasledujúceho vzťahu.

$$impact(parameter_i) = \frac{diff(D_1, D_2)}{tags(D_1, D_2)} - \frac{diff(D_{t1}, D_{t2})}{tags(D_{t1}, D_{t2})}$$

kde D_1 je HTML dokument stiahnutý z adresy *u(query)*, D_2 je dokument stiahnutý z adresy *u(query-parameter$_i$)*, D_{t1} a D_{t2} sú dokumenty stiahnuté z adresy *u(query)* v čase *t1* a *t2*, pričom $t1 \neq t2$.

3.1 Redukcia parametrov URI adresy

Pokiaľ algoritmus sťahovania stránok WebCralwer nájde neznámy parameter, tak vyhodnotí koeficient vplyvu na výslednú stránku. Pokiaľ je tento koeficient menší ako vhodne stanovená prahová hodnota *T*, tak neznámy parameter bude označený ako významný pre požadovanú stránku, inak neznámy parameter bude označený ako nevýznamný. Výsledky označenia významnosti alebo nevýznamnosti parametra sa ukladajú ako mená parametrov *name$_i$* alebo *!name$_i$* pre aktuálne spracovávané webové sídlo *S*.

Pri sťahovaní stránky z adresy *u(query)* potom algoritmus vytvorenia optimalizovanej URI adresy vytvára novú URI *u(query')* tak, všetky parametre *query*, tak aby platilo

$$query' = query \cap S(name_i)$$

kde *S(name$_i$)* je množina parametrov označených ako významné pre webové sídlo *S*.

4 Vyhodnotenie

Pre testovacie a ladiace účely optimalizácie URI adries bol použitý portál Joomla[5], kde bola otestovaná základná funkcionalita. Implementácia navrhnutej metódy bola integrovaná do nástroja WebCrawler, ktorá pri spracovávaní neznámych parametrov vyhodnotila ich významnosť.

Pri sťahovaní bez použitia optimalizácie bolo stiahnutých 91 stránok a s použitím optimalizácie (pre prahovú hodnotu T = 0,1) bolo stiahnutých 72 stránok. Teda počet spracovaných stránok bol zredukovaný približne o 20%.

Na overenie výsledkov bolo spracovaných 500 stránok z (pre prahovú hodnotu T = 0,1 – teda stránky, ktoré sa líšili viac ako o 10% boli vyhodnotené ako odlišné):

- www.profesia.sk a bolo identifikovaných 12 významných a 26 nevýznamných parametrov z celkového počtu 38 použitých parametrov;
- www.amazon.com a bolo identifikovaných 15 významných a 26 nevýznamných parametrov z celkového počtu 41 použitých parametrov.

Jeden z podstatných výsledkov bolo 100% identifikácia nevýznamnosti parametrov *PHPSESSION* a *session_id* a zároveň významnosti parametrov ako napríklad *action, company_id, search, offer_id, docId, redirectTo, id* a podobných.

5 Záver

Navrhnutá metóda má za cieľ zjednodušiť a zefektívniť spracovávanie dynamických stránok a zároveň obmedziť počet potrebných URI požiadaviek na webový server, aj keď na identifikovanie významnosti parametra sú potrebné ďalšie URI dotazy. Tieto sú však ukladané do konfiguračného súboru, takže pri ďašom spracovaní už nie je nutné ich opätovné vyhodnotenie. Táto metóda však aktívne upravuje URI adresu, tak aby neobsahovala nevýznamné parametre a ako ukázalo počet dotazov na Webový server sa v prípade portálu Joomla zredukoval približne o 20%. Táto metóda môže aj čiastočne eliminovať parametre, ktoré môžu spôsobiť nekonečný cyklus pri spracovávaní konkrétneho webového sídla.

[5] Joomla projekt, http://www.joomla.org/

Poďakovanie

This work is supported by projects Commius FP7-213876, NAZOU SPVV 1025/2004, SEMCO-WS APVV-0391-06, VEGA 2/7098/27.

Literatúra

[1] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. Science, 280(5360):98, 1998.

[2] Yeh P. J., Li. J.T., Yuan S. M.: Tracking the Changes of Dynamic Web Pages in the Existence of URL Rewriting, Proc. Fifth Australasian Data Mining Conference, AusDM2006, Sydney, November, 2006.

[3] Baeza-Yates R. and Castillo R.: Crawling the Infinite Web: Five Levels Are Enough, Algorithms and Models for the Web-Graph, Lecture Notes in Computer Science, Volume 3243/2004, ISBN: 978-3-540-23427-2.

[4] Gatial E., Laclavík M., Balogh Z., Hluchý L.: Web Page Categorization in Process of Focused Crawling. In: INFORMATICS2007: proceedings of the ninth international conference on informatics. Bratislava. Slovak Society for Applied Cybernetics and Informatics, 2007, pp. 95-99. ISBN 978-80-969243-7-0.

Morfologie češtiny znovu a lépe

Jaroslava Hlaváčová and David Kolovratník

Univerzita Karlova v Praze
Ústav formální a aplikované lingvistiky, Matematicko-fyzikální fakulta
Malostranské náměstí 25, 11800 Praha, Česká republika

Abstrakt *Nový systém pro automatickou morfologickou analýzu češtiny [1], který zde prezentujeme, vychází ze současného pražského systému. Kromě nové implementace stávajícího slovníku pomocí konečného automatu přidává algoritmy na rozpoznání neznámých tvarů. V současné fázi je to především využití seznamu předpon, které lidé připojují bez větších omezení ke slovům a činí je tak pro automatické metody nerozpoznatelnými. Stačí předponu rozpoznat, odtrhnout a analyzovat zbylý řetězec. Výsledky této analýzy lze pak jednoduše aplikovat na původní tvar s předponou. K rozpoznání neznámých vlastních jmen používáme heuristiku. V příspěvku stručně popíšeme stávající pražský systém a novou implementaci včetně zárodku guessru.*

1 Úvod

Automatická morfologická analýza češtiny již byla řešena, dokonce několikrát. Nejznámější nástroje vznikly v Praze [3] a v Brně [2]. Oba systémy využívají rozsáhlé morfologické slovníky, na jejichž základě jsou schopny rozeznat většinu slovních tvarů vyskytujících se v českých textech. Počet nerozpoznaných slov, tedy slovních tvarů, které nelze odvodit ze slovníků, činí 2 až 3% (viz [4]). Rozšiřování slovníku většinou už nepomůže, naopak zvyšuje homonymii, která potom způsobuje problémy v dalších fázích zpracování textů, především při desambiguaci. Řešením je spíše použití tzv. guessrů, založených na vlastnostech českého tvarosloví, které umožňují s velkou přesností odhadnout charakteristiky (tj. lemma a hodnoty morfologických kategorií) nerozpoznaných slovních tvarů. V článku navrhujeme guesser pro rozpoznání tvarů odvozených pomocí předpony. Využíváme při tom vysoké pravidelnosti českého jazyka při tomto velice produktivním způsobu odvozování.

2 Reprezentace morfologického slovníku

Základní myšlenky implementace vycházejí z reprezentace morfologického slovníku použité J. Hajičem [3]. V následujícím stručně představíme jeho řešení.

Popis morfologie má tři části: (1) slovotvorné vzory, (2) tvaroslovné vzory a (3) vlastní morfologický slovník. Heslo slovníku obsahuje **slovotvorný začátek**, jméno jeho slovotvorného vzoru a **společné lemma**. Pomocí slovotvorného vzoru se generuje seznam (nějak příbuzných) **tvaroslovných začátků** s tvaroslovnými vzory, které v dalším kroku slouží k vytvoření konkrétních slovních tvarů. Každému slovnímu tvaru je na základě tvaroslovného vzoru přiřazena morfologická značka a základní slovní tvar (lemma). Uvedené vztahy přibližuje příklad na obrázku 1.

Kromě výše popsaných hesel slovník obsahuje i konkrétní slovní tvary, které nelze vygenerovat pomocí vzoru (např. substantivum *kůň* s nepravidelným skloňováním), a k nim příslušné morfologické značky.

Odvozování systémem vzorů spolu se slovníkem popisuje všechny možné slovní tvary a k nim příslušné značky a lemmata. Pomocí zmíněných tří částí popisu morfologie lze tedy všechny slovní tvary vygenerovat. To se však nedělá — bylo by to neefektivní. Již původní popis má dobré vlastnosti, které umožňují kompaktní reprezentaci, zachovávající zvlášť začátky a zvlášť vzory, snadno využít k analýze. Tvary se doplňují za pomoci (upravených) vzorů ze slovotvorných začátků až při samotné analýze za běhu.

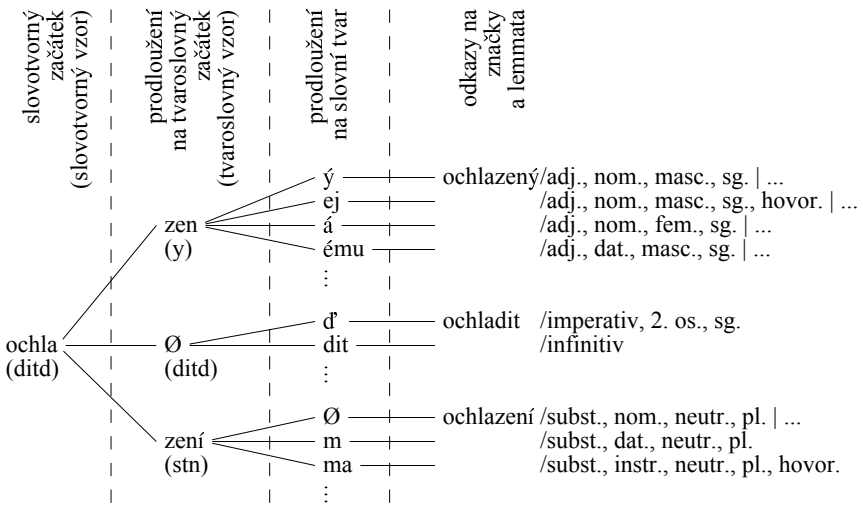
3 Morfologická analýza

Morfologická analýza je vlastně funkce, která každému slovnímu tvaru přiřadí množinu dvojic [lemma, morfologická značka]. Implementace tedy musí umět rozpoznat slovní tvar a na výstup poslat množinu všech možných příslušných dvojic.

3.1 Implementace analyzátoru

Všechny slovní tvary jsou v analyzátoru uloženy ve formě **trie**[1]. Kdybychom však do trie uložili rovnou celé slovní tvary, bylo by to příliš neefektivní. Z toho důvodu se do něj ukládají jen slovotvorné začátky převzaté ze slovníku. Z nich potom vedou dva odkazy. Jeden na složené schema (viz další odstavec), podle kterého se generují slovní tvary, druhý na řetězcovou instrukci, která ze slovotvorného začátku odvozuje **společné lemma** (to může později sloužit jako základ pro lemma analyzovaného tvaru). Řetězcová instrukce zde kompaktně reprezentuje společné lemma uvedené spolu se slovotvorným začátkem ve slovníku. Využívá toho, že si oba řetězce jsou často

[1] Trie chápeme jako (1,n)-četný zakořeněný strom. Hrany stromu jsou ohodnoceny znaky (případně řetězcem, dojde-li je kompresi). Uzlu stromu lze přiřadit slovo, které vznikne sřetězením znaků na cestě od kořene k danému uzlu. Taková slova označená příznakem tvoří množinu klíčů. Ke klíči je v trie uložena také hodnota.



Obr. 1. Představa odvozování slovních tvarů ze slovotvorného začátku.

velmi podobné. Slovotvorný začátek je uložen v trie celý a bude při analýze k dispozici jako výchozí řetězec pro instrukci.

Složené schema v sobě kombinuje oba stupně vzorů. Nahrazuje tak dvoukrokové (viz obrázek 1) odvození jednokrokovým. Každému slovotvornému schematu odpovídá jedno složené. To je organizováno jako hešovací tabulka. Jejími klíči jsou spojení konců pravidel slovotvorného vzoru a příslušných konců vzorů tvaroslovných (například -zen+ej, -zen+ého, -zen+ému, . . .).

Hodnotou vloženou ke klíči je opět dvojice odkazů. Jeden identifikuje skupinu značek, které popisují morfologické vlastnosti tvaru, druhý řetězcovou instrukci, která odvozuje lemma ze základu lemmatu. Základem je buď společné lemma nebo slovotvorný začátek. Jeden klíč může být vložen vícekrát s různými dvojicemi odkazů. Všechny výskyty přispívají k celkovému obrazu, doplňují tedy bohatost tvarosloví.

Řetězcová instrukce je trojice [*počet, řetězec, volič základu*]. Slouží ke kompaktnímu uložení řetězce, který sdílí začátek s nějakým známým, zvnějšku získaným, základem. Nový řetězec se získá (odvodí, spočítá) ze základu odmazáním *počtu* znaků od konce a připojením *řetězce*. Jednobitový *volič základu* vybírá jeden ze dvou možných základů. Toho se využívá pro odvození lemmatu — může vycházet ze společného lemmatu nebo slovotvorného základu. Například instrukce [3, „zení“, *společné lemma*] uložená ve složeném schematu ze společného lemmatu (známého základu) „ochladit“ odvodí lemma pro tvary podstatného jména „ochlazení“ odtržením tří znaků a připojením konce „zení“. Společné lemma však bylo dříve spočteno ze slovotvorného základu instrukcí [0, „dit“, *slovotvorný základ*]. Ta je uložena v trie u slovotvorného základu.

3.2 Vyhledání tvaru

Navržené rozdělení datových struktur umožňuje snadné nalezení tvaru. Tvar se postupně znak po znaku vyhledává v trie. V každém uzlu se testuje, zda je v něm nastaven příznak konce slovotvorného začátku a odkaz na složené schema. Pokud ano, hledá se dosud nepřečtený zbytek tvaru v hešovací tabulce prodloužení složeného schematu. Najde-li se, je již přečtená část slovotvorným základem. V hešovací tabulce se potom zjistí množina značek náležících k tvaru. Tímto způsobem se vyhledají všechny možné analýzy každého slovního tvaru.

Lemma je sestaveno pomocí řetězcové instrukce nalezené ve složeném schematu. Ta vychází buď ze slovotvorného začátku (ten je určen rozdělením tvaru na část nalezenou v trie a na zbytek) nebo ze základu lemmatu. Základ lemmatu se odvozuje ze slovotvorného základu řetězcovou instrukcí nalezenou v trie.

Dále tento algoritmus nazýváme **základní analýzou**. Ta je rozšiřována o schopnost rozpoznat předpony.

4 Předpony

4.1 Předpony podchycené ve slovníku

Ve slovníku nejsou předpony obecně nijak podporovány, kromě předpon negace (*ne-*) a superlativu (*nej-*). Negování i stupňování se provádí připojením příslušného morfému před slovo. Proto lze říct, že jistá podpora předpon ve slovníku je. Je však specializovaná.

Zvláštní podpora pro morfémy negace a superlativu ve slovníku má své opodstatnění: (1) negace a stupňování je potřeba i v případě, že není k dispozici jiná podpora předpon (třeba ta níže navrhovaná), (2) slovník přesně říká,

se kterými slovy lze morfémy superlativu a negace kombinovat, zatímco dále navrhované řešení předpon omezuje kombinace jen podle slovních druhů a nakonec (3) je to důsledkem toho, že o morfémech superlativu a negace se dozvídáme jinak než o ostatních předponách (slovník versus seznam předpon).

Předpony jsou v češtině běžnou a velmi užívanou součástí jazyka. Analyzátor, který by je zcela ignoroval, by byl stěží použitelný. Jsou tedy podporovány i bez navrhovaného rozšíření. Slova s předponou jsou uvedena ve slovníku jako samostatná hesla, bez uvedení slovotvorných souvislostí. Slova „čistit“ a „vyčistit“ tedy nejsou nijak propojena. Výhodou je opět zejména přesnost (správnost a citlivost) popisu.

4.2 Předpony zavedené seznamem

Námi navrhované rozpoznávání předpon má v mnohém opačné vlastnosti. Předpony jsou vypsány v samostatném seznamu zcela nezávisle na slovníku. Předpona se uvádí s **příznaky**, které říkají, se kterými slovními druhy se může pojit, např.

```
-dobro N                    -dopo NV
-dolno N                    -dovy NV
```

Příznak N znamená, že uvedená předpona dovoluje spojení se substantivy a adjektivy, příznak V se slovesy.

Mezi předponou a slovníkovým heslem není žádná jiná vazba než příznaky slovního druhu. Možnosti spojování nejsou nijak jinak omezeny. To se může jevit jako chybné řešení, protože s různými slovy lze kombinovat jen některé předpony. Nicméně to není problém, protože morfologická analýza si neklade za cíl vymezovat slovní zásobu jazyka („správná“ ani aktivně užívaná slova). Naopak, je snaha analyzovat co nejvíce slov.

Seznam předpon přichází na řadu až tehdy, když základní analýza pomocí slovníku tvar nerozpozná. Řešení pomocí seznamu umožňují použít dvě důležitá pozorování. Zaprvé je to skutečnost, že tvar se připojením předpony (většinou) nemění uvnitř[2]. Slovotvorný začátek, tvar i lemma se změní pouhým předřazením písmen předpony na začátek. A za druhé se nemění morfologické značky. Nemuselo by tomu tak být. Například předpony mění vid sloves, opět ale můžeme spoléhat na to, že všechny běžné případy jsou zahrnuty ve slovníku. Použití předpon má skutečně jen nabídnout řešení netradičních a nepodchycených slovních tvarů.

Zpočátku jsme chtěli používat analýzu pomocí předpon i u slov, která základní morfologická analýza rozpozná. Může se totiž stát, že některé tvary jsou homonymní s jinými, které slovník neobsahuje. Po prvních experimentech jsme ale od toho upustili, protože vycházelo velké množství falešně rozpoznaných tvarů. Například slovo „ves“ bylo rozděleno na předponu „v“ a substantivum „es“ (plurál od „eso“). Výsledná analýza potom běžnému tvaru „ves“ přiřadila navíc i nic neznamenající lemma „veso“. Podobných „veselých“ analýz se objeví celá řada.

[2] Tvary, kde ke změně dochází, jsou již zpravidla ve slovníku zachyceny.

4.3 Implementace předpon

Podmínkou pro rozpoznávání předpon je, že se při kompilaci nachystalo **trie předpon** (je volitelné). Trie se připravuje ze seznamu předpon pro odvozování. Ten se zadává jako zvláštní textový vstup kompilátoru.

Ačkoli trie začátků a trie předpon jsou tentýž typ datové struktury, staví a používají se nezávisle. Spojení by představovalo připojení trie začátků za všechny koncové uzly trie předpon. Protože by napojení bylo všude stejné, nepřineslo by analyzátoru žádnou dodatečnou informaci.

5 Analýza krok za krokem

5.1 Analýza čísel a interpunkce

V prvním kroku se morfologická analýza vypořádává s interpunkcí a posloupnostmi arabských číslic volitelně s desetinnou tečkou. Tyto jednoduché případy se řeší přímo, bez použití slovníku. Ostatní slovní tvary se zpracují následovně.

5.2 Základní analýza

Nejprve se řetězec tvaru použije celý tak, jak je, k sestupu do trie, jak bylo popsáno výše. To může vyústit

- v nalezení jednoho či několika lemmat spolu se značkami náležejícími ke tvaru, nebo
- v neúspěšnou analýzu (nenalezení žádného slovního tvaru v trie).

Základní analýza je procedura využívaná opakovaně v následujících krocích.

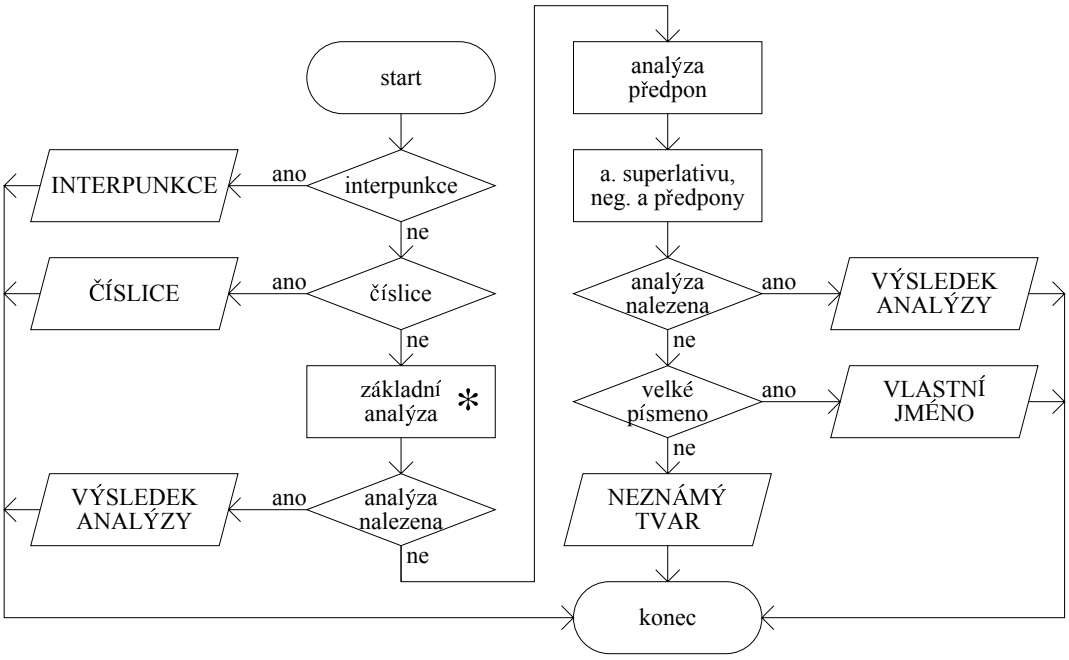
5.3 Negace a stupňování

Pokud tvar začíná morfémem negace a/nebo stupňování, jsou tyto morfémy rozpoznány a na zbylou část tvaru je vždy znovu spuštěna základní analýza.

Uvažují se dva morfémy samostatně a jedna jejich kombinace:

- negativní morfém „ne“, který neguje,
- morfém „nej“, který tvoří z komparativu superlativ, a
- kombinace obou tvořící negované superlativum.

Například slovo „nejnepořádnější“ je rozděleno a analyzováno čtyřmi způsoby:



∗ včetně analýzy případných morfémů negace a superlativu

Obr. 2. Vývojový diagram průběhu analýzy.

1. 0 + nejnepořádnější
2. ne + jnepořádnější
3. nej + nepořádnější
4. nejne + pořádnější

První rozdělení odpovídá algoritmu základní analýzy a provádí se vždy (už v předchozím kroku). Další jsou podmíněny přítomností písmen morfémů na začátku slova. V tomto případě uspěje jen poslední analýza, protože jedině komparativ „pořádnější“ je rozpoznán základní analýzou (za předpokladu, že „nepořádnější“ není ve slovníku jako samostatné heslo).

Uspěje-li vyhledávání nějakého rozdělení, může nebo nemusí být uznáno za platnou analýzu. Záleží na slovním druhu, zda jej lze kombinovat s uvažovanými morfémy. Po vyhledání konce tvaru, tedy když je znám slovní druh a značky, se filtrují výsledky.

Spojení morfémů s tvarem může být uznáno ze dvou důvodů. Zaprvé takové spojení mohou naznačovat žolíky ve značkách na pozicích popisujících negaci či stupeň a zadruhé může být obhájeno slovním druhem, dovoluje-li danou kombinaci. Negaci lze použít na afirmativní substantiva, adjektiva a slovesa, superlativa pak lze tvořit z komparativu adjektiv a adverbií.

Jsou-li rozpoznány jak negativní morfém tak morfém pro superlativum, musejí oba nalézt oporu v nějakém pravidle (viz předchozí odstavec). Když nenajdou, analýza se zahodí. Jinak je považována za přijatelnou. Značky se upraví, aby odpovídaly přítomným morfémům.

5.4 Odvozování předponou

Nebyla-li dosud nalezena žádná platná analýza, mohlo by se jednat o slovo odvozené předponou. Dojde-li na analýzu odvozovaní předponami, hledá se od začátku tvaru předpona v trie předpon. Je-li nějaká nalezena, pokračuje se hledáním zbytku zadaného tvaru v trie začátků a dále běží základní analýza. Najde-li se zbytek jako slovní tvar, prověřuje se, zda jde předpona se slovním druhem dohromady. Pokud ano, je analýza přijata, jinak je odmítnuta. Kombinace zkratek s předponami se nepřipouští.

Analýza pokračuje hledáním delších předpon (a kratších tvarů). Ty by mohly představovat jiné dělení (či odvození) slova.

Analyzátor předpon uvažuje jen jedinou předponu. Vícečetné kombinace se nehledají. Přicházejí-li v úvahu, musí být zadány jako samostatná předpona v seznamu. Tím se navíc podchytí správné uspořádání předpon. Například můžeme „dovysekat trávu před garáží“. Neříká se ale „vydosekat“.

Poté, co se vyčerpají možnosti jedné předpony, zkouší se ještě jedna cesta. Slovo by totiž mohlo být utvořeno předponou v kombinaci s morfémem negace či stupňování.

Morfém superlativu může předcházet negativnímu morfému a jeden nebo oba mohou předcházet odvozovací předponě a ta konečně bude následována tvarem (například „nej—ne—dovy—sekanější“). Slovní druh tvaru musí korespondovat se všemi nalezenými morfémy (mohou být uznány i na základě žolíků ve značce) i předponou. Není

tedy dovoleno např. spojení morfému „nej“ se substantivy. Opět se hledá jen jediná předpona.

5.5 Přehled analýzy

Obrázek 2 znázorňuje vývojový diagram průběhu analýzy.

6 Shrnutí a výhledy do budoucna

Představená nová verze morfologické analýzy již byla implementována. V základní formě poskytuje stejné výsledky jako zmiňovaná pražská verze morfologické analýzy. Je třeba ještě provést vyhodnocení výsledků předponového guessru, tj. zda a nakolik se zlepší rozpoznání slovních tvarů, které základní analýza na základě slovníku rozpoznat neumí. Dále plánujeme rozšířit guesser na koncovky a přípony, které jsou v českém jazyce také velmi produktivní.

Poděkování

Tato práce byla podpořena granty Informační společnosti č. 1ET101120503 a 1ET101120413 poskytovanými Grantovou agenturou AV ČR a grantem č. 100008/2008 poskytnutým Grantovou agenturou UK.

Reference

1. http://ufal.mff.cuni.cz/morfo/
2. Sedláček R.: Morphological analyser of Czech - ajka. http://nlp.fi.muni.cz/projekty/ajka/
3. Hajič, J.: Disambiguation of Rich Inflection. (Computational Morphology of Czech) Praha, Karolinum 2004.
4. Hlaváčová, J.: Morphological Guesser of Czech Words. Proc. TSD 2001, Springer-Verlag Berlin Heidelberg 2001. str. 70-75.

Evoluce parametrů páření

Tomáš Holan

KSVI MFF UK, Praha
Tomas.Holan@mff.cuni.cz

Abstrakt *Zkoumáme vývoj míry rodičovské investice a dalších parametrů páření a jejich závislost na globálních parametrech prostředí. K tomu používáme počítačovou simulaci prostředí a umělých bytostí a jejich evoluce. Text popisuje model prostředí, bytostí a jejich parametrů páření, provedené experimenty, problémy výpočtu a dosavadní výsledky.*

1 Úvod

Chceme vytvořit počítačovou simulaci prostředí, ve kterém žijí (pohybují se, interagují, množí se a hynou) umělé bytosti. Sledujeme přitom několik cílů.

Tím nejvzdálenějším cílem je snaha sledovat vývoj komunikace, Utváření výrazových prostředků, a jejich vývoj, od jednoduchých signálů snad až k vytvoření jazyka resp. jazyků. Simulované prostředí a simulované bytosti přitom budou jistě velice zjednodušeny, pokud chceme výpočetně zvládnout výpočet pohybu mnoha generací mnoha bytostí, musí být zjednodušeny až na úplné minimum. Proto i komunikace, kterou chceme sledovat, se musí týkat toho, co je pro živé bytosti nejnaléhavější — potravy a rozmnožování.

Abychom mohli posoudit, kde leží přípustná míra zjednodušení a zda máme naději dojít k nějakému výsledku, snažíme se nejprve vytvořit model, ve kterém se budou bytosti pohybovat v prostředí, kde roste potrava a budou se pářit na základě posouzení vhodnosti potenciálního partnera.

Jako kriteria sloužící k výběru partnera uvažujeme schopnost jedince získávat potravu (z této schopnosti plyne naděje, že i jeho potomci budou schopni se uživit) a míru rodičovské investice.

Rodičovská investice (například [1]) je újma, kterou živé bytosti postupují ve prospěch svých potomků, aby jim usnadnili přežití a reprodukci. Různé druhy živočichů vykazují různou výši rodičovské investice, a u téhož druhu mohou být odlišné formy i výše rodičovské investice u každého z pohlaví.

Vedlejším cílem našeho snažení je sledovat, jak se budou vyvíjet simulované populace, jak se budou měnit jejich parametry i případné „demografické" ukazatele, jako je průměrný počet dětí, výše rodičovské investice otců a matek, zda dojde k oddělení pohlaví a další.

Obzvláště nás potom zajímá, jak budou tyto parametry záviset na změnách globálních podmínek, například zda při menší dostupnosti potravy budou rodičovské investice menší, stejné nebo větší.

Text popisuje aktuálně implementovaný model prostředí a bytostí, první problémy a jejich řešení a první výsledky.

2 Model

2.1 Prostředí

Simulované prostředí je rozděleno do čtverečků, na kterých roste potrava, pro každý čtvereček s individuální pravděpodobností zadávanou před začátkem simulace jako mapa úrodnosti.

Potrava je jediného druhu, bytosti se živí pouze potravou, neexistuje potravní řetězec.

Smyslem individuální úrodnosti je dovolit modelovat vývoj na mapě, kde budou oblasti úrodnější i méně úrodné. Očekáváme, že se umělý život nejprve rozvine v oblastech úrodných a teprve později, až se u jedinců vyvine lepší schopnost hledat potravu, osídlí i oblasti méně úrodné.

Množství potravy je navíc ovlivňováno globálním nastavením úrodnosti, které dovoluje zkoumat změny v populaci i v parametrech jedinců při změnách v dostupném množství potravy i nastavení vyšší úrodnosti v počátečních fázích evoluce.

Parametry prostředí jsou:

- globální úrodnost (hodnota, kterou se násobí pravděpodobnost, s jakou na políčku vyroste potrava)
- množsví energie obsažené v jednotce potravy
- maximální povolená zásoba energie jedince
- využitelnost energie z potravy v závislosti na věku (staří jedinci spotřebovávají na jeden krok více energie a tak musí získávát více potravy)
- doba čekání potomka
- míry mutace algoritmu i parametrů páření

2.2 Bytosti

Bytosti jsou popsány proměnným stavem a neměnnou genetickou informací.

Stav bytosti sestává z pozice, orientace (jeden ze čtyř směrů), tím, jestli a jak dlouho se jedinec nachází ve stavu čekání potomka, a zásobou energie.

Genetická informace obsahuje algoritmus pohybu a parametry páření.

2.3 Algoritmus pohybu

V každém kroku simulace každá bytost, která není ve stavu, kdy čeká potomka, provede jednu z následujících akcí: otočit doleva, otočit doprava, krok vpřed, nedělat nic. Pokud má organismus paměť, může být s každou akcí svázán přechod do jiného stavu paměti.

O svých akcích se jedinec rozhoduje podle obsahu tří sousedních políček – nalevo, před sebou a napravo (za sebe nevidí), případně stavu paměti.

Protože na každém pozorovaném políčku bytost rozlišuje jen několik možných obsahů (prázdné, potrava, jedinec vhodný k páření, jedinec nevhodný k páření), lze celý algoritmus popsat jako funkci, kterou můžeme reprezentovat trojrozměrným resp. čtyřrozměrným (stav paměti) polem.

Pokud jedinec vkročí na políčko, kde je potrava, zkonzumuje ji a získá množství energie dané globálním parametrem prostředí.

V každém kroku organismus spotřebovává určité množství energie, pokud nemá dostatek energie, hyne. Abychom podpořili generační výměnu, množství spotřebovávané energie roste s věkem jedince.

Na počátku jsou bytosti náhodně rozmístěny a mají určité počáteční množství energie. Nově narozená bytost dostává určité množství energie od rodiče, toto množství je jedním z parametrů páření.

2.4 Páření

Organismy se rozmnožují pářením, podmínkou páření je vzájemné splnění podmínek páření.

Podmínky páření každého jedince zahrnují:

- dolní limit energie vlastní
  (když jedinec hladoví, nemnoží se)
- dolní limit energie partnera
  (ti, kdo si neumí najít potravu, nebudou vhodnými partnery)
- množství energie předávané narozenému potomkovi
  (rodičovská investice matky)
- množství energie požadované po partnerovi
  (poplatek za páření, rodičovská investice otce)
- požadavek na rodičovskou investici partnera
  (chci partnera, který ma v genech dostatečnou rodičovskou investici)
- horní limit energie předané partnerovi
  (kolik je ochoten zaplatit)

Poslední dva parametry mají význam pouze u *jedince vyvolávajícího páření (otec)*, předposlední dva parametry mají význam pouze u *jedince přijímajícího páření (matka)*, nicméně jsou součástí genetické informace všech jedinců.

Průběh páření: Jedinec vyvolávající páření (otec) vyvolá páření akcí **krok vpřed** na políčko, kde stojí jedinec přijímající páření (matka), přitom se jeho poloha nezmění a v případě vzájemného splnění podmínek páření

- dojde k předání energie od otce k matce, množství je určeno parametrem *množství energie požadované po partnerovi* matky (poplatek za páření)
- otec může příštím tahem pokračovat
- matka K tahů *čeká* a potom na náhodně vybraném nejbližším volném políčku vznikne nový jedinec, K je parametr prostředí
- nový jedinec dostane od matky energii v množství daném jejím parametrem (investice do potomka).
- genetická informace nového jedince vznikne zkřížením a mutací genetické informace rodičů.

Při přiblížení organismů se vyhodnotí vzájemné splnění podmínek páření a podle toho se každý organismus ve výhledu jiného jedince jeví jako *jedinec vhodný k páření* nebo *jedinec nevhodný k páření*. Jedinec, který *čeká*, je *jedinec nevhodný k páření*.

V této versi modelu nejsou oddělena pohlaví, každý jedinec proto může vystupovat v roli otce i v roli matky.

Energii potomkům předává pouze jeden rodič (matka), rodičovská investice otce je ale zajištěna zprostředkovaně prostřednictvím poplatku za páření.

3 První výpočty, první zkušenosti

Popsaný model jsme implementovali, implementace dovoluje práci v režimu, kdy zobrazuje prostředí a všechny jedince, i v režimu, kdy pouze počítá a vypisuje výsledky (vyšší výkon).

Velikost simulovaného prostředí je určena velikostí načtené mapy úrodnosti. Pro první pokusy jsme zvolili prostředí velikosti 32x24 jako kompromis mezi rychlostí výpočtu a tím, že na menší mapě populace nedokázaly přežít.

Křížení genetické informace probíhá mnohabodově (bez zvláštního důvodu), každý krok algoritmu pohybu a každá hodnota parametrů páření jsou přebírány náhodně od otce nebo od matky.

Míra mutace je rozdělena do dvou parametrů, zvlášť pro každou část genetické informace.

O nastavení míry globální úrodnosti se zmíníme vzápětí spolu s popisem problémů, jedinci mají omezitelnou výši energetických zásob, aktuálně heuristicky nastavenou na množství potřebné pro padesát kroků mladého jedince.

Během těchto experimentů jsme narazili na určité problémy, nyní popíšeme jejich řešení.

3.1 Přežití

Na začátku simulace jsou do prostředí náhodně umístěni náhodně vygenerovaní jedinci.

Protože algoritmus pohybu i parametry páření jsou také náhodně nastaveny, většina jedinců nedokáže hledat potravu a není schopná se pářit, kvůli nevhodnému nastavení svých parametrů i proto, že je jejich algoritmus nevede k páření.

Proto během prvních kroků většina populace vyhyne a zůstanou pouze ti jedinci, kteří se umí pohybovat alespoň natolik, aby si našli nějakou potravu.

Ti z nich, kteří umí hledat potravu, ale nedokáží se spářit, vyhynou také, protože s rostoucím věkem budou potřebovat na doplnění energie stále více potravy.

Pokud přežije dostatečný počet jedinců schopných páření i hledání potravy, začne se populace opět rozmnožovat, až dosáhne počtu, který bude v rovnováze s tím, jak v prostředí přibývá potrava.

To ale znamená, že u takto fungujícího modelu v budoucnosti bude většina populace potomky několika málo (podle velikosti mapy, v našem případě dvaceti až padesáti) jedinců, což má špatný dopad na diversitu populace.

Možností, jak se této situaci vyhnout, je několik: opakování výpočtů, doplňování novými jedinci (migranti), nakonec jsme zvolili počáteční nastavení globální míry úrodnosti potravy na nejvyšší mez a její postupné snižování v době, kdy už přežilo několik generací a tedy se jedinci dovedou pářit. Toto řešení jsme použili pro další popisované experimenty, i když si jsme vědomi toho, že počátečním upřednostňováním páření před hledáním potravy pravděpodobně ovlivňujeme výsledek evoluce.

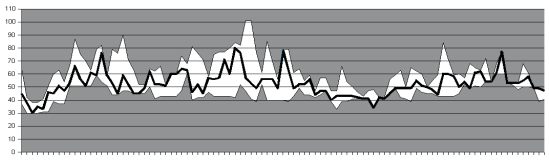
3.2 Rozmanitost jedinců i populace v čase

Chceme sledovat hodnoty vybraných parametrů umělých bytostí po určité době vývoje. Jedinci v populaci mají ale pro tentýž parametr různé hodnoty, takže se musíme rozhodnout, co budeme sledovat, resp. co budeme považovat za výslednou hodnotu parametru.

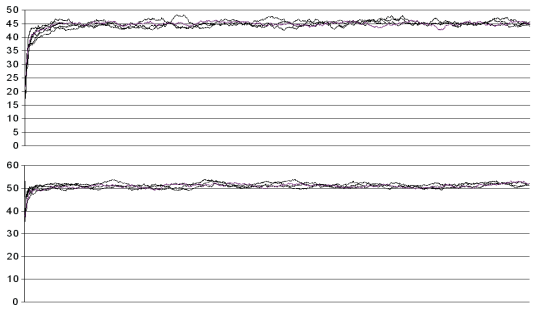
V počátcích experimentů jsme sledovali histogram resp. procentuální zastoupení jednotlivých hodnot, ale i to bylo obtížně uchopitelné a porovnatelné, takže nakonec po každý parametr počítáme medián, a navíc ještě zachycujeme 25 a 75-procentní percentily. To nám dává příležitost nahlédnout, do jaké míry jsou hodnoty parametrů v populaci rozptýleny.

I tyto hodnoty však v populaci oscilují a sledujeme-li okamžité hodnoty percentilů u několika populací zároveň, jsou mezi nimi poměrně velké rozdíly.

Když ale místo okamžitých hodnot začneme sledovat klouzavý průměr, můžeme ve všech populacích pozorovat vývoj k podobným hodnotám a navíc vidíme, že k dosažení těchto hodnot stačí menší počet kroků.



Obr. 1. Vývoj parametru Poplatek za páření: medián mezi 25- a 75-procentními percentily.



Obr. 2. Vývoj parametru Investice a Poplatek za páření, klouzavé průměry (z posledních 100 hodnot) mediánů ze 6 populací v průběhu 20 milionů kroků.

3.3 Nečekané vyhynutí

Čekali jsme, že pokud simulaci necháme běžet například 10 milionů kroků (věk bytostí se pohybuje v jednotkách desítek), možná dokonvergují všechny populace ke stejnému výsledku.

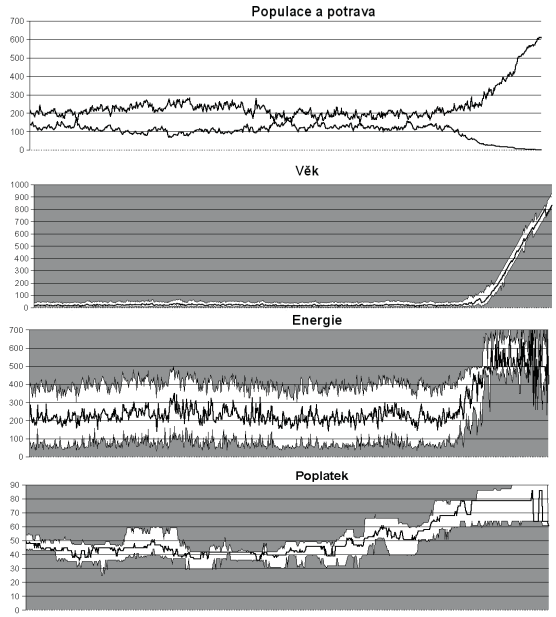
To se určitým způsobem opravdu stalo – z generací, které přečkaly milion kroků, žádná nepřežila deset milionů kroků, v různých okamžicích (od cca 1 milionu do 5 milionů kroků) u všech populací došlo k tomu, že během přibližně deseti tisíc kroků vyhynuly.

Když hovoříme o tom, že populace vyhynula během deseti tisíc kroků, máme na mysli to, že během této krátké doby po předchozím, zdánlivě ustáleném vývoji dojde k výrazným změnám parametrů:

- růst věku jedinců, zhruba na dvacetinásobek
- růst energetických zásob jedince, zhruba na trojnásobek
- růst poplatku za páření (investice otců)

Vytipovali jsme několik možných vysvětlení:

- Při dostatku energie mohou někteří jedinci získávat energii převážně nebo dokonce výhradně z poplatku za páření. Takto získaná energie jim dovoluje úspěšné páření, aniž by zaručovala kvalitní algoritmus hledání potravy.
- Potrava roste s pravděpodobností odpovídající nastavení globálního parametru úrodnosti. To znamená, že potrava roste nezávisle na spotřebě (téměř nezávisle,



Obr. 3. Vývoj parametrů jedinců během posledních 10.000 kroků před vyhynutím: Populace (roste) a Potrava (klesá), Věk, Energie jedince a Poplatek (25, 50 a 75 procentní percentily.)

potrava roste pouze na políčkách, na kterých ještě není, takže když ji nikdo nespásá, neroste).
To ale dovoluje, aby rostla celková energie obsažená v simulovaném světě (součet energie v potravě a energie v individuálních zásobách bytostí) a to může vést k tomu, že poplatek za páření přestane hrát omezující roli.

– Jev, který Dawkins [2] nazývá „závody ve zbrojení". Požadavky na poplatek za páření splní pouze partneři s vysokou energií, tedy kvalitnější partneři, tedy je dobré poplatek zvyšovat. Dokud nedojde k hodnotě, která bude nesplnitelná. Pokud jsou si parametry jedinců v populaci podobné, může to vést k vyhynutí celé populace.

Problém vyhynutí nakonec upravila úprava formy i pravděpodobnosti mutace podmínek páření, i když samozřejmě stále nemáme jistotu, že populace zčistajasna nevyhyne někdy v budoucnosti.

4 První výsledky

Následující výsledky byly spočteny během výpočtů trvajících 5.000.000 kroků jako klouzavé průměry ze 100 mediánů zjišťovaných vždy po 10.000 krocích. Uvádíme interval hodnot naměřených během pěti opakování experimentu.

Při míře úrodnosti potravy 0.08 a době čekání 3 kroky byla velikost populace **2086–2124**, investice do potomků v rozmezí **26.6–27.7** a poplatek za páření **33.4–35.0**. Investice matek tedy byla menší než energie vyinkasovaná od otce.

Při snížení úrodnosti potravy na polovinu (0.04) a stejné hodnotě ostatních parametrů prostředí byla výsledná velikost populace **1390–1436**, investice do potomků v rozmezí **33.2–35.8** a poplatek za páření **38.0–40.0**. Při nižší dostupnosti potravy tedy investice do potomků u obou rodičů stoupla.

5 Závěr

Navrhli a implementovali jsme model umělého světa, který dovoluje simulovat evoluci umělých bytostí a experimentovat s jejími parametry a omezeními.

Provedli jsme první experimenty, vyřešili první problémy a spočetli první výsledky, a nyní máme nástroj na hledání odpovědí na otázky jako: jak souvisí rodičovská investice obou rodičů nebo průměrný věk s dostupným množstvím potravy, s délkou čekání, jaký je v populaci poměr otců a matek, jaký je u matek poměr energie získané z potravy a energie získané za páření (a nepředané dětem), co se stane, když omezíme maximální množství energetických zásob, nebo oddělíme pohlaví. . . a další.

Reference

1. Trivers, R.L.: Parental investment and sexual selection. In B. Campbell (Ed.), Sexual selection and the descent of man, 1871-1971 (pp. 136-179). Chicago, IL: Aldine.
2. Dawkins, R.: Slepý hodinář (Zázrak života očima evoluční biologie), Paseka 2002

Ovládanie mobilného telefónu prostredníctvom počítača cez rozhranie Bluetooth

Matej Jurikovič and Peter Pištek
Fakulta informatiky a informačných technológií
Slovenska technicka univerzita v Bratislave
Ilkovičova 3, 842 16 Bratislava 4, Slovenská republika

Abstrakt. *Témou príspevku je možnosť ovládania funkcií mobilného telefónu prostredníctvom počítača použitím rozhrania Bluetooth. V dnešnej dobe má každý mobilný telefón vlastný softvér pre komunikáciu a ovládanie niektorých funkcií pomocou počítača. Našim cieľom je vytvorenie univerzálneho riešenia pre čo najviac typov mobilných telefónov, ktoré by poskytovalo plnohodnotnú prácu a ovládanie. Keďže väčšina ľudí preferuje mobilné riešenia, rozhodli sme sa pre využitie štandardného komunikačného protokolu Bluetooth. Naše riešenie využíva virtuálny sériový port, pomocou ktorého komunikujeme s mobilným telefónom bez nutnosti využívať dodatočný softvér na strane telefónu. Pre ovládanie niektorých funkcií využívame AT príkazy, ktoré sú zapisované na sériový port. V našej práci je popísaná problematika komunikácie pomocou sériového portu s mobilným telefónom, s využitím AT príkazov. Jednou z funkcií, ktoré náš softvér poskytuje je aj práca s SMS správami. Detailne popisujeme PDU formát, v ktorom sú tieto správy kódované.*

1 Úvod

V súčasnosti má každý mobilný telefón vlastný softvér, ktorý do rôznej miery umožňuje jeho ovládanie z počítača. Väčšinou umožňujú len základnú prácu ako zálohovanie SMS správ, prácu zo súbormi. My sme sa rozhodli vytvoriť univerzálne riešenie pre čo najväčšie množstvo typov mobilných telefónov. Veľa ľudí dáva prednosť bezdrôtovým riešeniam komunikácie, ktorá je viac mobilná a poskytuje väčšie pohodlie. Z tohto dôvodu sa javí ako najlepšie pre komunikáciu technológia Bluetooth alebo IrDa. Nevýhodou IrDa je malá vzdialenosť medzi komunikujúcimi zariadeniami a nutná priama viditeľnosť. Preto ako najpohodlnejšie riešenie pre užívateľa pôsobí Bluetooth.

2 Úvod do Bluetooth

Bluetooth je štandardný komunikačný protokol, primárne navrhnutý pre zariadenia s nízkym odberom energie. Slúži na prenos dát na krátke vzdialenosti (v závislosti od odberu energie: 1 meter, 10 metrov, alebo 100 metrov) za pomoci vysielačov/prijímačov umiestnených v každom zariadení. Bluetooth umožňuje týmto zariadeniam komunikovať pokiaľ sú na dosah. Zariadenia používajú rádio komunikačný systém, preto nemusia byť vzájomne priamo viditeľné v priebehu komunikácie. Bluetooth využíva nelicencovanú frekvenciu 2.4 GHz. Jednotlivé bluetooth triedy určujú typ zariadenia a podporované služby, pričom tieto informácie sa prenášajú počas procesu nachádzania (discovery process). [1]

2.1 Bežné využitie Bluetooth

Bezdrôtová kontrola komunikácie medzi mobilným telefónom a headsetom. Toto bolo jedno z prvých využití, ktoré spopularizovali Bluetooth. Ďalšími sú bezdrôtová komunikácia medzi počítačmi na malom priestore, kde nie je nutná veľká šírka prenosového pásma, bezdrôtová komunikácia medzi počítačmi a vstupno-výstupnými zariadeniami.

Prenos súborov medzi zariadeniami prostredníctvom OBEX protokolu, prenos kontaktov, synchronizácia kalendára a pripomienok prostredníctvom protokolu OBEX. Nahrádza tradičnú sériovú komunikáciu, GPS prijímače, medicínske vybavenie, skener čiarových kódov a prístrojov na kontrolu premávky.

2.2 Nadviazanie spojenia

Ľubovoľné zariadenie môže vykonať prieskum za účelom nájdenia pripojiteľných zariadení. Ľubovoľné zariadenie môže byť nastavené na odpoveď na takýto prieskum. Avšak, ak zariadenie snažiace sa o pripojenie vie adresu zariadenia, na ktoré sa snaží pripojiť, posiela mu priame komunikačné požiadavky. Využitie zariadenia môže vyžadovať spárovanie zariadení alebo potvrdenie požiadavky jeho majiteľom, ale v princípe môže byť komunikácia samotná nadviazaná medzi zariadeniami a byť udržiavaná pokiaľ sa nedostanú mimo dosah.[11]

Každé zariadenie ma unikátnu 48-bitovú adresu. Avšak adresa sa zvyčajne nezobrazuje počas hľadania zariadení, namiesto toho sa zobrazuje Bluetooth meno zariadenia, ktoré môže byť nastavené užívateľom.

2.3 Párovanie

Dvojica zariadení môže nadviazať dôveryhodné spojenie pomocou naučenia sa (užívateľského vstupu) spoločného tajného kľúča známeho ako univerzálny kľúč. Zariadenie, ktoré chce komunikovať iba s dôveryhodnými zariadeniami môže šifrovať autentifikáciu pre iné zariadenia. Dôveryhodné zariadenia môžu rovnako tak šifrovať aj dáta prenášané vzduchom, takže ich nikto nemôže odpočúvať. Šifrovanie môže by aj vypnuté a univerzálny kľúč je uložený v súborovom systéme zariadenia, nie na samotnom Bluetooth čipe. Odkedy sú Bluetooth adresy stále a párovanie nastavené, dokonca sa môže meniť Bluetooth meno zariadenia. Párovanie môže byť vymazané v ľubovoľnom čase jedným zo zariadení. Zariadenia zvyčajne v dnešnej dobe vyžadujú párovanie alebo potvrdenie predtým ako nadviažu spojenie alebo povolia vzdialený prístup k ich službám. Niektoré zariadenia ako napríklad telefóny Sony Ericsson zvyčajne akceptujú OBEX vizitky a poznámky bez nutnosti párovania alebo potvrdzovania. [11]

3 Sériová komunikácia

Z hľadiska ISO/OSI modelu prepojovania otvorených systémov možno povedať, že sériový port využíva fyzickú, linkovú a z časti transportnú vrstvu.

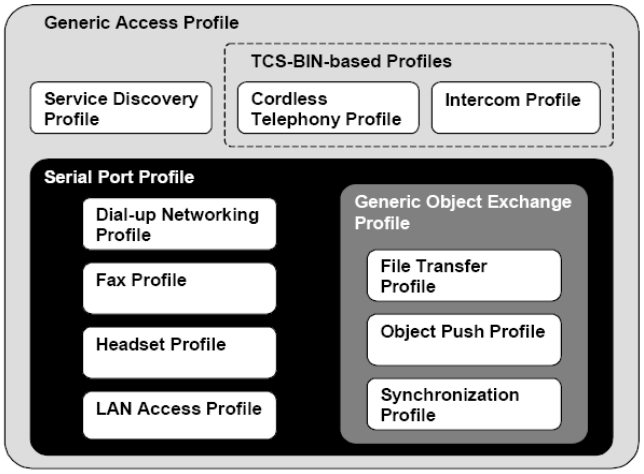
Štandard RS-232 definuje napätia a prenosové rýchlosti medzi zariadeniami, ktoré ho používajú. Podľa štandardu RS-232 sú definované dve zariadenia, ktoré môžeme pomocou sériového kábla prepojiť a to Data Terminal Equipment (DTE) a Data Communications Equipment (DCE). V našom prípade skratka DTE reprezentuje počítač a skratka DCE zariadenie, ktoré je s ním prepojené prostredníctvom sériovej linky.

3.1 Virtuálny sériový port

Virtuálny sériový port je jedným z variantov programu zaisťujúceho presmerovanie COM portu. Presmerovanie COM portu je špecializovaný ovládač pre operačné systémy Microsoft Windows. Virtuálne sériové porty dokážu emulovať všetky funkcie sériového portu ako prenosovú rýchlosť, dátové bity, paritné bity, stop bity, atd. Výhodou virtuálneho sériového portu je, že cez tento port je možné posielať a prijímať dáta prostredníctvom Bluetooth.

4 Štruktúra profilov

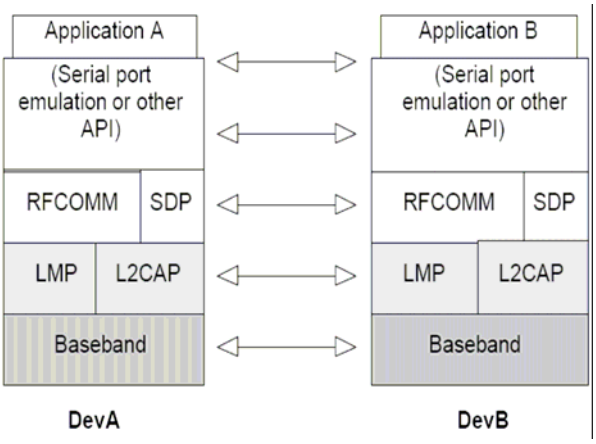
Na Obr. 1 je zobrazená štruktúra bluetooth profilov a ich závislostí. Profil je závislý na ďalšom profile pokiaľ používa jednotlivé časti iného profilu.



Obr. 1. Profily Bluetooth.

Na ďalšom obrázku je znázornený model protokolov.

Baseband, LMP a L2CAP sú bluetooth protokoly OSI vrstiev jedna a dva. RFCOMM je bluetooth adaptácia GSM TS 07.10 a poskytuje transportný protokol pre emuláciu sériového portu. SDP je Bluetooth Service Discovery Protocol. Vrstva emulácie portu, zobrazená na obrázku dva, je entita emulujúca sériový port, alebo poskytujúca API aplikáciam.[6]



Obr. 2. Model protokolov.

5 AT príkazy

Spoločnosť Hayes Communications v roku 1977 predstavila nový produkt Smartmodem, ktorý vyriešil problém vtedajších modemov, ktoré pre vytáčanie čísel používali samostatné periférne zariadenia. Tento nový modem sa dokázal prepnúť do dvoch módov :

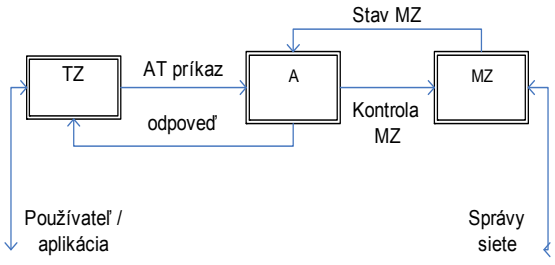
Dátový mód: V tomto móde sú informácie, ktoré modem prijme reprezentované ako dáta.

Príkazový mód: V tomto móde sú prijaté informácie reprezentované ako príkazy.

Vďaka tomuto prístupu bolo možné pristupovať k funkciám modemu jednoduchšie. Na prepnutie z dátového módu do príkazového módu sa používa prerušovacia sekvencia („+++“). Spolu s týmto modemom bola predstavená aj skupina príkazov pre ovládanie niektorých funkcií modemu. Skoro všetky príkazy začínajú dvojpísmenovou sekvenciou AT – pre získanie pozornosti modemu (AT - attention). Z tohto dôvodu sa často nazývajú AT príkazy.

5.1 Architektúra

Architektúra (viď Obr. 3) špecifikuje profil AT príkazov, používaných pre ovládanie funkcií mobilného zariadenia (MZ) pomocou terminálového zariadenia (TZ) cez adaptér (A). Jednotlivé časti tejto architektúry (MZ, TZ, A) sú reprezentované ako samostatné entity.

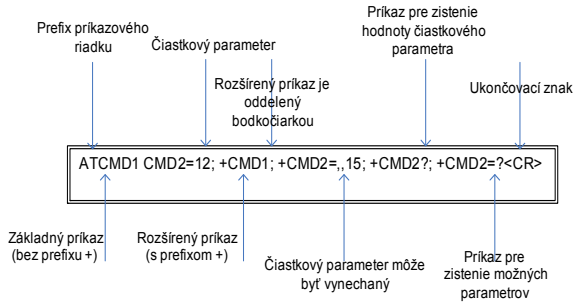


Obr. 3. Architecture of the AT commands.

Rozhranie medzi TZ a A komunikuje pomocou sériového káblu, infračervenej linky, respektíve pomocou ľubovoľného sériového rozhrania. Pre korektnú prácu väčšina AT príkazov vyžaduje 8-bitové dáta a z tohto dôvodu je odporúčané nastaviť rozhranie (TZ - A) na 8-bitový mód.

5.2 Syntax

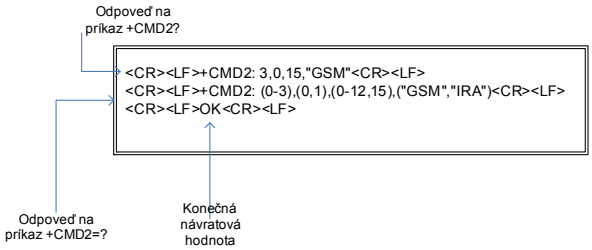
AT príkazy sú spájané v príkazovom riadku. Každý riadok musí začínať prefixom AT a musí byť ukončený znakom CR (carriage return). Telo príkazu pozostáva z viditeľných ASCII znakov. Znaky, ktoré predchádzajú AT prefixu sú ignorované. Znaky, ktoré sú za prefixom AT a pred ukončovacím znakom (CR) sú považované za príkaz. Na nasledujúcom obrázku je zobrazená základná štruktúra príkazového riadku.[4]



Obr.4. Základná štruktúra príkazového riadku s AT príkazom.

Ak boli všetky príkazy v príkazovom riadku spracované úspešne, návratová hodnota <CR><LF>OK<CR><LF> je poslaná z adaptéra do terminálového zariadenia. V prípade, že príkaz nie je akceptovaný mobilným zariadením (poprípade príkaz je nesprávny), návratová hodnota je <CR><LF>ERROR<CR><LF>.[4]

Na nasledujúcom obrázku je zobrazená odpoveď na príklad príkazu z obrázka č.4.



Obr.5. Odpoveď z príkladu na obrázku č.4.

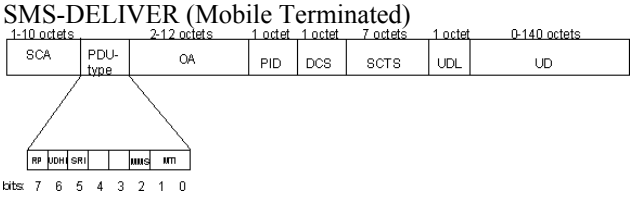
Pomocné návratové hodnoty informujú o progrese adaptéru (napríklad vytvorenie spojenia je signalizované reťazcom CONNECT) a nevyžiadané návratové hodnoty informujú o udalostiach, ktoré nie sú priamo spojené so zadávanými príkazmi (napríklad indikácia zvonenia je signalizovaná reťazcom RING).[4]

6 PDU formát

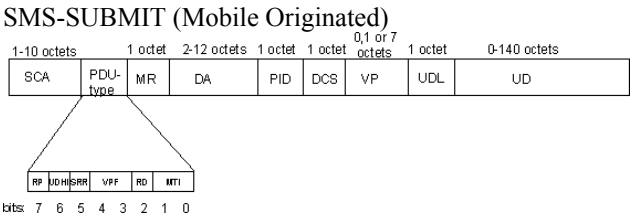
SMS správy, tak ako špecifikuje organizácia ETSI môžu byt 160 znakov dlhé, ak každý znak je posunutý do 7-bitovej abecedy. Osem bitové správy (max. 140 znakov) zvyčajne nie sú prezerateľné na mobilnom telefóne ako textové správy.

PDU (protocol description unit) mód ponúka možnosť poslať binárnu informáciu v sedem alebo osem bitovom formáte. Toto je veľmi užitočné pokiaľ je potrebné poslať komprimované dáta, binárne dáta alebo si chcete vytvoriť svoje vlastné kódovanie znakov v binárnom prúde. [7]

Sú dva spôsoby na posielanie a prijímanie SMS správ: textový mód a PDU mód. Textový mód (nedostupný na niektorých telefónoch) je len kódovanie bitového prúdu reprezentovaného PDU módom. Znaky sa môžu líšiť v niekoľkých kódovacích alternatívach pri zobrazovaní SMS správ.



Obr. 6. SMS-deliver.



Obr. 7. SMS-submit.

Jednoduchý príklad na PDU formát: (“hello world”)
0001000C9124913041818400000BE8329BFD06DDDF723619

00 – Dĺžka informácie Service Center Address. V tomto prípade sa použije číslo uložené v telefóne.

01 – PDU typ. Prvý oktet odosielanej SMS.

00 – Referenčné číslo posielanej správy. Priradí telefón

0C – Dĺžka telefónneho čísla príjemcu (počet čísel, nie oktetov)

91 – Formát telefónneho čísla (medzinárodné- 91, národné 81).

249130418184 – Telefónne číslo príjemcu (+421903141848)

00 – Určenie formátu a protokolu pre doručenie správy

00 – Informácia o kódovaní správy

0B – Počet znakov správy pred jej zakódovaním.

E8329BFD06DDDF723619 – text „hello world“

7 Program

Cieľom projektu bolo vytvorenie aplikácie umožňujúcej vzdialené ovládanie mobilného telefónu.
Vstupy programového systému sú :

- Číslo vytvoreného virtuálneho sériového portu.

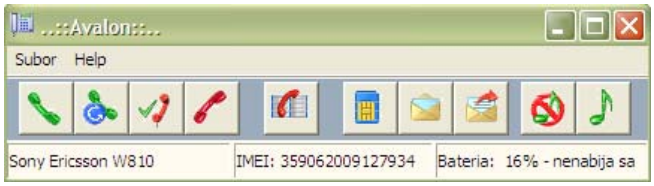
- Telefónne číslo na ktoré sa má uskutočniť hovor, poprípade poslať SMS správa.
- Súbor, ktorý sa má preniesť na mobilný telefón.
- Text posielanej SMS správy.

Výstupom programového systému sú :

- Informácie o pripojenom zariadení (značka telefónu, IMEI číslo, stav batérie, atd.).
- Výpis aktuálnych hovorov.
- Výpis SMS správ načítaných z mobilného telefónu.
- Stiahnutý súbor z mobilného telefónu.

Program komunikuje s mobilným telefónom prostredníctvom technológie Bluetooth, pričom komunikácia prebieha pomocou virtuálneho sériového rozhrania.

V prvom kroku je nutné spárovať mobilný telefón a počítač. Počítač pridelí číslo virtuálneho sériového portu. Pri spustení programu sa zvolí tento port, ktorý sa následne otvorí.



Obr. 8. Hlavné okno.

Na komunikáciu s mobilným telefónom sú použité spomínané AT príkazy. Prostredníctvom týchto príkazov je ovládaný mobilný telefón a získavajú sa požadované informácie. Program umožňuje kontrolovať stav mobilného telefónu ako napr. Stav batérie, signálu, nové SMS atď.

Čítanie informácií zo sériového rozhrania zabezpečuje samostatné vlákno programu, ktoré poskytuje plynulosť používania aplikácie.

Pre SMS sme sa rozhodli implementovať univerzálny avšak náročnejší PDU formát, ktorý podporujú všetky mobilné telefóny aj starších typov a umožňuje interpretáciu rôznych štandardov vyšších vrstiev. Využitie PDU nespočíva len v čítaní SMS správ ale aj v ich posielaní. Pri zadávaní telefónneho čísla sa musí používať medzinárodný štandardný formát (napr. +421915151515).

Obr. 9. Ukážka načítaných SMS.

Program je vyvíjaný v jazyku C++ pre platformu Windows.

Pre každý podporovaný typ mobilu je vytvorený samostatný objekt, ktorý obsahuje sadu podporovaných AT príkazov a funkcií pre prácu s nimi. Nie sú použité žiadne dodatočné knižnice, sú používané základné systémové volania a štandardné funkcie. Tento prístup umožňuje nezávislosť od licenčných podmienok už iných komerčných riešení. Vytvorený program je viacvláknový, čo umožňuje kontrolovanie niektorých stavov mobilného telefónu bez "zamŕzania" aplikácie.

8 Zhrnutie

Program kontroluje a ovláda základné funkcie mobilného telefónu. Nie je nutný žiadny dodatočný softvér od výrobcu danej značky mobilného telefónu. Medzi možné rozšírenia aplikácie, ktorú by sme naďalej chceli vyvíjať, je podpora širšej ponuky mobilných telefónov. Tiež by sme chceli do budúcnosti zahrnúť podporu práce so súbormi, rovnako tak aj s kontaktmi uloženými v telefóne, ktoré sú založené na protokole OBEX. Hlavným cieľom je vytvorenie programu, ktorý by umožňoval plnohodnotnú prácu (najmä samotné telefonovanie) s mobilným telefónom bez nutnosti odísť od počítača.

Poďakovanie

Článok bol vypracovaný v rámci riešenia národného projektu VEGA VG 1/3104/06 (Systémy gridového počítania a jeho komponenty).

Referencie

[1] Bluetooth Special Interest Group, IrDA Interoperability.
[2] Bluetooth Special Interest Group, Generic Object Exchange Profile.
[3] Bluetooth Special Interest Group, File Transfer Profile.
[4] Bluetooth Special Interest Group, Object Push Profile.
[5] Bluetooth Special Interest Group, Baseband Specification.
[6] Bluetooth Special Interest Group, LMP Specification.
[7] A technical introduction to SMS messaging, http://www.activexperts.com/activsms/sms/technical/
[8] Global system for mobile communication - SMS PDU-Mode, dostupný na: http://www.gsmworld.it/frame.asp?URL=http://www.gsmworld.it/smspdu.htm
[9] Developers Guidelines - AT Commands Online Reference, dostupný na: www.sonyericsson.com/downloads/dg_at_2004_r6a.pdf
[10] Support Guide for the Nokia Phones and AT Commands, dostupný na: nds1.nokia.com/phones/files/guides/Nokia_AThelp.pdf
[11] Specification of the Bluetooth System, dostupný na: http://www.tscm.com/Bluetoothprofiles.pdf
[12] Nathan J. Muller - Bluetooth Demystified, publisher McGraw-Hill, 396 p , 2001
[13] Hunag, Albert S. – Rudolph, Larry : Bluetooth Essentials for programmers, Cambridge University Press, 2007, 193 s. ISBN 978-0-521-70375-8.
[14] MULLER, Nathan J. : Bluetooth Demystified, McGraw-Hill Companies, 381 s. ISBN 0-07-136323-8.

Vyhľadávanie a použitie proto-fuzzy konceptov

Ondrej Krídlo and Stanislav Krajči

Prírodovedecká fakulta, Univerzita P. J. Šafárika v Košiciach, Slovenská republika,
ondrej.kridlo@upjs.sk, stanislav.krajci@upjs.sk

Abstrakt *Cieľom tohto príspevku je definovať tzv. proto-fuzzy koncepty, ako „univerzálny“ základ fuzzy formálnej konceptovej analýzy pre vyhľadávanie jednostranných fuzzy konceptov. Fuzzy formálny kontext je trojica tvorená množinou objektov, množinou atribútov a fuzzy binárnou reláciou, ktorá určuje stupeň príslušnosti atribútu objektu, proto–fuzzy koncept je trojica tvorená množinou objektov, množinou atribútov a hodnotou, ktorá je maximálny spoločný stupeň príslušnosti všetkých atribútov z danej množiny všetkým objektom danej množiny objektov. Proto-fuzzy koncepty budeme vyhľadávať pomocou rezov a priemetov do objektovo–hodnotovej resp. atribútovo–hodnotovej roviny, čím budeme problematiku prenášať do reči klasickej formálnej konceptovej analýzy.*

1 Úvod a motivácia

Predstavme si skupinu desiatich spolužiakov strednej školy a ich študijné výsledky z desiatich rôznych predmetov, napr. ako je uvedené nižšie v tabuľke. Z dôvodu šetrenia priestoru budeme predmety označovať skratkami (Ma – Matematika, Sj – Slovenský jazyk, Fy – Fyzika, Ge – Geografia, Bi – Biológia, Nj – Nemecký jazyk, Aj – Anglický jazyk, Ch – Chémia, Nos – Náuka o spoločnosti, De – Dejepis). Namiesto mien študentov budeme používať začiatočné písmená, ako je uvedené v tabuľke.

		Ma	Sj	Fy	Ge	Bi	Nj	Aj	Ch	Nos	De
F	Ferko	1	1	1	3	2	1	2	2	1	2
J	Jožko	3	1	2	1	1	1	1	3	1	1
A	Alenka	3	2	3	1	1	1	1	3	2	2
N	Noro	4	2	4	3	2	2	1	2	3	2
M	Majka	1	1	1	1	1	1	1	1	1	1
E	Evka	1	1	1	1	1	1	1	1	1	1
L	Lucia	1	1	1	2	2	2	2	1	2	2
D	David	2	3	4	3	4	1	1	2	2	2
P	Peter	2	1	2	1	1	2	2	3	1	2
T	Tomáš	1	3	2	2	2	2	2	3	1	2

Tab. 1. Príklad formálneho fuzzy kontextu.

Daná tabuľka je konkrétnym príkladom fuzzy kontextu. Študenti v danom príklade vystupujú ako objekty, predmety ako atribúty a známky ako stupne príslušnosti dvojíc objekt–atribút, v našom prípade študent–predmet, fuzzy binárnej relácii, čo je v našom prípade tabuľka známok. Našou úlohou je vyhľadávať zhluky, resp. skupiny študentov podobných z hľadiska úspešnosti. Na rozdiel od klasickej formálnej konceptovej analýzy podobnosť objektov v jednostrannom fuzzy koncepte nebudú určovať klasické, ale fuzzy podmnožiny atribútov. Pod jednostranným fuzzy konceptom si budeme predstavovať dvojicu tvorenú klasickou podmnožinou množiny objektov a fuzzy podmnožinou množiny atribútov.([1]).

Východiskom tohto článku je definovať tzv. *proto–fuzzy koncepty*, ktoré predstavujú trojicu tvorenú klasickou množinou objektov, klasickou množinou atribútov a stupeň príslušnosti, ktorý neprekračuje prislúchajúci stupeň danej fuzzy binárnej relácie. Každý prvok takejto trojice je maximálny vzhľadom na ostatné dva. Proto-fuzzy koncepty je si možné predstaviť ako „základnú stavebnú jednotku“ jednostranných fuzzy konceptov. Ak sa na hodnoty v tabuľke dívame ako na najvyššie hodnoty ktoré príslušná dvojica objekt–atribút nadobúda, teda že jej „vyhovujú“ aj všetky od nej menšie, tak proto-fuzzy koncept vyzerá ako „kváder“ vyhovujúcich trojíc objekt–atribút–stupeň príslušnosti. V našom príklade to znamená, že študent zvláda nejaký predmet na príslušnú známku z tabuľky, tak ho zvláda aj na každú nižšiu.

Príklady niektorých proto-fuzzy konceptov sú uvedené v tretej kapitole.

2 Základné definície

Definícia 1. *Formálnym kontextom nazývame trojicu* $\langle \mathcal{O}, \mathcal{A}, R \rangle$*,* $\mathcal{O}$ *je konečná množina tzv.* objektov*,* $\mathcal{A}$ *je konečná množina tzv.* atribútov *a* $R \subseteq \mathcal{O} \times \mathcal{A}$ *je binárna relácia, určujúca ktorý objekt spĺňa aký atribút, resp. ktorý atribút patrí akému objektu.*

Definícia 2. *Fuzzy formálnym kontextom nazývame štvoricu* $\langle \mathcal{O}, \mathcal{A}, L, \mathcal{R} \rangle$*,* $\mathcal{O}$ *je konečná množina* objektov*,* $\mathcal{A}$ *je konečná množina* atribútov*,* L *je úplný reziduovaný zväz a* $\mathcal{R}$ *je fuzzy podmnožina* $\mathcal{O} \times \mathcal{A}$*, tj. zobrazenie z* $L^{\mathcal{O} \times \mathcal{A}}$*, resp. fuzzy binárna relácia.*

Definícia 3. *Pre každé* $l \in L$ *definujme zobrazenia* $\uparrow_l$: $\mathcal{P}(\mathcal{O}) \to \mathcal{P}(\mathcal{A})$ *a* $\downarrow_l$: $\mathcal{P}(\mathcal{A}) \to \mathcal{P}(\mathcal{O})$ *nasledovne: Pre každú* $O \subseteq \mathcal{O}$

$$\uparrow_l (O) = \{a \in \mathcal{A} : (\forall o \in O)\mathcal{R}(o, a) \geq l\}$$

a pre každú $A \subseteq \mathcal{A}$

$$\downarrow_l (A) = \{o \in \mathcal{O} : (\forall a \in A)\mathcal{R}(o, a) \geq l\}.$$

Definícia 4. *Nech $\langle \mathcal{O}, \mathcal{A}, L, \mathcal{R} \rangle$ je fuzzy kontext. Hovoríme, že dvojica $\langle O, A \rangle$ je l-**koncept** $akk \uparrow_l (O) = A$, a zároveň $\downarrow_l (A) = O$, resp. je klasickým formálnym konceptom v klasickom kontexte $\langle \mathcal{O}, \mathcal{A}, R_l \rangle$, pričom*

$$R_l = \{(o, a) \in \mathcal{O} \times \mathcal{A} : \mathcal{R}(o, a) \geq l\}.$$

Kontext $\langle \mathcal{O}, \mathcal{A}, R_l \rangle$ nazvime l-rez. *Množinu všetkých konceptov v l-reze označíme $\mathcal{K}_l$.*

V našom príklade predstavuje l-rez pohľad na úspešnosť žiakov pre konkrétny stupeň l, resp. dáva odpoveď na otázku: Zvláda daný študent ($o \in \mathcal{O}$) učivo predmetu ($a \in \mathcal{A}$) na známku aspoň $l \in L$? Koncept $\langle O, A \rangle$ z množiny $\mathcal{K}_l$ potom predstavuje zhluk alebo skupinu študentov O, z ktorých každý dosahuje zo všetkých predmetov z množiny A aspoň známku l.

1	Ma	Sj	Fy	Ge	Bi	Nj	Aj	Ch	Nos	De
F	•	•	•		•	•	•	•	•	•
J		•	•	•	•	•	•		•	•
A		•		•	•	•	•		•	•
N		•			•	•	•	•		•
E	•	•	•	•	•	•	•	•	•	•
M	•	•	•	•	•	•	•	•	•	•
L	•	•	•	•	•	•	•	•	•	•
D	•					•	•	•	•	•
P	•	•	•	•	•	•	•		•	•
T	•		•	•	•	•	•		•	•

Tab. 2. 2-rez.

Preskúmaním všetkých rezov by sme prišli k pozorovaniu, že mnohé koncepty v jednotlivých rezoch sa opakujú a informácia, že napr. Majka a Evka zvládajú všetky predmety na aspoň dvojku nie je „úplná“, pretože Majka a Evka zvládajú všetky predmety na jednotku. Teda predchádzajúca informácia nebola „uzavretá“ v niektorom rozmere.

Teraz si ukážeme zaujímavú vlastnosť *konvexnosti* a *uzavretosti vzhľadom na suprémum* zhodných klasických konceptov v jednotlivých rezoch. Spomínané vlastnosti rezov rozširujú prínos článku [2], v ktorom sa autori tiež zaoberali rezmi.

Lema 1. *Nech $l_1, l_2 \in L$ také, že $l_1 \leq l_2$, tak pre ľubovoľné $O \subseteq \mathcal{O}$ a $A \subseteq \mathcal{A}$ platí, že $\uparrow_{l_1} (O) \supseteq \uparrow_{l_2} (O)$ a $\downarrow_{l_1} (A) \supseteq \downarrow_{l_2} (A)$.*

Dôkaz. Dôkaz urobíme pre $\uparrow$. Dôkaz pre $\downarrow$ je analogický. Ak $l_1 \leq l_2$, tak $\{a \in \mathcal{A} : (\forall o \in O) : \mathcal{R}(o, a) \geq l_1\}$ je nadmnožinou $\{a \in \mathcal{A} : (\forall o \in O) : \mathcal{R}(o, a) \geq l_2\}$, a teda je $\uparrow_{l_1} (O) \supseteq \uparrow_{l_2} (O)$. □

Lema 2. *Nech $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ a $l_1, l_2 \in L$. Potom platí, že*

$$\uparrow_{l_1} (O) \cap \uparrow_{l_2} (O) = \uparrow_{l_1 \vee l_2} (O),$$

a zároveň

$$\downarrow_{l_1} (A) \cap \downarrow_{l_2} (A) = \downarrow_{l_1 \vee l_2} (A).$$

Dôkaz. Ak $a \in \uparrow_{l_1} (O) \cap \uparrow_{l_2} (O)$, tak pre každé $o \in O$ platí, že $\mathcal{R}(o, a) \geq l_1$, a zároveň $\mathcal{R}(o, a) \geq l_2$. Z toho ale plynie, že pre každé $o \in O$ je $\mathcal{R}(o, a) \geq l_1 \vee l_2$, teda $a \in \uparrow_{l_1 \vee l_2} (O)$. Teda

$$\uparrow_{l_1} (O) \cap \uparrow_{l_2} (O) \subseteq \uparrow_{l_1 \vee l_2} (O).$$

Z predchádzajúcej lemy plynie, že

$$\uparrow_{l_1 \vee l_2} (O) \subseteq \uparrow_{l_1} (O),$$

a zároveň

$$\uparrow_{l_1 \vee l_2} (O) \subseteq \uparrow_{l_2} (O).$$

Z toho ale máme že

$$\uparrow_{l_1 \vee l_2} (O) \subseteq \uparrow_{l_1} (O) \cup \uparrow_{l_2} (O).$$

Pre $\downarrow$ sa dôkaz urobí analogicky. □

Veta 1. *Nech $l_1, l_2 \in L$ a nech $\langle O, A \rangle \in \mathcal{K}_{l_1} \cap \mathcal{K}_{l_2}$. Potom pre každé $l \in L$ také, že $l_1 \leq l \leq l_2$ platí: $\langle O, A \rangle \in \mathcal{K}_l$.*

Dôkaz. Z lemy 1 a $\langle O, A \rangle \in \mathcal{K}_{l_1} \cap \mathcal{K}_{l_2}$ platí, že
$A = \uparrow_{l_1} (O) \supseteq \uparrow_l (O) \supseteq \uparrow_{l_2} (O) = A$
$O = \downarrow_{l_1} (A) \supseteq \downarrow_l (A) \supseteq \downarrow_{l_2} (A) = O$
Teda $\uparrow_l (O) = A$ a $\downarrow_l (A) = O$. Teda $\langle O, A \rangle \in \mathcal{K}_l$.
□

Veta 2. *Nech $l_1, l_2 \in L$ a nech $\langle O, A \rangle \in \mathcal{K}_{l_1} \cap \mathcal{K}_{l_2}$. Potom $\langle O, A \rangle \in \mathcal{K}_{l_1 \vee l_2}$.*

Dôkaz. Z lemy 2 plynie, že

$$\uparrow_{l_1 \vee l_2} (O) = \uparrow_{l_1} (O) \cap \uparrow_{l_2} (O) = A \cap A = A,$$

a zároveň

$$\downarrow_{l_1 \vee l_2} (A) = \downarrow_{l_1} (A) \cap \downarrow_{l_2} (A) = O \cap O = O.$$

Teda $\langle O, A \rangle \in \mathcal{K}_{l_1 \vee l_2}$. □

3 Proto-fuzzy koncepty a ich využitie

Definícia 5. *Proto-fuzzy konceptom nazveme trojicu $\langle O, A, l \rangle \in \mathcal{P}(\mathcal{O}) \times \mathcal{P}(\mathcal{A}) \times L$, pričom platí, že $\langle O, A \rangle \in \bigcup_{k \in L} \mathcal{K}_k$, a zároveň $l = \sup\{k \in L : \langle O, A \rangle \in \mathcal{K}_k\}$. Množinu všetkých proto-fuzzy konceptov budeme označovať $\mathcal{K}^P$.*

V našom príklade bude proto-fuzzy koncept $\langle O, A, l \rangle$ predstavovať skupinu študentov O, ktorých najlepšia spoločná známka za všetky predmety z množiny A je l, pričom každý prvok trojice je vzhľadom na ostatné maximálny. V nasledujúcich tabuľkách sú uvedené niektoré proto-fuzzy koncepty nášho príkladu.

{F, J, A, N, M, E, L, D, P, T}
{Sj, Ge, Nj, Aj, Ch, Nos, De}
3

{J, P, M, E, L}	{F, M, E, L}
{Sj, Fy, Ge, Bi, Nos, Nj, Aj}	{Ma, Fy}
2	1

Množinu všetkých proto-fuzzy konceptov použijeme na tvorbu jednostranných fuzzy konceptov a to pomocou zobrazení, ktoré definujeme nižšie.

K definíciám budeme potrebovať jeden pomocný pojem, takzvaného zúženia $\mathcal{K}^P$ pre nejakú množinu objektov. Je to množina všetkých dvojíc tvorených množinou atribútov a stupňom príslušnosti, takých že existuje nadmnožina danej množiny objektov, ktorá s dvojicou dáva proto-fuzzy koncept. Ak $O \subseteq \mathcal{O}$, tak zúženie prislúchajúce množine O označíme ako $\mathcal{K}^P_O$.

Definícia 6. *Nech $O \subseteq \mathcal{O}$ je ľubovoľná množina objektov. Množinu $\mathcal{K}^P_O = \{\langle A, l\rangle \in \mathcal{P}(\mathcal{A}) \times L : (\exists B \supseteq O)\ \langle B, A, l\rangle \in \mathcal{K}^P\}$ budeme nazývať* zúžením *množiny proto-fuzzy konceptov prislúchajúcim množine objektov O.*

Pomocou zúžení môžeme definovať hľadané zobrazenia.

Definícia 7. *Definujme zobrazenia*

$$\Uparrow\colon 2^{\mathcal{O}} \to L^{\mathcal{A}},$$
$$\Downarrow\colon L^{\mathcal{A}} \to 2^{\mathcal{O}}$$

nasledovne: Nech $\widetilde{A}$ je L-fuzzy množina atribútov a $O \subseteq \mathcal{O}$ je množina objektov.

$$\Downarrow(\widetilde{A}) = \bigcup\{B \subseteq \mathcal{O} : (\forall a \in \mathcal{A})(\exists\langle A, l\rangle \in \mathcal{K}^P_B)$$
$$a \in A\ \&\ l \geq \widetilde{A}(a)\}$$
$$\Uparrow(O)(a) = \sup\{l \in L : (\exists\langle A, l\rangle \in \mathcal{K}^P_O) a \in A\}$$

Lema 3. *Nech $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ a $l \in L$ také, že pre každý objekt o množiny O a pre každý atribút a množiny A je $\mathcal{R}(o, a) \geq l$. Potom existujú množiny $\overline{O} \supseteq O$, $\overline{A} \supseteq A$ a hodnota $k \in L$, že $k \geq l$ a $\langle\overline{O}, \overline{A}, k\rangle \in \mathcal{K}^P$.*

Dôkaz. Je dané, že $(\forall o \in O)(\forall a \in A)\mathcal{R}(o, a) \geq l$. Vezmime

$$\overline{A} = \uparrow_l(O) = \{a \in \mathcal{A} : (\forall o \in O)\mathcal{R}(o, a) \geq l\} \supseteq A.$$

Potom

$$\overline{O} = \downarrow_l(\overline{A}) = \downarrow_l(\uparrow_l(O))$$

a z faktu, že pre každé $l \in L$ dvojica zobrazení $(\uparrow_l, \downarrow_l)$ tvorí Galoisovú konnexiu plynie, že $\downarrow_l(\uparrow_l(O)) \supseteq O$ a $\langle\overline{O}, \overline{A}\rangle \in \mathcal{K}_l$. Za k vezmime hodnotu

$$k = \sup\{m \in L : \langle\overline{O}, \overline{A}\rangle \in \mathcal{K}_m\}$$

z vety 2 plynie $\langle\overline{O}, \overline{A}\rangle \in \mathcal{K}_k$ a teda

$$\langle\overline{O}, \overline{A}, k\rangle \in \mathcal{K}^P.$$

□

Lema 4. *Nech $l \in L$, $O_1, O_2 \subseteq \mathcal{O}$ a $\langle A_1, l_1\rangle \in \mathcal{K}^P_{O_1}$, $\langle A_2, l_2\rangle \in \mathcal{K}^P_{O_2}$ také, že $A_1 \cap A_2 \neq \emptyset$ a $l_1 \wedge l_2 \geq l$. Potom existuje $\langle A, k\rangle \in \mathcal{K}^P_{O_1 \cup O_2}$ také, že $A \supseteq A_1 \cap A_2$ a $k \geq l$.*

Dôkaz. $\langle A_1, l_1\rangle \in \mathcal{K}^P_{O_1}$ z toho plynie

$$(\forall o \in O_1)(\forall a \in A_1)\mathcal{R}(o, a) \geq l_1.$$

$\langle A_2, l_2\rangle \in \mathcal{K}^P_{O_2}$ z toho plynie

$$(\forall o \in O_2)(\forall a \in A_2)\mathcal{R}(o, a) \geq l_2.$$

Teda

$$(\forall o \in O_1 \cup O_2)(\forall a \in A_1 \cap A_2)\mathcal{R}(o, a) \geq l_1 \wedge l_2 \geq l.$$

A z lemy 3 plynie, že

$$(\exists O \supseteq O_1 \cup O_2)(\exists A \supseteq A_1 \cap A_2)(\exists k \in L : k \geq l)$$
$$\langle O, A, k\rangle \in \mathcal{K}^P$$

a teda

$$\langle A, k\rangle \in \mathcal{K}^P_{O_1 \cup O_2}.$$

□

Lema 5. *Nech $O \subseteq \mathcal{O}$, $\langle A_1, l_1\rangle$, $\langle A_2, l_2\rangle \in \mathcal{K}^P_O$ také, že $A_1 \cap A_2 \neq \emptyset$. Potom existuje $A \subseteq \mathcal{A}$ a $l \geq l_1 \vee l_2$ také, že $\langle A, l\rangle \in \mathcal{K}^P_O$.*

Dôkaz. Pre všetky $o \in O$ a pre všetky $a \in A_1 \cap A_2$ platí

$$\mathcal{R}(o, a) \geq l_1 \text{ a } \mathcal{R}(o, a) \geq l_2.$$

A teda

$$\mathcal{R}(o, a) \geq l_1 \vee l_2.$$

Z uvedeného vyššie a lemy 3 vyplýva

$$(\exists B \supseteq O)(\exists A \supseteq A_1 \cap A_2)(\exists l \in L : l \geq l_1 \vee l_2)$$
$$\langle B, A, l\rangle \in \mathcal{K}^P.$$

A teda

$$\langle A, l\rangle \in \mathcal{K}^P_O.$$

□

Lema 6. *Nech $O_1, O_2 \subseteq \mathcal{O}$ také, že $O_1 \subseteq O_2$. Potom $\mathcal{K}^P_{O_1} \supseteq \mathcal{K}^P_{O_2}$.*

Dôkaz. Z $O_1 \subseteq O_2$ vyplýva, že množina

$$\{\langle A_1, l_1\rangle \in \mathcal{P}(A) \times L : (\exists B_1 \supseteq O_1)\langle B_1, A_1, l_1\rangle \in \mathcal{K}^P\}$$

je nadmnožinou množiny

$$\{\langle A_2, l_2\rangle \in \mathcal{P}(A) \times L : (\exists B_2 \supseteq O_2)\langle B_2, A_2, l_2\rangle \in \mathcal{K}^P\}.$$

A teda

$$\mathcal{K}^P_{O_1} \supseteq \mathcal{K}^P_{O_2}.$$

□

Veta 3. *Dvojica zobrazení* $(\Uparrow, \Downarrow)$ *tvorí Galoisovú konexiu medzi zväzom podmnožín* $\mathcal{P}(\mathcal{O})$ *a zväzom fuzzy podmnožín* $\widetilde{\mathcal{P}}(\mathcal{A})$.

Dôkaz. Pre ľubovoľnú množinu objektov O a fuzzy-množinu atribútov $\widetilde{A}$ dokážeme, že O je podmnožinou $\Downarrow(\widetilde{A})$ práve vtedy, ak $\widetilde{A}$ je fuzzy-podmnožinou $\Uparrow(O)$.
$\boxed{\Rightarrow}$ $O \subseteq \Downarrow(\widetilde{A}) = \bigcup\{B \subseteq \mathcal{O} : (\forall b \in \mathcal{A})(\exists\langle A, l\rangle \in \mathcal{K}_B^P) b \in A$ & $l \geq \widetilde{A}(b)\}$. Nech $a \in \mathcal{A}$ ľubovoľný atribút. Z lemma 4 plynie, že existujú $A_a \subseteq \mathcal{A}$ a $l_a \in L$, že platí $a \in A_a$, $l_a \geq \widetilde{A}(a)$ a $\langle A_a, l_a\rangle \in \mathcal{K}^P_{\Downarrow(\widetilde{A})}$. Z predpokladu $O \subseteq \Downarrow(\widetilde{A})$ plynie $\mathcal{K}_O^P \supseteq \mathcal{K}^P_{\Downarrow(\widetilde{A})}$. Teda $\langle A_a, l_a\rangle \in \mathcal{K}_O^P$. Potom $\widetilde{A}(a) \leq l_a \leq \sup\{l \in L : (\exists\langle A, l\rangle \in \mathcal{K}_O^P) a \in A\} = \Uparrow(O)(a)$. Pretože a je ľubovoľný atribút z $\mathcal{A}$ a z nerovnosti vyššie plynie $\widetilde{A}$ je fuzzy-podmnožina $\Uparrow(O)$.
$\boxed{\Leftarrow}$ Nech $a \in \mathcal{A}$ je ľubovoľný atribút. Označíme

$$l_a = \Uparrow(O)(a) = \sup\{l \in L : (\exists\langle A, l\rangle \in \mathcal{K}_O^P) a \in A\}.$$

Z predpokladu, že $\widetilde{A}$ je fuzzy-podmnožinou $\Uparrow(O)$ plynie, že pre každý atribút $a \in \mathcal{A}$ je $\widetilde{A}(a) \leq l_a$. Z lemy 5 plynie, že existuje $A_a \subseteq \mathcal{A}$ taká, že $\langle A_a, l_a\rangle \in \mathcal{K}_O^P$, a z toho
$O \in \{B \subseteq \mathcal{O} : (\forall b \in \mathcal{A})(\exists\langle A, l\rangle \in \mathcal{K}_B^P) a \in A \;\&\; l \geq \widetilde{A}(b)\}$, teda
$O \subseteq \bigcup\{B \subseteq \mathcal{O} : (\forall b \in \mathcal{A})(\exists\langle A, l\rangle \in \mathcal{K}_B^P) a \in A \;\&\; l \geq \widetilde{A}(b)\} = \Downarrow(\widetilde{A})$.
Teda množina O je podmnožinou $\Downarrow(\widetilde{A})$. □

4 Vyhľadávanie proto-fuzzy konceptov

Proto-fuzzy koncepty budeme vyhľadávať pomocou rezov a priemetov do objektovo–hodnotovej, resp. atribútovo–hodnotovej roviny. Ak si hodnoty v tabuľke predstavíme ako „stĺpčeky“ umiestnené v trojrozmernom kvádri na ktorý sa dívame zhora, tak dané priemety sú pohľady z prislúchajúcich strán.

Rezy sme si už definovali už skôr.

Definícia 8. *Nech*

$$R_{\mathcal{A}} = \{(o, l) \in \mathcal{O} \times L : (\forall a \in \mathcal{A})\mathcal{R}(o, a) \geq l\}$$

a

$$R_{\mathcal{O}} = \{(a, l) \in \mathcal{A} \times L : (\forall o \in \mathcal{O})\mathcal{R}(o, a) \geq l\}.$$

Potom objektovo–hodnotový pohľad na fuzzy kontext je klasický formálny kontext

$$\langle \mathcal{O}, L, R_{\mathcal{A}}\rangle.$$

Atribútovo–hodnotový pohľad je

$$\langle \mathcal{A}, L, R_{\mathcal{O}}\rangle.$$

	1	2	3	4	5
Ferko			•	•	•
Jožko			•	•	•
Alenka			•	•	•
Noro				•	•
Majka	•	•	•	•	•
Evka	•	•	•	•	•
Lucia		•	•	•	•
David				•	•
Peter			•	•	•
Tomáš			•	•	•

Tab. 3. Objektovo - hodnotový pohľad.

	1	2	3	4	5
Matematika				•	•
Slovenský jazyk			•	•	•
Fyzika				•	•
Geografia			•	•	•
Biológia				•	•
Nemecký jazyk		•	•	•	•
Anglický jazyk		•	•	•	•
Chémia			•	•	•
Náuka o spoločnosti			•	•	•
Dejepis		•	•	•	•

Tab. 4. Atribútovo - hodnotový pohľad.

Pohľady prislúchajúce nášmu príkladu sú v nasledujúcich tabuľkách.

Pre vytváranie konceptov v pohľadoch si zadefinujeme zobrazenia spĺňajúce Galoisovu konexiu. Klasickej podmnožine objektov, resp. atribútov budeme priraďovať stupeň príslušnosti, najlepší spoločný pre všetky dvojice kartézskeho súčinu podmnožiny objektov, resp. atribútov a celej množiny atribútov, resp. objektov.

Definícia 9. *Definujme zobrazenia*
$\uparrow_{\mathcal{A}}: 2^{\mathcal{O}} \to L$ *a* $\downarrow_{\mathcal{A}}: L \to 2^{\mathcal{O}}$,
resp.
$\uparrow_{\mathcal{O}}: 2^{\mathcal{A}} \to L$ *a* $\downarrow_{\mathcal{O}}: L \to 2^{\mathcal{A}}$
nasledovne:
Nech $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ *a* $l \in L$, *tak*
$\uparrow_{\mathcal{A}}(O) = \inf\{\sup\{l \in L : (o, l) \in R_{\mathcal{A}}\} : o \in O\}$
$\downarrow_{\mathcal{A}}(l) = \{o \in \mathcal{O} : (o, l) \in R_{\mathcal{A}}\}$
$\uparrow_{\mathcal{O}}(A) = \inf\{\sup\{l \in L : (a, l) \in R_{\mathcal{O}}\} : a \in A\}$
$\downarrow_{\mathcal{O}}(l) = \{a \in \mathcal{A} : (a, l) \in R_{\mathcal{O}}\}$

Veta 4. *Dvojice zobrazení* $(\uparrow_{\mathcal{A}}, \downarrow_{\mathcal{A}})$ *a* $(\uparrow_{\mathcal{O}}, \downarrow_{\mathcal{O}})$ *tvoria Galoisove konexie medzi zväzmy podmnožín* $\mathcal{P}(\mathcal{O})$, *resp.* $\mathcal{P}(\mathcal{A})$ *a zväzom* L.

Dôkaz. Tvrdenie dokážeme iba pre prvú dvojicu. Ukážeme platnosť štyroch podmienok nutných pre platnosť Galoisovej konexie.

1. Nech $O_1 \subseteq O_2 \subseteq \mathcal{O}$. Z toho vyplýva, že

$$\{\sup\{l \in L : (o,l) \in R_\mathcal{A}\} : o \in O_1\} \subseteq$$

$$\{\sup\{l \in L : (o,l) \in R_\mathcal{A}\} : o \in O_2\}$$

a z vlastností infima je

$$\inf\{\sup\{l \in L : (o,l) \in R_\mathcal{A}\} : o \in O_1\} \geq$$

$$\inf\{\sup\{l \in L : (o,l) \in R_\mathcal{A}\} : o \in O_2\},$$

a teda

$$\uparrow_\mathcal{A}(O_1) \geq \uparrow_\mathcal{A}(O_2).$$

2. Nech $l_1, l_2 \in L$, $l_1 \leq l_2$. Potom

$$\{o \in \mathcal{O} : (o,l_1) \in R_\mathcal{A}\} \supseteq \{o \in \mathcal{O} : (o,l_2) \in R_\mathcal{A}\}$$

a teda

$$\downarrow_\mathcal{A}(l_1) \supseteq \downarrow_\mathcal{A}(l_1)$$

3. Nech $O \subseteq \mathcal{O}$. Označme

$$s_{o'} = \sup\{l \in L : (o',l) \in R_\mathcal{A}\},$$

pre ľubovoľné $o' \in O$. Z definície $\uparrow_\mathcal{A}$ je

$$\uparrow_\mathcal{A}(O) \leq s_o$$

a z toho máme

$$\downarrow_\mathcal{A}(\uparrow_\mathcal{A}(O)) = \{o \in \mathcal{O} : (o, \uparrow_\mathcal{A}(O)) \in R_\mathcal{A}\} \supseteq$$

$$\supseteq \{o \in \mathcal{O} : (o, s_{o'}) \in R_\mathcal{A}\}.$$

Napokon

$$\downarrow_\mathcal{A}(\uparrow_\mathcal{A}(O)) \supseteq \bigcup_{o' \in O} \{o \in \mathcal{O} : (o, s_{o'}) \in R_\mathcal{A}\} \supseteq O$$

4. Nech $l \in L$ ľubovoľné. Potom pre každé

$$o' \in \downarrow_\mathcal{A}(l) = \{o \in \mathcal{O} : (o,l) \in R_\mathcal{A}\}$$

platí, že

$$\sup\{l \in L : (o',l) \in R_\mathcal{A}\} \geq l.$$

A teda

$$\uparrow_\mathcal{A}(\downarrow_\mathcal{A}(l)) =$$

$$= \inf\{\sup\{l \in L : (o,l) \in R_\mathcal{A}\} : o \in \downarrow_\mathcal{A}(l)\} \geq l.$$

□

Veta 5. *Nech $l \in L$, $A_1, A_2 \subseteq \mathcal{A}$, $O_1, O_2 \subseteq \mathcal{O}$ také, že $\langle \mathcal{O}, A_1, l\rangle, \langle O_1, \mathcal{A}, l\rangle \in \mathcal{K}^P$ a $\langle O_2, A_2\rangle \in \mathcal{K}_l$ pre kontext $\langle \mathcal{O} \setminus O_1, \mathcal{A} \setminus A_2, R_l\rangle$. Potom*

$$\langle O_1 \cup O_2, A_1 \cup A_2, l\rangle, \in \mathcal{K}^P.$$

Dôkaz. Ukážeme, že $A_1 \cup A_2 = \downarrow_l (O_1 \cup O_2)$, a zároveň $O_1 \cup O_2 = \uparrow_l (A_1 \cup A_2)$.

Ak $a \in A_1$, tak pre každé $o \in \mathcal{O}$ je $(o,a) \in R_l$.

Ak $a \in A_2$, tak pre každé $o \in O_1 \cup O_2$ je $(o,a) \in R_l$. Teda ak $a \in A_1 \cup A_2$, tak pre každé

$$o \in (\mathcal{O} \cap (O_1 \cup O_2)) = O_1 \cup O_2$$

je $(o,a) \in R_l$.

Teda

$$A_1 \cup A_2 \subseteq \uparrow_l (O_1 \cup O_2).$$

Opačnú inklúziu ukážeme sporom. Nech existuje $a \in \uparrow_l (O_1 \cup O_2)$, a zároveň $a \notin A_1 \cup A_2$.

Z $a \in \uparrow_l (O_1 \cup O_2)$ plynie, že pre každé $o \in O_1 \cup O_2 \supseteq O_2$ je $(o,a) \in R_l$. Z $a \notin A_1 \cup A_1$ plynie, že $a \in (\mathcal{A} \setminus (A_1 \cup A_1)) = ((\mathcal{A} \setminus A_1) \setminus A_2)$. To je ale spor s predpokladom $\langle O_2, A_2\rangle \in \mathcal{K}_l$ pre kontext $\langle \mathcal{O} \setminus O_1, \mathcal{A} \setminus A_1, R_l\rangle$.

Druhá rovnosť sa ukáže analogicky. □

Ukážeme si teraz ako to bude vyzerať v praxi. Pomocou zobrazení $\uparrow_\mathcal{A}$, $\downarrow_\mathcal{A}$, $\uparrow_\mathcal{O}$ a $\downarrow_\mathcal{O}$ získame koncepty v pohľadoch, pričom si treba uvedomiť, že ak napr. $\langle O, l\rangle \in \mathcal{K}_\mathcal{A}$, tak $\langle O, \mathcal{A}, l\rangle \in \mathcal{K}^P$, pretože $\mathcal{A}$ je uzavretá.

Z tabuliek pohľadov je ľahko vidieť ich koncepty.

Pre stupeň (známku) 4 je to $\langle \mathcal{O}, \mathcal{A}, 4\rangle$, ak označíme všetkých študentov ako $\mathcal{O}$ a množinu všetkých predmetov ako $\mathcal{A}$.

Pre známku 3 sú to

$$\langle \mathcal{O} \setminus \{\mathrm{N}, \mathrm{D}\}, \mathcal{A}, 3\rangle$$

a

$$\langle \mathcal{O}, \mathcal{A} \setminus \{\mathrm{Ma}, \mathrm{Fy}, \mathrm{Bi}\}, 3\rangle.$$

Pre známku 2 sú to

$$\langle \{\mathrm{M}, \mathrm{E}, \mathrm{L}\}, \mathcal{A}, 2\rangle$$

a

$$\langle \mathcal{O}, \{\mathrm{Nj}, \mathrm{Aj}\}, 2\rangle.$$

Pre známku 1 sú to

$$\langle \{\mathrm{M}, \mathrm{E}\}, \mathcal{A}, 1\rangle$$

a

$$\langle \mathcal{O}, \emptyset, 1\rangle.$$

3	Ma	Fy	Bi
N			•
D	•		

Tab. 5. Pomocný subkontext 3-rezu.

Podľa vety vytvorýme pomocný subkontext 3-rezu. V tomto kontexte $\langle\{\mathrm{N},\mathrm{D}\},\{\mathrm{Ma},\mathrm{Fy},\mathrm{Bi}\},3\rangle$ sú iba dva koncepty $\langle\{\mathrm{N}\},\{\mathrm{Bi}\}\rangle$ a $\langle\{\mathrm{D}\},\{\mathrm{Ma}\}\rangle$. Podľa vety potom okrem

$$\langle\mathcal{O}\setminus\{\mathrm{N},\mathrm{D}\},\mathcal{A},3\rangle,\langle\mathcal{O},\mathcal{A}\setminus\{\mathrm{Ma},\mathrm{Fy},\mathrm{Bi}\},3\rangle\in\mathcal{K}^P$$

aj

$$\langle\mathcal{O}\setminus\{\mathrm{N}\},\mathcal{A}\setminus\{\mathrm{Fy},\mathrm{Bi}\},3\rangle\in\mathcal{K}^P$$

a

$$\langle\mathcal{O}\setminus\{\mathrm{D}\},\mathcal{A}\setminus\{\mathrm{Ma},\mathrm{Fy}\},3\rangle\in\mathcal{K}^P.$$

Pokračujeme pre stupeň 2. Vytvoríme pomocný subkontext 2-rezu. A z neho začneme získavať kontexty a každý aplikujeme na kontexty pohľadov s hodnotou 2.

2	Ma	Sj	Fy	Ge	Bi	Ch	Nos
F	•		•	•	•	•	•
J		•	•	•	•		•
A		•		•	•		•
N		•			•	•	
D	•					•	•
P	•	•	•	•	•		•
T	•		•	•	•		•

Tab. 6. Pomocný subkontext 2-rezu.

Pre veľký počet konceptov subkontextu 2-rezu a 1-rezu nebudeme všetky ich koncepty uvádzať.

$\langle\{J,P\},\{Sj,Fy,Ge,Bi,Nos\}\rangle$ je konceptom pomocného subkontextu 2-rezu. A keďže

$$\langle\{M,E,L\},\mathcal{A},2\rangle,\langle\mathcal{O},\{Nj,Aj\},2\rangle\in\mathcal{K}^P,$$

tak aj

$$\langle\{J,P,M,E,L\},\{Sj,Fy,Ge,Bi,Nos,Nj,Aj\},2\rangle\in\mathcal{K}^P.$$

1	Ma	Sj	Fy	Ge	Bi	Nj	Aj	Ch	Nos	De
F	•	•	•			•			•	
J		•		•	•	•	•		•	•
A				•	•	•	•			
N							•			
L	•	•	•					•		
D						•	•			
P		•		•	•				•	
T	•								•	

Tab. 7. Pomocný subkontext 1-rezu.

Pre koncepty pomocného subkontextu 1-rezu postupujeme podobne.

5 Záver

V budúcnosti sa chceme venovať optimalizácii, algoritmizácii a aplikácii načrtnutých postupov v praxi.

Na záver chceme poďakovať kolegom RNDr. Petrovi Eliašovi PhD. a RNDr. Jozefovi Pócsovi za cenné rady a konštruktívne pripomienky.

Príspevok vznikol s podporou grantu 1/3129/06 Slovenskej grantovej agentúry VEGA.

Referencie

1. Krajči, S.: Cluster Based Efficient Generation Of Fuzzy Concepts Neural Network World 5/03 521–530
2. Snášel V., Ďuráková D., Krajči S., Vojtáš P.: Merging Concept Lattices of α-cuts Of Fuzzy Contexts Contributions To General Algebra 14 Proceedings of the Olomouc Conference 2002 (AAA 64) and the Postdam Conference 2003 (AAA 65) Verlag Johanes Heyn, Klagenfurt 2004
3. Bělohlávek R.:Concept Lattices And Order In Fuzzy Logic, Anals of Pure and Applied Logic, to appear
4. Bělohlávek R.: Lattices Generated by Binary Fuzzy Relations, Tatra Mountains Mathematical Publications 16/99 11–19
5. Ganter B., Wille R.: Formal Concept Analysis, Mathematical Foundations, Springer Verlag 1999, ISBN 3-540-62771-5

Integral representations in the form of neural networks with infinitely many units

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 *Model complexity of neural networks is investigated using tools from nonlinear approximation and integration theory. Estimates of network complexity are obtained from inspection of upper bounds on rates of approximation by neural networks with an increasing number of units. The estimates can be applied to a wide class of functions which can be represented as integrals with kernels corresponding to various types of computational units.*

1 Introduction

Integral transformations (such as Fourier, Gabor or Radon) play an important role in applied science. Also functions computable by neural-network units can be used as kernels of integral transformations. Originally, such transforms were used to prove the universal approximation property of neural networks, that is their capacity to approximate arbitrarily closely all continuous or $\mathcal{L}^p$-functions. Approximations of functions by networks with infinitely many perceptrons were derived from Radon transform [2,5] and by networks with Gaussian radial-basis functions from convolutions with the Gaussian kernel [16].

Integral transforms in the form of networks with infinitely many units can also be used to get some insight into the impact of the choice of type of network units on its model complexity. Such insights can be derived from inspection of upper bounds on rates of approximation of functions and on convergence of error functionals with increasing number of network units. A useful tool for deriving such upper bounds is a result from nonlinear approximation theory by Maurey [17], Jones [6] and Barron [1]. Various authors combined Maurey-Jones-Barron's theorem with suitable integral representations of functions in the form of networks with infinitely many units. Barron [1] considered functions representable as weighted Fourier transforms, Girosi and Anzelotti [4] studied convolutions with suitable kernels, Kůrková et al. [14] derived an integral representation in the form of a network with infinitely many Heaviside perceptrons for smooth compactly supported functions, and Kainen et al. [9] extended this representation to functions vanishing sufficiently rapidly at infinity.

In this paper, we present a unifying framework for estimation of model complexity for networks with general types of computational units. Using the reformulation of Maurey-Jones-Barron's theorem from [10] in terms of a norm tailored to the type of network units, we derive an upper bound on this norm holding for functions representable as integral transforms with arbitrary bounded kernels. This bound enables us to estimate network complexity for a wide class of functions and training data.

The paper is organized as follows. In section 2, we introduce notation and describe integral operators defined by computational units. In section 3, we state estimates of rates of approximation and convergence of error functionals over neural networks in terms of variational norms. In section 4, we present our main results on geometric characterization of variational norm and estimates of its magnitude for functions obtained by integral transforms with kernels corresponding to quite general computational units.

2 Integral operators induced by computational units

Neural-network units compute functions depending on parameters (such as weights, biases, centroids). So formally they can be described as mappings

$$\phi : \Omega \times Y \to \mathbb{R},$$

where Ω is a set of variables and Y is a set of parameters. Usually, $\Omega \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^p$.

Network units computing a two-variable function ϕ induce a mapping

$$\Phi : Y \to \mathcal{X},$$

from the set of parameters Y to a suitable function space $\mathcal{X}$; Φ is defined for all $y \in Y$ as

$$\Phi(y)(x) = \phi(x, y).$$

We denote by

$$\Phi(Y) = \{\phi(., y) \,|\, y \in Y\}$$

the set of all functions computable by the unit ϕ with parameters in the set Y. We write $s_\Phi = \sup_{y \in Y} \|\phi(., y)\|_{\mathcal{X}}$.

For example, *perceptrons with an activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ induce a mapping Φ_σ on $\mathbb{R}^{d+1}$ defined for $(v, b) \in \mathbb{R}^d \times \mathbb{R} = \mathbb{R}^{d+1}$ as

$$\Phi_\sigma(v, b)(x) = \sigma(v \cdot x + b).$$

Similarly, *radial-basis functions* with the radial function $\psi : \mathbb{R} \to \mathbb{R}$ induce a mapping Φ_ψ on $\mathbb{R}^{d+1}$ defined for $(v, b) \in \mathbb{R}^d \times \mathbb{R} = \mathbb{R}^{d+1}$ as

$$\Phi_\psi(v, b)(x) = \psi(b\|x - v\|).$$

One-hidden-layer feedforward neural networks with units computing the function ϕ can compute input-output functions from the set

$$\operatorname{span}_n \Phi(Y) = \left\{ \sum_{i=1}^{n} w_i \phi(., y_i) \,|\, w_i \in \mathbb{R},\, y_i \in Y \right\}.$$

The set $\Phi(Y)$ is sometimes called a *dictionary* of functions. The number n expresses the *model complexity*, it is the number of units in the hidden layer.

A network unit computing a function ϕ also induces an integral operator. Such an operator depends on a measure μ on Y and is defined for a function w in a suitable function space as

$$L_\phi(w)(x) = \int_Y w(y)\phi(x, y)d\mu(y).$$

Metaphorically, the operator L_ϕ assigns to an output-weight function w an input-output function of a one-hidden layer neural network with infinitely many units, where the units compute functions $\phi(., y)$ with $y \in Y$.

3 Rates of approximation and optimization by neural networks

Rates of approximation of multivariable functions and of convergence of error functionals over one-hidden-layer neural networks with an increasing number n of units can be estimated using Maurey-Jones-Barron's estimate [17,6,1]. Here, we use reformulation of this estimate from [10] in terms of a norm tailored to the type of network units.

This norm is defined for any bounded nonempty subset G of a normed linear space $(\mathcal{X}, \|.\|_\mathcal{X})$, as the Minkowski functional of the closed convex symmetric hull of G. It is called *G-variation* and is denoted $\|.\|_G$, i.e.,

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

where the closure cl is taken with respect to the topology generated by the norm $\|.\|_\mathcal{X}$ and conv denotes the convex hull. G-variation is a norm on the subspace of $\mathcal{X}$ formed by those f for which is $\|f\|_G$ finite.

The following proposition states basic properties of the variational norm: (i) follows directly from the definition and for (ii) see [14] and [7]. For other properties of variational norm see [11].

Proposition 1. *Let $(\mathcal{X}, \|.\|_\mathcal{X})$ be a normed linear space, G its nonempty bounded subset. Then:*
(i) for all $f \in \mathcal{X}$ representable as $f = \sum_{i=1}^k w_i g_i$ with all $g_i \in G$ and $w_i \in \mathbb{R}$, $\|f\|_G \leq \sum_{i=1}^k |w_i|$;
(ii) for all $f \in \mathcal{X}$, $\{f_i\}_{i=1}^\infty \subset \mathcal{X}$ satisfying $b_i = \|f_i\|_G < \infty$ for all i, $\lim_{i\to\infty} \|f_i - f\|_\mathcal{X} = 0$ and $b = \lim_{i\to\infty} b_i$, one has $\|f\|_G \leq b$.

The following theorem is a reformulation of the result by Maurey, Jones and Barron in terms of variational norm from [10] (see also [11]).

Theorem 1. *Let $(\mathcal{X}, \|.\|_\mathcal{X})$ be a Hilbert space, G its bounded nonempty subset, $s_G = \sup_{g\in G} \|g\|_\mathcal{X}$. Then for every $f \in \mathcal{X}$ and every positive integer n,*

$$\|f - \operatorname{span}_n G\|_\mathcal{X}^2 \leq \frac{s_G^2 \|f\|_G^2 - \|f\|_\mathcal{X}^2}{n}.$$

For $G = \Phi(Y)$ with Φ induced by various computational unit functions ϕ, Theorem 1 has been used as a tool for estimation of rates of approximation by one-hidden-layer neural networks. It can be also used for estimation of rates of convergence of error functionals.

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

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

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

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

Complexity of training data with respect to a type of computational units $\Phi(Y)$ can be measured by the *speed of decrease* of infima of error functionals over $\operatorname{span}_n \Phi(Y)$ 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.

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

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

$$f_\rho(u) = \int_V v \, d\rho(v|u),$$

where $\rho(v|u)$ is the *conditional (w.r.t. u) probability measure* on V (see, e.g., [3]). The global minimum of the empirical error $\mathcal{E}_z$ is achieved at any function interpolating the sample z.

The next theorem gives upper bounds on the speed of decrease of infima of error functionals over neural networks with an increasing number of units computing ϕ. Its proof utilizes Theorem 1 and an equivalence of minimization of error functionals with minimization of distances from certain functions (see [12]).

Theorem 2. *Let d, m, n be positive integers, both $U \subset \mathbb{R}^d$ and $V \subset \mathbb{R}$ be compact, $z = \{(u_i, v_i) \in U \times V \,|\, i = 1, \ldots, m\}$ with all u_i distinct, ρ be a nondegenerate probability measure on $U \times V$, and $\Phi(Y)$ be a bounded subset of $\mathcal{L}^2(U, \rho_U)$ with $s_\Phi = \sup_{y \in Y} \|\phi(., y)\|_{\mathcal{L}^2(U,\rho_U)}$. Then*

$$\inf_{f \in span_n \Phi(Y)} \mathcal{E}_\rho(f) - \mathcal{E}_\rho(f_\rho) \leq \frac{s_\Phi^2 \|f_\rho\|^2_{\Phi(Y)}}{n}$$

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

$$\inf_{f \in span_n \Phi(Y)} \mathcal{E}_z(f) \leq \frac{s_\Phi^2 \|h\|^2_{\Phi(Y)}}{n}.$$

So the critical factor for speed of convergence of empirical error functional is the magnitude of the $\Phi(Y)$-variation of the function from which the training data are chosen. Thus comparing magnitudes of $\Phi(Y)$-variations for various types of hidden unit functions ϕ, one can get some understanding how model complexity of neural networks is influenced by the choice of type of units.

4 Estimates of variation of functions representable as integrals in the form of networks with infinitely many units

To obtain comparisons of magnitudes of variational norms for various types of computational units, one needs some estimates of such norms. Our main result gives an upper bound on $\Phi(Y)$-variation for functions representable as integrals in the form of networks with infinitely many units computing ϕ, i.e., functions which are in the range of the integral operator L_ϕ.

The classes of functions which can be represented for various types of units as integrals in the form of infinite neural networks are quite large. For perceptron networks, such class contains all smooth compactly supported functions as well as all functions decreasing sufficiently rapidly at infinity (in particular, the Gaussian function) [14,9].

The proof of our main theorem is based on the following geometric characterization of variational norm. By $\mathcal{X}^*$ is denoted the *dual* of $\mathcal{X}$ (the space of all bounded functionals on $\mathcal{X}$) and by $G^\perp$ the *orthogonal compliment* of G, i.e., the set of all elements l of $\mathcal{X}^*$ such that for some $g \in G$, $l(g) \neq 0$.

Theorem 3. *Let $(\mathcal{X}, \|.\|_\mathcal{X})$ be a Banach space, G be its nonempty bounded subset and $f \in \mathcal{X}$ be such that $\|f\|_G < \infty$. Then*

$$\|f\|_G = \sup_{l \in \mathcal{X}^* - G^\perp} \frac{l(f)}{\sup_{g \in G} |l(g)|}.$$

The proof of Theorem 3 is based on Hahn-Banach Theorem and Proposition 1. A special case of Theorem 3 for a Hilbert space was proven in [15]. In the Hilbert space case, Theorem 3 gives some intuitive understanding to variational norm, it implies that G-variation is large for functions which are almost orthogonal to the set G.

Theorem 3 together with commutativity of bounded linear functionals with Lebesgue integration enable us to prove the following estimate of $\Phi(Y)$-variation for functions representable as networks with infinitely many units computing the function ϕ.

Theorem 4. *Let $\Omega \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^p$, μ be a Borel measure on Y, and $(\mathcal{X}, \|.\|_\mathcal{X})$ be a Banach space such that each its element is either a pointwise defined function on Ω or it is a class of μ-a.e. equivalent functions. Let $\phi : \Omega \times Y \to \mathbb{R}$ be a mapping such that $\Phi(Y) = \{\phi(., y) \,|\, y \in Y\}$ is a bounded subset of $\mathcal{X}$. Then for every $f \in \mathcal{X}$ such that $f = L_\phi(w)$ for some $w \in \mathcal{L}^1(Y, \mu)$, we have*

$$\|f\|_{\Phi(Y)} \leq \|w\|_{\mathcal{L}^1(Y,\mu)}.$$

Theorem 4 shows that for a function f representable as a neural network with infinitely many units computing ϕ, the $\Phi(Y)$-variation is bounded from above by the $\mathcal{L}^1$-norm of the output-weight function w. Various special cases of this theorem have been derived by a variety of proof techniques, but they all required some restrictions on the space $\mathcal{X}$ (e.g., boundedness of evaluation functionals [7,8]), on Ω and Y (e.g., compactness [14]), and on Φ and w (e.g., continuity [7,8]). Theorem 4 only assumes that the output-weight function w is in $\mathcal{L}^1(Y, \mu)$ (so that the upper bound exists) and that the set $\Phi(Y)$ is bounded (which is necessary for the definition of $\Phi(Y)$-variation).

Combining Theorem 4 with Theorem 1 we get an upper bound on rates of approximation by neural networks in Hilbert spaces.

Corollary 1. *Let $\Omega \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^p$, μ be a Borel measure on Y, and $(\mathcal{X}, \|.\|_\mathcal{X})$ be a Hilbert space such that each its element is either a pointwise defined function on Ω or it is a class of μ-a.e. equivalent functions. Let $\phi : \Omega \times Y \to \mathbb{R}$ be a mapping such that $\Phi(Y) = \{\phi(., y) \,|\, y \in Y\}$ is a bounded subset of $\mathcal{X}$. Then for every $f \in \mathcal{X}$ such that $f = L_\phi(w)$ for some $w \in \mathcal{L}^1(Y, \mu)$, we have*

$$\|f - \mathrm{span}_n \Phi(Y)\|^2_\mathcal{X} \leq \frac{s_\Phi^2 \|w\|^2_{\mathcal{L}^1(Y,\mu)} - \|f\|^2_\mathcal{X}}{n}.$$

Combining Theorem 4 and Proposition 1 with various integral representations (such as convolutions with Guassian kernels or representations in the form of plane waves from [14]) one can get estimates of variational norm with respect to many types of computational units for a wide class of functions.

Note that in this paper we have only stated the version of the Maurey-Jones-Barron's theorem for Hilbert spaces, however various extensions to other Banach spaces (e. g., space of continuous functions with the supremum norm and $\mathcal{L}^p$-spaces) have been proven. All these estimates depend on the variational norm and thus Theorem 4 can be used to obtain estimates of rates of approximation also in $\mathcal{L}^p$-spaces and in the space of continuous functions with the supremum norm.

Acknowledgement

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.

Reference

1. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory **39** (1993) 930–945
2. Carroll, S.M., Dickinson, B.W.: Construction of neural nets using the Radon transform. In: Proceedings of IJCNN. Volume I., New York, IEEE Press (1989) 607–611
3. Cucker, F., Smale, S. : On the mathematical foundations of learning. Bulletin of the AMS **39** (2002) 1–49
4. Girosi, F., Anzellotti, G.: Rates of convergence for radial basis functions and neural networks. In: Artificial Neural Networks for Speech and Vision (pp. 97–113), London, R. J. Mammone (Ed.), Chapman & Hall (1993)
5. Ito, Y.: Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. Neural Networks **4** (1991) 385–394
6. Jones, L.K.: A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. Annals of Statistics **24** (1992) 608–613
7. Kainen, P.C., Kůrková, V.: An integral upper bound for neural-network approximation. Submitted to Neural Computation, http://www.cs.cas.cz/research, Research Report V-1023, Institute of Computer Science, Prague (2008)
8. P. C. Kainen, V. Kůrková: Estimates of network complexity and integral representations. In *ICANN 2008* LNCS 5163, (Eds. Kůrková, R. Neruda, J. Koutník) (pp. 31-40). Berlin, Heidelberg: Springer-Verlag, 2008
9. Kainen, P.C., Kůrková, V., Vogt, A.: Integral combinations of Heavisides. Mathematische Nachrichten (2008) (to appear)
10. Kůrková, V.: Dimension–independent rates of approximation by neural networks. In Warwick, K., Kárný, M., eds.: Computer–Intensive Methods in Control and Signal Processing: Curse of Dimensionality (pp. 261–270), Birkhauser, Boston (1997)
11. Kůrková, V.: High-dimensional approximation and optimization by neural networks, chapter 4. In: Suykens, J., Horváth, G., Basu, S., Micchelli, C., Vandewalle, J., eds.: Advances in Learning Theory: Methods, Models and Applications (pp. 69–88), IOS Press, Amsterdam (2003)
12. Kůrková, V.: Estimates of data complexity in neural-network learning. In: J. Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plil, eds.: SOFSEM 2007, LNCS 4362 (pp. 377–387), Berlin, Heidelberg: Springer-Verlag (2007)
13. Kůrková: Minimization of error functionals over perceptron networks. Neural Computation **20**(1) (2008) 252–270
14. Kůrková, V., Kainen, P.C., Kreinovich, V.: Estimates of the number of hidden units and variation with respect to half-spaces. Neural Networks **10** (1997) 1061–1068
15. Kůrková, V., Savický, P., Hlaváčková, K: Representations and rates of approximation of real-valued Boolean functions by neural networks. Neural Networks **11** (1998), 651–659
16. Park, J., Sandberg, I. W.: Approximation and radial-basis-function networks. Neural Computation **5**, 305–316, 1993.
17. Pisier, G.: Remarques sur un resultat non publié de B. Maurey. Seminaire d'Analyse Fonctionelle 1980-81 **I**(12) (1981)

Axiomatizovaná architektúra znalostí a dokazovanie jej splniteľnosti v UML modeloch

Miroslav Líška
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava, Slovenská Republika
miroslav_liska@formal-analysis.com

Abstrakt. *Cieľom tohto príspevku je snaha skvalitniť modelom riadený softvérový proces prostredníctvom zlepšenia návrhu UML modelov. Hlavou myšlienkou pre splnenie tohto cieľa je vytvorenie axiomatickej architektúry znalostí, ktorá by znalosti o jednotlivých doménach a objektoch reálneho sveta uchovávala vo forme rôznych matematických teórií, ktorých splniteľnosť by bolo možné v študovanom UML modeli dokazovať. Takúto matematickú metódu je následne možné posunúť do kontextu metód umelej inteligencie, kde by uvedenú metódu bolo možné vnímať ako expertný systém, ktorý bude tvorca pri návrhu UML modelu využívať. Prototyp takého softvéru bude predstavený na záver tohto príspevku.*

1 Úvod

Aj keď správny UML model nie je hneď postačujúcou podmienkou pre vytvorenie správneho výsledného informačného systému, tak určite je podmienkou minimálne nutnou. Softvérový proces založený na modelom riadenom vývoji postupne produkuje dokumenty obsahujúce UML modely (analytické modely, návrhárske modely), medzi ktorými je vhodné dodržiavať presne definované závislosti, aby bolo možné dosiahnuť zhodu medzi špecifikáciou a realizáciou produktu. Z uvedeného vyplýva, že ak je niektorý model je vytvorený nesprávne, resp. neúplne, tak sa táto chyba v konečnom dôsledku prejaví na výslednom produkte, ktorý je potrebné prepracovať. V nasledujúcej kapitole sa pokúsim vymenovať pár vybratých problémov, ktoré sa priamo podieľajú na správnosti výsledného UML modelu.

2 Vybrané problémy návrhu UML modelov

2.1 Komplexnosť predmetnej domény

Čím je predmetná doména komplexnejšia (zložitejšia), tým je komplexnejšia aj jej špecifikácia vo forme UML modelov. V nich je potrebné zachytiť všetky podstatné cieľe, požiadavky, funkcionalitu, biznis logiku predmetnej domény spolu z jej všetkými podstatnými obmedzeniami a mnoho ďalších vlastností, ktoré musí výsledný informačný system zohľadňovať. Navyše, takmer vždy musí nový informačný system odpovedať nielen akuálnej predmetnej doméne, ale aj pravdepodobnej budúcej, do ktorej je predpoklad, že bude rozšírený. Z uvedeného jasne vyplýva, že je vhodné, aby tvorca UML modelov mal dostatočné znalosti o predmetnej doméne, či o pravdepodobných budúcich doménach, aby bol jeho návrh UML modelov čo najefektívnejší. Avšak, aj keď takýto tvorca UML modelov v organizácii existuje, problémom je, že jeho znalosti sú implicitné (tj. len v jeho hlave), a ak organizácia potrebuje viac takýchto rolí pre viac projektov, prípadne ak takýto UML tvorca zmení prácu, tak má organizácia reálny problém [1].

2.2 Presnosť významu UML modelu

Problémom je semiformálna architektúra jazyka UML, ktorá jednak zapríčiňuje možnosť rozličnej interpretácie UML modelu (tj. rozličné pochopenie významu), ale hlavne neumožňuje vykonať presnú analýzu pravdivosti UML modelu prostredníctvom matematickej logiky a jej rôznych konceptov (konzistencia, dokázateľnosť a iné) [2].

V súčasnosti existuje viacero formálnych metód, ktoré spresňujú jazyk UML prostredníctvom úplnej formalizácie jeho syntaxe a sémantiky, avšak ich použitie v reálnych komerčných projektoch je obmedzené a navyše sa sústreďuje len na samostný jazyk UML [3, 4, 5, 6].

3 Axiomatická architektúra znalostí

Na základe uvedených vybratých problémov je zrejmé, že ich riešenie spočíva vo vysporiadaní sa z množstvom znalostí o rôznych predmetných doménach, ďalej je potrebné tieto znalosti explicitne špecifikovať v nejakom jazyku tak, aby sa dalo rozhodnúť, či študovaný UML model je vytvorený na základe týchto znalostí (je v nich splnený). Na základe týchto požiadaviek na riešenie je akiste zrejmé, že ak by boli jednotlivé znalosti o doménach reprezentované matematickými teóriami, tak by bolo možné dokazovať, či nejaký UML model je modelom týchto teórií, tj. či sú tieto teórie v UML modeli splenené.

3.1 Motivácia vytvorenia

V reálnom svete podliehajú reálne objekty existencii viacerým rôznym pravidlám (teóriám) súčasne. Tak napr. na auto, ktoré sa pohybuje po ceste, pôsobí gravitácia, súčasne musí toto auto vyhovovať pravidlám pre bezpečnosť, rovnako na neho pôsobia korózne vplyvy počasia, klesá jeho cena súvisiaca s jeho opotrebovaním, atď. ... Tieto rôzne pravidlá (teórie) je efektívne študovať jednak v ich špecifickosti (oddelene), ale zároveň aj z pohľadu ich interakcie. Fyzici skúmajú auto z pohľadu jeho fyzikálnych vlastností, ekonómovia z pohľadu jeho ekonomickej hodnoty, bezpečnostní technici zase skúmajú auto z jeho bezpečnostných hľadísk. Nie je možné, resp. nie je efektívne, aby jeden človek ovládal všetky tieto teórie. A to platí z dôvodu buď zložitosti samotných teórií, alebo nie je vždy nutné študovať objekt z pohľadu všetkých teórií súčasne.

Na základe tejto oddelenosti teórií je potom možné tieto teórie znova použiť pri štúdiu v iných prípadoch. Tak napr. konštruktéri lietadla akiste už môžu využiť vypracovanú gravitačnú teóriu (špecifikovanú pre potreby štúdia fyzikálnych vlastností auta) aj pri štúdii fyzikálnych vlastností lietadla, pričom ju pravdepodobne rozšíria o aspekt letu, pádu, atď. ... Pokiaľ by sa ale zistili nejaké nezrovnalosti správania sa lietadla podľa tejto gravitačnej teórie, a objavili by jej ďalšie vlastnosti, akiste by stálo za pokus použiť tieto jej nové vlastnosti aj pri výskume samotného auta.

3.2 Splniteľnosť teórií v UML modely

UML model vizualizuje rôzne modely (systémy) reálneho sveta prostredníctvom štruktúr, alebo správania sa [7]. Raz je štrukturálny UML diagram použitý na modelovanie organizačnej štruktúry, inokedy je použitý na modelovanie knižničného systému, či štruktúry aplikácie pre evidenciu zamestnancov, atď.. Modely správania sa zas môžu opisovať procesy schvaľovania faktúr, či interakciu používateľa so systémom na spracovanie produktov, atď. Akiste doména aplikácie UML nie je ohraničená.

Táto skutočnosť priamo predurčuje validáciu UML diagramov jednak oproti pravidlám tvorby UML (teória UML) ale zároveň aj oproti iným pravidlám (teóriám) reálneho sveta (napr. schvaľovanie úverov, zákony implementované v informačných systémoch, gravitačné teórie, atď.).

3.3 Formálny model AKA4UML

Aby bola táto architektúra čo najviac otvorená, či použiteľná, resp. analyzovateľná, použijem všeobecne používaný základný matematický jazyk predikátovej logiky prvého rádu [8, 9]. Budem tak schopný používať výhody predikátovej logiky prvého rádu, a zároveň tým umožním aj budúce efektívne rozšírenie konceptu AKA4UML matematickými konceptmi, ktoré rozširujú predikátovú logiku prvého rádu (napr. modálne logiky, či extenzionálne a intenzionálne špecifikovanie sémantiky a iné) [10].

3.3.1 Definícia AKA4UML (Axiomatic Knowledge Architecture for UML)

Nech existuje relačná štruktúra AKA4UML= $\{T_1, T_2, ...T_n\}$, kde T_i je teóriou nejakého doménového konceptu (napr. ***teória UML***, ***teória schvaľovania faktúr***), obsahujúca množinu špeciálnych axiómov tejto teórie Ti=$\{A_{i1}, A_{i2}$ $...A_{ij}, A_{ij+1}, ... A_{im}\}$, j<=m.

Potom hovoríme, že UML model (jeho interpretácia prostredníctvom jazyka L) je splnený v AKA4UML a píšeme M(UML)|=AKA4UML, ak je v M(UML) splnený každý špeciálny axióm teórie T_i, teda $\forall$ i,j (M(UML) |= A_{ij}, A_{ij} T_i, i<=n, j<=m) [11]. Stratégia dokazovania splniteľnosti bude založená na dôkaze sporom, teda budem dokazovať negáciu nasledovnej kontradikcie

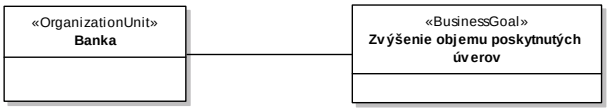
$$T_1 \wedge T_2 \wedge ... \wedge T_n \wedge \neg \mathtt{M}(\mathrm{UML}).$$

Aby som konkrétnejšie predstavil použitie AKA4UML architektúry, pokúsim sa ju teoreticky vysvetliť na konkrétnom príklade:

Nech existuje jednoduchý diagram UML reprezentujúci skutočnosť, že banka má za cieľ zvýšiť objem úverov. Úlohou je vytvoriť spôsob, pomocou ktorého by bolo možné dokázať, že je UML model vytvorený správne podľa nasledovných pravidiel:

1. organizačná jednotka a jej biznis cieľ musia byť reprezentované ako triedy spojené s asociáciou,
2. trieda reprezentujúca organizačnú jednotku musí mať stereotyp OrganizationUnit,
3. trieda reprezentujúca cieľ organizačnej jednotky musí mať stereotyp BusinessGoal.

Pre jednoduchosť, ktorá však nezníži názornosť metódy, bude postačovať jednoduchý diagram UML znázornený na obrázku Obr. 1.



Obr. 1. UMLdiagram1 - Stratégia nejakej banky.

Pozn.: táto úloha sa dá riešiť aj prostredníctvom vytvorenia UML profilu obsahujúceho stereotypy OrganizationUnit a BusinessGoal, ktoré by obsahovali definované OCL obmedzenia (trieda so stereotypom OrganizationUnit musí byť spojená s triedou so stereotypom BusinessGoal) a použiť validačné možnosti CASE nástrojov prostredníctvom OCL jazyka. Účelom tejto práce je však maximalizácia formálneho charakteru metódy a z nej plynúcich možností, ktorými formálne metódy disponujú, preto sa nebudem validáciou prostredníctvom jazyka OCL zaoberať [12].

Ako riešiť uvedenú úlohu prostredníctvom formálneho prístupu AKA4UML, ktorý som definoval vyššie v texte?

Pri pohľade na tento model je zrejmé, že diagram musí byť správny po stránke pravidiel jazyka UML, teda budem potrebovať ***teóriu 1 - UML (T_1)***:

Predikáty = {C: byť triedou, A: byť binárnou asociáciou, S: byť stereotypom}

Axiómy =

$\exists x \exists y (C(x) \vee A(x,y) \vee S(x,y))$ Existuje buď trieda, asociácia alebo stereotyp.	(A1.1)
$\forall x \forall y (A(x,y) \Rightarrow (C(x) \wedge C(y)))$ Pre každú asociáciu platí, že zdroj a koniec asociácie sú triedy.	(A1.2)
$\forall x \forall y (S(x,y) \Rightarrow C(x))$ Ak existuje stereotyp, potom musí patriť triede.	(A1.3)

Okrem toho bude musieť model zodpovedať aj biznis stratégii (napr. každá organizačná jednotka má svoj biznis cieľ, tento cieľ nie je splnený z dôvodu problémov, ... atď.), teda budem potrebovať aj ***teóriu biznis stratégie (T_2)***:

Predikáty = {OU: byť organizačnou jednotkou, BG: byť biznis cieľom}

Axiómy =

$\forall x \exists y BG(x) \Rightarrow OU(y)$ Pre každý biznis cieľ existuje organizačná jednotka.	(A2.1)
$\exists x OU(x)$ Existuje organizačná jednotka.	(A2.2)

Keďže stereotypy sú jednou z možností, ako rozširovať jazyk, a môžeme ho chápať aj ako samostatný koncept, vytvorím aj samostatnú ***teóriu rozšírenia UML modelu pre potreby modelovania stratégie*** prostredníctvom UML jazyka ***(T_3)***:

Axiómy =

$\forall x(OU(x) \Rightarrow (C(x) \wedge S(x, \text{OrganizationUnit}))$ Ak je element organizačná jednotka, potom musí byť reprezentovaná ako trieda so stereotypom OrganizationUnit.	(A3.1)
$\forall x(BG(x) \Rightarrow (C(x) \wedge S(x, \text{BusinessGoal})))$ Ak je element biznis cieľ, potom musí byť reprezentovaná ako trieda so stereotypom BusinessGoal.	(A3.2)
$\forall x \forall y((BG(x) \Rightarrow OU(y)) \Rightarrow A(x, y))$ Ak je element biznis cieľ organizačnej jednotky, tak tieto musia byť navzájom spojené asociáciou.	(A3.3)

Pozn.: Tieto uvedené teórie (ich špeciálne axiómy) definujem intuitívne pre potreby tohto príspevku, nepovažujem ich za všeobecne platné, či akceptovateľné. Napr. jazyk UML je v súčasnosti formalizovaný viacerými metódami, ako som sa snažil naznačiť aj v tejto práci [13, 14, 15, 16]. Z tohto dôvodu nazvem formalizáciu UML v tomto príspevku ako ***teória 1- UML***, aby som zvýraznil skutočnosť, že je definovaná mnou intuitívne. Pre potreby tohto príspevku je to jednak postačujúce, jednak je to aj zámer. Nejde mi o konkrétnu presnú teóriu UML, skôr mi ide o definovanie metódy validácie UML modelov oproti viacerým teóriám súčasne (AKA4UML).

Keď to zhrniem, mám špecifikované tri teórie T_1, T_2, T_3 v jazyku L (predikátová logika), ktoré vytvárajú axiomatizovanú architektúru znalostí, teda AKA4UML=$\{T_1, T_2, T_3\}$. Aby som mohol vykonať dôkaz splniteľnosti týchto teórií v UML modeli z obrázka Obr.1, interpretujem tento diagram rovnako do jazyka predikátovej logiky a označím ho ako interpretáciu jazyka M_1, teda M_1 = M(UMLdiagram1), ktorý bude mať nasledovnú špecifikáciu:

M1={M, Pr}, kde M je množina indivíduí a Pr je realizácia predikátových symbolov, teda M={Banka, Zvýšenie objemu poskytovaných úverov}, Pr={C(Banka), C(Zvýšenie objemu poskytovaných úverov), S(Banka, OrganizationUnit), S(Zvýšenie objemu poskytovaných úverov, BusinessGoal), A(Banka, Zvýšenie objemu poskytovaných úverov)}.

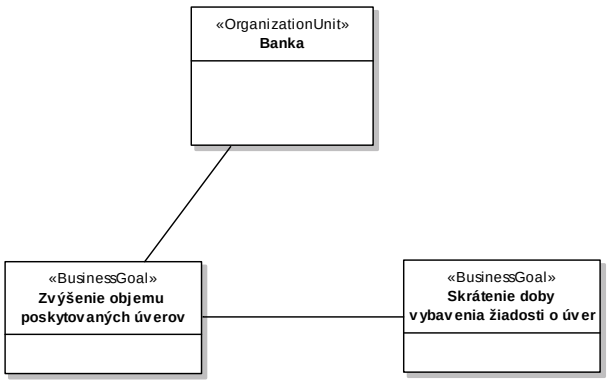
Potom môžem dokazvať, či platí

M(UMLdiagram1)|=AKA4UML,

teda, či je UMLdiagram1 modelom teórii $T_1 \wedge T_2 \wedge T_3$ (axiomatickej architektúry znalostí), ktorý je možné dokázať napr. rezolvenciou prostredníctvom dokázania, že formula

$$T_1 \wedge T_2 \wedge T_3 \wedge \neg M(\text{UML})$$

je kontradikcia. Rezolvenciu som uviedol zámerne, nakoľko je táto metóda vhodná na automatizáciu. V tejto práci nebudem uvádzať podrobný dôkaz, pravdou však je, že je AKA4UML v UML modely z Obr.1 splnená. Aby bola metóda ešte názornejšia, uvediem aj UML model 2, na Obr.2, v ktorom AKA4UML splnená nie je, nakoľko nie je v ňom splnený axióm $A_{2.1.}$.

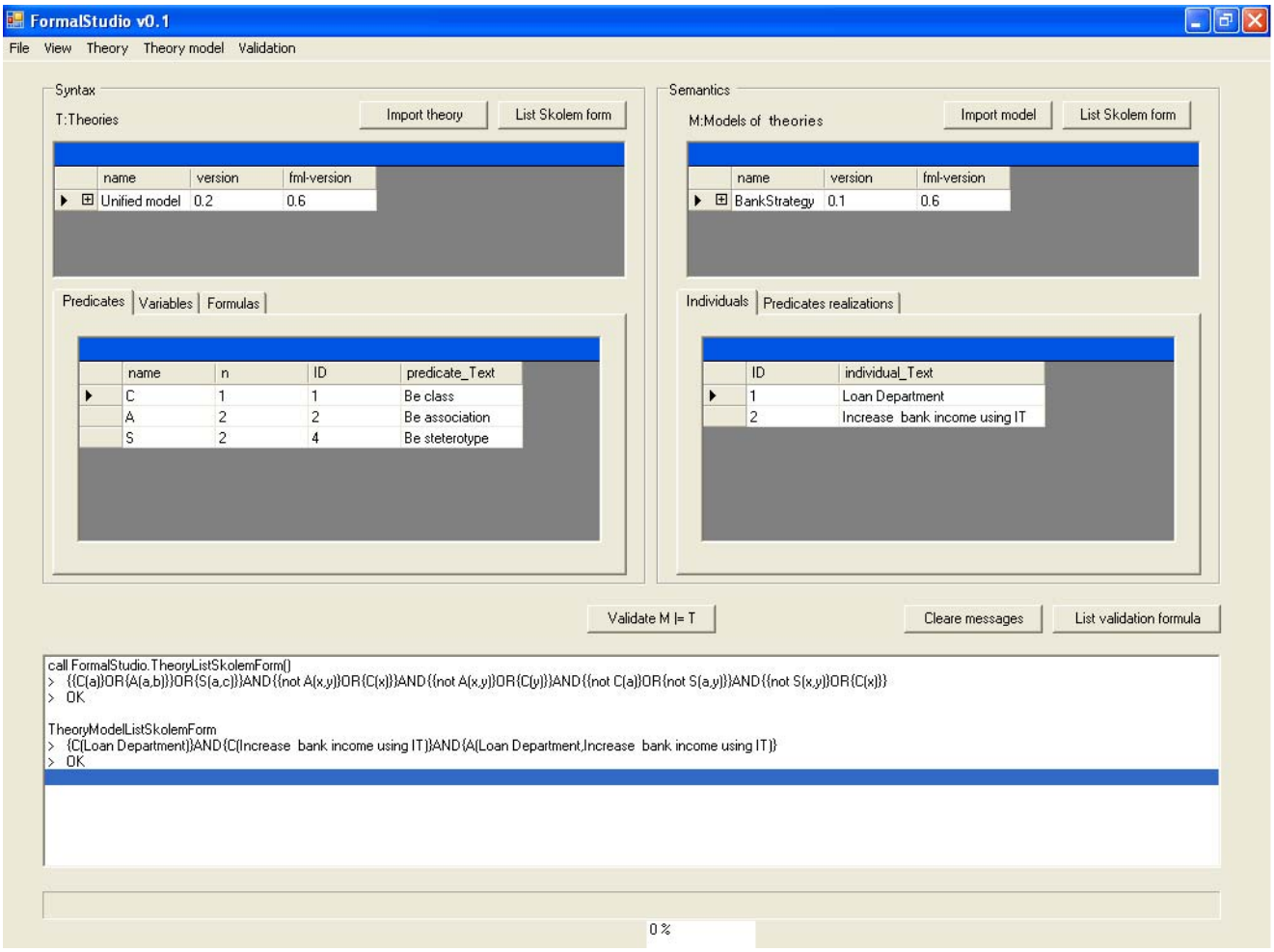


Obr. 2. UMLdiagram2 - Stratégia nejakej banky.

3.4 Prínosy a možnosti AKA4UML

V tejto kapitole sa pokúsim definovať prínosy validácie UML modelov oproti architektúre AKA4UML. Za hlavné prínosy považujem:

1. definovanie spôsobu, akým je možné validovať UML modely nielen oproti pravidlám jazyka UML, ale aj oproti iným teóriám z reálneho sveta (pretože UML vždy vyjadruje nejaký koncept z reálneho sveta),
2. formálny model tejto architektúry je založený na jednoduchom, všeobecne uznávanom koncepte predikátovej logiky prvého rádu (teória, model teórie …), teda má charakter byť zrozumiteľný širokému spektru formálnych analytikov,
3. architektúra AKA4UML je efektívne rozšíriteľná o modálne logiky, o extenzívne a intenzívne špecifikovanie sémantiky,



Obr. 3. Používateľské prostredie nástroja FormalStudio.

4. rovnako je architektúra AKA4UML efektívne rozšíriteľná nielen pre potreby validácie UML modelov, ale aj pre validáciu iných interpretovaných sformalizovaných entít (napr. formalizovaný text požiadaviek na informačný systém, či iné...).

3.5 AKA4UML v kontexte expertného system

V súvislosti so špecifikáciou architektúry AKA4UML som sa zameral aj na vytvorenie podporného softvérového nástroja pre túto metódu validácie UML modelov [17, 18].

Na ľavej strane používateľského rozhrania je vidieť zoznam teórii spolu so špecifikáciou ich jazyka (predikáty, premenné, axiómy) a na pravej strane je zas vidieť naimportovaný a interpretovaný UML model spolu s jeho detailnou špecifikáciou (individuá, realizácia predikátových symbolov). Hlavnou funkcionalitou tohto nástroja je spustenie akcie dôkazu splnenia teórií v modeli teórií.

Prostredníctvom tohto nástroja môže tvorca modelu dokazovať, či axiomatická architektúra znalostí je splnená v jeho modeli. Ak nie, tak môže model upravovať dovtedy, kým sa splnenie dokáže.

4 Budúca práca

Verím, že som svoje ciele definované v tomto príspevku naplnil aspoň na postačujúcej úrovni odpovedajúcej účelu tohto príspevku. Uvedomujem si potrebu ďalšieho rozvoja formálnej metódy AKA4UML pre jej reálnejšie uplatnenie vo vývojovom procese, rovnako ďalšieho štúdia predikátovej logiky, jazyka UML, či štúdia iných formálnych metód, ktorých cieľom je zefektívnenie špecifikačného a validačného procesu UML diagramov.

V blízkej budúcnosti sa rovnako budem orientovať na možnosť importovania teórie, či modelu teórie z RDF dokumentu, čo je dnes stavebný kameň sémantického webu.

Literatúra

1. P. Návrat, et al, *Artificial Intelligence*. Slovak University of Technology in Bratislava, 2002.
2. OMG, *UML Infrastructure*, v. 2.1.1. Formal/07-02-06, 2007.
3. E. Amir, "Object-Oriented First-Order Logic". In *Proceedings of the {IJCAI}-99 Workshop on Nonmonotonic Reasoning, Action and Change*, 1999.
4. R. Marcano, N. Levy, "Using B formal specifications for analysis and verification of UML/OCL models". Laboratoire PRISM, Université de Versailles.
5. M. Richters, "A Precise Approach to Validating UML Models and OCL Constraints". *PhD thesis*, Bremen University, 2002.
6. N. Amálio, S. Stephney, F. Polack, "Formal proof from UML model". In: *ICFEM'04*, Seattle, USA, 2004.

7. G. Booch, I. Jacobson, J. Rumbaugh, *The Unified Modelling Language Reference Manual*. Addison Wesley. 2004.
8. V. Kvasnička, J. Pospíchal, *Mathematical logics*. Slovak University of Technology in Bratislava, 2006.
9. P. Štepánek, *Predicate logics*. Faculty of Mathematics and Physics at Charles University in Prague, 2000.
10. P. Cmorej, *An Introduction into logical syntax and semantics*. Iris, 1998.
11. M. Líška, "UML model validation with axiomatic knowledge architecture". *PhD thesis*, Slovak University of Technology in Bratislava, 2007.
12. OMG, *Object Constraint Language*, v. 2.0. Formal/06-05-01, (2006).
13. R. Laleau, A. Mammar, "An overview of a method and its support tool for generating B specifications from UML Notations". – In: *Proceedings of the 15th Int. Conf. on Automated Software Engineering, ASE'2000*, Grenoble, France, September, 2000.
14. H. Ledang, J. Souquiéres, "Integration of UML Views using B Notations". LORIA - Université Nancy, France.
15. J. Woodcock, J. Davies, *Using Z*. Specification, Refinement and Proof. Prentice Hall, 1996.
16. A. Evans, R. France, K. Lano, B. Rumpe, "The UML as formal modeling notation". In: *Computer Standarts & Interfaces*, Seattle, USA, 1998.
17. M. Líška, "Constructing multi-theories expert system for UML models validation". In *1st Workshop on Intelligent and Knowledge oriented Technologies*, Bratislava, SAV, 2006.
18. M. Líška, "Constructing expert system for Model Driven Development". In *Informatics'07*, Slovak Society for Applied Cybernetics and Informatics Bratislava, (2007).

Evaluation of XPath fragments using lambda calculi

Pavel Loupal and Karel Richta

Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University
Karlovo nám. 13, 121 35 Praha 2, Czech Republic
loupalp@fel.cvut.cz, richta@fel.cvut.cz

Abstract. *XML Path Language (XPath) is the most important standard for navigation within XML documents nowadays. In this paper we present the state of our current research that is focused on using a functional framework based on simply typed lambda calculi and a general type system - XML-λ - for description of semantics of a query language. With such formalism we are able to describe the semantics of all language constructs and evaluate XPath queries using the XML-λ virtual machine.*

1 Motivation and problem statement

The World Wide Web Consortium designed the XML language [2] for an exchange of data on the Web and elsewhere. Then, there arose many subsequent proposals of query languages. One of the most important of them is the XML Path Language – XPath [4]. It is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. Later on, its extended version (denoted as 2.0) became a crucial part of the XQuery 1.0 standard [1].

As all experimenters who try to develop an efficient implementation of the standard we have faced this problem during our work on a prototype of a native XML database management system ExDB [10] within our research group. While the XPath engine is usually heavily used it is worth to put extra effort into its optimal design.

With respect to the fact that we deal for a long time with a functional data model for XML and its properties our current research is focused on using such model for evaluating XPath queries. Proposed paper suggest a way how to transform XPath queries into their functional version and then evaluate it using a functional virtual machine. Actually, it is only an outline of the research (ergo "Work in Progress") but so far the result is promising.

Our long-term goal is to extend the scope of this work to XPath 2.0 and further to XQuery 1.0 – here we plan to formalize so called *Core XQuery 1.0* [9] that is formed mainly by the FLWOR (For–Let–Where–OrderBy–Return) expression and XPath node selection. Results of this research are also expected to be published as a consistent work in the doctoral thesis of the first author.

2 Related work

The essential document – XML Path Language Version 1.0 [4] – defines the syntax of the language and its informal semantics. The most important part is the description of the data model used (XML Infoset [5]) and denotation of all location paths (i.e. axis, node tests and predicates). The standard also defines few basic functions for manipulation with XML documents.

Successive works propose various extended data models and discuss the efficiency and complexity of XPath evaluation algorithms. From the point of XPath's semantics, Wadler discussed the denotational semantics, for example here [13] or [14], seen from an XSLT point of view.

Gottlob et al. [7, 8] propose their own denotational semantics and discuss mostly the time and space complexity of XPath evaluation algorithms using their proposals.

3 Prerequisite specifications

We expect that the reader is familiar with XML, XPath and other W3C's specifications. XML-λ is not so known and therefore, for convenience, we repeat basic facts in following sections.

3.1 XML Path 1.0

XML Path 1.0 (XPath) [4] is a W3C standard for locating nodes in an XML tree. It is a variable-free language based on so called location steps. By successive evaluation of these steps we obtain a result. The result can be either a node-set, a boolean, a number or a string value.

For purpose of this paper we consider only part of this standard sometimes referred as "Core XPath". Such subset is defined in various ways, see for example [6], [7] or [13]. Here, we use the version shown in Figure 1 written using Extended Backus-Naur Form.

For example, for the "XMP" Use Case published in the XML Query Use Cases [3] we may write a query that selects all books with given title and price as

$$doc(\text{"}bib.xml\text{"})/bib/book$$
$$[\ title = \text{"}TCP/IP\ Illustrated\text{"}\ and\ price\ >\ 399\ ]$$

Note that for readers familiar with XPath 1.0 it should not be difficult to understand the meaning of the query while the syntax of these two languages is very similar.

We can imagine a virtual XPath machine, whose principle is shown in the Figure 2. This machine evaluates CoreXPath queries and returns a set of XML nodes. In the next section we describe how it will be simulated by our XML-λ Framework.

$CoreXPath$ ::= $RelativePath \mid AbsolutePath$
$RelativePath$ ::= $\epsilon \mid Step\ (\text{'/'}\ Step)*$
$AbsolutePath$ ::= '/' $RelativePath$
$Step$ ::= $AxisName$ '::' $NodeTest\ Predicate*$
$AxisName$ ::= 'ancestor' | 'ancestor-or-self' | 'attribute' | 'child' | 'descendant' | 'descendant-or-self' | 'following' | 'following-sibling' | 'namespace' | 'parent' | 'preceding-sibling' | 'preceding' | 'self'
$NodeTest$ ::= '$*$' | $QName$
$Predicate$ ::= '[' $Expr$ ']'
$Expr$ ::= Further ommited. See [4, rule (14)].

Fig. 1. Core XPath 1.0 Grammar in EBNF.

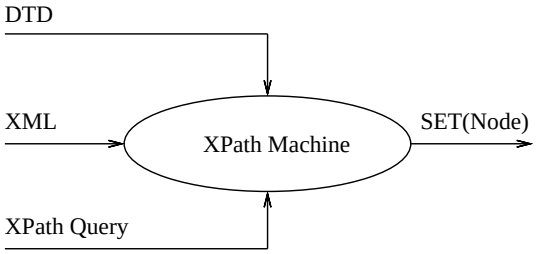


Fig. 2. Virtual XPath Machine Principle.

3.2 XML-λ framework

XML-λ is a proposal published by Pokorný [11, 12]. It is denoted as a functional framework for manipulating XML. The most important difference to existing W3C specifications it uses a functional data model instead of tree- or graph-oriented model. Author's primary motivation was to see XML documents as a database that conforms to an XML schema (defined, for example, by DTD) and to gain a possibility to use a functional language, particularly a simply typed λ-calculus enriched with n–tuples, as a query language for such database.

The cornerstones of the framework are three components related to its type system: *element types*, *element objects* and *abstract elements*. Figure 3 shows the relationships between basic terms of W3C standards and the XML-λ Framework. We can imagine these components as the data dictionary in relational database systems.

Element types are derived from a particular DTD, i.e. for each element defined in the DTD there exists exactly one element type in the set of all available element types (called T_E). Consequently, we denote E as a set of *abstract elements*. Set members are of element types.

Element objects are basically functions of type either $E \rightarrow String$ or $E \rightarrow (E \times \ldots \times E)$. Application of these functions to an *abstract element* allows access to element's content. *Elements* are, informally, values of *element objects*, i.e. of functions. For each $t \in T_E$ there exists a corresponding t-object.

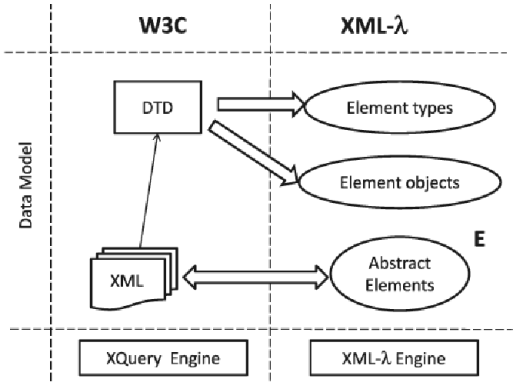


Fig. 3. Relationship Between W3C and XML-λ Models.

For convenience, we add a "nullary function" (also known as 0-ary function) into our model. This function returns a set of all abstract elements of a given element type from an XML document.

Finally, we can say that in XML-λ the instance of an XML document is represented by a subset of E and a set of respective t-objects.

As an example, we can rewrite the XPath query from Section 3.1 using XML-λ as

$$xmldata(\text{"}bib.xml\text{"});$$
$$lambda\ b\ (/bib/book(b)\ and\ b/title\ =\ \text{"}TCP/IP\ Illustrated\text{"}\ and\ b/price\ >\ 399);$$

Such query is then evaluated as a λ-term (see [11, 12]).

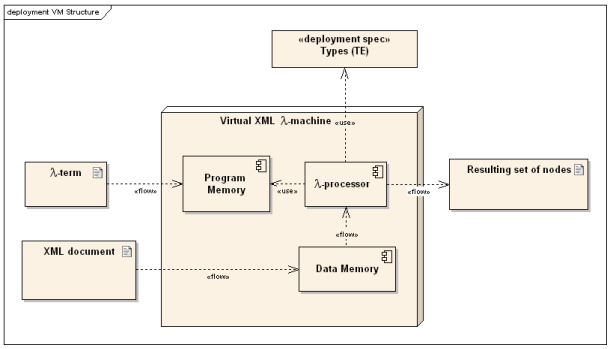


Fig. 4. Virtual XML-λ Machine Principle.

4 Solution proposal

We split the problem into two parts: (1) Transformation of XPath query into a λ–term and (2) evaluation of acquired term in the XML-λ virtual machine (VM). Formally, the

first step can be defined as the mapping $Comp$ from the input query into a λ–term, which serves as a code for the XML-λ VM. The translation is driven by required structure – let us suppose, that D is the required DTD, and we get the following signature:

$$Comp_D \ : \ CoreXPath \ \rightarrow \ \lambda{-}term$$

In the second step we evaluate resulting λ–term by XML-λ VM. In general, the XML-λ VM can be formally described as a pair of mappings (the expression $SET(X)$ denotes a set of all sets containing elements of the type X):

$$\begin{aligned} Eval_M &: \lambda{-}term \times XML{-}Doc \rightarrow \\ &\rightarrow \ SET(XML{-}Node) \\ Next_M &: \lambda{-}term \times XML{-}Doc \rightarrow \\ &\rightarrow \ SET(XML{-}Doc) \end{aligned}$$

The expression $Next_M[\![t]\!]$ denotes the meaning of λ-term t as a mapping from the input state s of XML-λ VM into a new state $Next_M[\![t]\!](s)$. The computation can produce a set of resulting XML-nodes $Eval_M[\![t]\!](s)$. The whole picture is a combination of both steps (translation and interpretation), so resulting mapping Sem_D is:

$$\begin{aligned} Sem_D \ : \ CoreXPath \ &\times XML{-}Doc \ \rightarrow \\ &\rightarrow \ SET(XML{-}Node) \end{aligned}$$

$$Sem_D[\![expr]\!] = Eval_M[\![Comp_D[\![expr]\!]]\!]$$

where $expr$ is an arbitrary CoreXPath expression.

The expression $Sem_D[\![expr]\!]$ denotes the meaning of CoreXPath query $expr$ on the well-formed document (well-formed according to the XML-schema D) as a mapping from an XML-document into resulting set of XML-nodes. E.g.:

$$\begin{aligned} Sem_D[\![\text{"}/bib/book[1]/title\text{"}]\!](bib.xml) \ &= \\ &= \{\text{"}TCP/IP \ Illustrated\text{"}\} \end{aligned}$$

The transformation $Comp_D$ can be defined by particular evaluation steps accordingly to the CoreXPath grammar (abbreviated):

$$\begin{aligned} &Comp_D \ [\![\epsilon]\!] = \lambda x(.) \\ &Comp_D \ [\![Step/Path]\!] = \\ &\quad = \lambda x(Comp_D[\![Path]\!](Comp_D[\![Step]\!])) \\ &Comp_D \ [\![/]\!] = \lambda x(root(x)) \\ &Comp_D \ [\![/Path]\!] = \\ &\quad = \lambda x(Comp_D[\![Path]\!](Comp_D[\![/]\!])) \\ &Comp_D \ [\![self :: *]\!] = \lambda x(//) \\ &Comp_D \ [\![self :: x]\!] = \lambda x(x) \\ &Comp_D \ [\![self :: x[i]]\!] = \lambda x(x[i]) \\ &Comp_D \ [\![parent :: p/x]\!] = \lambda x(p/x) \\ &Comp_D \ [\![child :: x]\!] = \lambda x(./x) \\ &Comp_D \ [\![ancestor :: p//x]\!] = \lambda x(p/x) \\ &Comp_D \ [\![descendant :: x]\!] = \lambda x(.//x) \end{aligned}$$

$$\begin{aligned} &Comp_D \ [\![preceding{-}sibling :: p[i]//x]\!] = \\ &\quad = \ \lambda x(p[i-1]/x) \ if \ i > 2 \\ &Comp_D \ [\![following{-}sibling :: p[i]//x]\!] = \\ &\quad = \ \lambda x(p[i+1]/x) \ if \ i < last() \end{aligned}$$

The query "$/bib/book[1]/title$" will be compiled into $Comp_D[\![\text{"}/bib/book[1]/title\text{"}]\!]$, which is the following λ–term:

$$\begin{aligned} &Comp_D[\![\text{"}/bib/book[1]/title\text{"}]\!] = \\ &\quad = \ \lambda x(Comp_D[\![\text{"}/book[1]/title\text{"}]\!](Comp_D[\![\text{"}/bib\text{"}]\!]) \\ &\quad = \ \lambda x(Comp_D[\![\text{"}/title\text{"}]\!](Comp_D[\![\text{"}/book[1]\text{"}]\!])(/bib) \\ &\quad = \ \lambda x((Comp_D[\![\text{"}/title\text{"}]\!](/book[1]))(/bib) \\ &\quad = \ \lambda x((/title)(/book[1]))(/bib) \\ &\quad = \ \lambda x(/bib/book[1]/title) \end{aligned}$$

Its meaning on the input document $bib.xml$ will be:

$$\begin{aligned} &Sem_D[\![\text{"}/bib/book[1]/title\text{"}]\!](bib.xml) = \\ &\quad = \ Eval_M[\![Comp_D[\![\text{"}/bib/book[1]/title\text{"}]\!]]\!](bib.xml) \\ &\quad = \ Eval_M[\![\lambda x(/bib/book[1]/title)]\!](bib.xml) \\ &\quad = \ \lambda x(/bib/book[1]/title)(bib.xml) \\ &\quad = \{\text{"}TCP/IP \ Illustrated\text{"}\} \end{aligned}$$

5 Conclusion and future work

We have presented the status of our current research focused on using a functional framework for description of XPath 1.0 semantics. We are now in an inceptive phase where we define basic formalisms and check the suitability of the method for the purpose requested.

As outlined above, we expect to extend the scope of this work to XPath 2.0 and later to XQuery 1.0. In general, it means redefinition of the input syntax and formalization of the XML-λ virtual machine. This research should unambiguously lead to a complete work that will be part of the doctoral thesis of the first author of this paper.

Acknowledgement

This work has been supported by the Ministry of Education, Youth and Sports under Research Program No. MSM 6840770014 and also by the grant project of the Czech Grant Agency No. GA201/06/0756.

References

1. S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language, January 2007. http://www.w3.org/TR/xquery/.
2. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (XML) 1.0 (fourth edition), August 2006. http://www.w3.org/TR/2006/REC-xml-20060816.

3. D. Chamberlin, P. Fankhauser, D. Florescu, M. Marchiori, and J. Robie. XML Query Use Cases, March 2007. http://www.w3.org/TR/2007/NOTE-xquery-use-cases-20070323/.
4. J. Clark and S. DeRose. XML Path Language (XPath) 1.0, November 1999. http://www.w3.org/TR/xpath.
5. J. Cowan and R. Tobin. XML information set (second edition), April 2004. http://www.w3.org/TR/2004/REC-xml-infoset-20040204/.
6. P. Genevès and J.-Y. Vion-Dury. XPath formal semantics and beyond: A Coq-based approach. In *TPHOLs '04: Emerging Trends Proceedings of the 17th International Conference on Theorem Proving in Higher Order Logics*, pages 181–198, Salt Lake City, Utah, United States, August 2004. University Of Utah.
7. G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *LICS*, pages 189–202. IEEE Computer Society, 2002.
8. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.
9. P. Hloušek. *XPath, XQuery, XSLT: Formal Approach*. PhD thesis, Dept. of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague, 2005.
10. P. Loupal. Experimental DataBase (ExDB) Project Homepage. http://swing.felk.cvut.cz/˜loupalp.
11. J. Pokorný. XML functionally. In B. C. Desai, Y. Kioki, and M. Toyama, editors, *Proceedings of IDEAS2000*, pages 266–274. IEEE Comp. Society, 2000.
12. J. Pokorný. XML-λ: an extendible framework for manipulating XML data. In *Proceedings of BIS 2002*, pages 160–168, Poznan, 2002.
13. P. Wadler. A formal semantics of patterns in XSLT and XPath. *Markup Lang.*, 2(2):183–202, 2000.
14. P. Wadler. Two Semantics for XPath, January 2000. available online at http://homepages.inf.ed.ac.uk/wadler/papers/xpath-semantics/xpath-semantics.pdf.

XCase - Nástroj pro modelování XML schémat založený na principu MDA

Martin Nečaský, Jakub Klímek, Lukáš Kopenec, Lucie Kučerová, Jakub Malý, and Kateřina Opočenská

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University
martin.necasky@mff.cuni.cz, http://www.necasky.net/xcase

Abstrakt *XML je velmi oblíbeným formátem pro reprezentaci dat. S tím, jak roste množství dat reprezentovaných v XML, roste i důležitost správného návrhu XML schémat těchto reprezentací. Návrh na úrovni jazyků pro popis XML schémat, jako například XML Schema, však přináší řadu problémů. Zdá se, že tyto problémy by bylo možné vyřešit modelováním XML schémat na úrovni konceptuální, což se osvědčilo například i při navrhování relačních databází. V tomto článku popisujeme vznikající nástroj XCase, který je určen pro modelování XML schémat. Pro modelování je využit konceptuální model pro XML data zvaný XSem. Ten aplikuje tzv. modelem řízenou architekturu (MDA), což je obecný přístup k návrhu informačních systémů, a lze ho s výhodami použít i pro modelování XML schémat.*

1 Úvod

XML je dnes běžným formátem pro reprezentaci dat (výměna dat, logický databázový model, ...). S tím, jak poroste množství dat reprezentovaných v XML, bude kladen stále větší důraz na efektivní návrh těchto reprezentací. Bude také nutné, aby se jejich návrh stal součástí celkového procesu návrhu informačního systému. Pod návrhem reprezentací dat ve formátu XML se dnes obecně rozumí návrh XML schémat v nějakém vhodném jazyce, jako např. XML Schema [8]. Tyto jazyky ale neposkytují dostatečnou míru abstrakce od reprezentace dat v XML, která je vyžadována především v počátečních fázích návrhu. Situaci lze přirovnat k návrhu relačních databází, kde také nezačínáme návrhem schémat tabulek, ale pracujeme nejprve na konceptuální úrovni.

Moderním přístupem k návrhu informačních systémů je tzv. modelem řízená architektura (*Model–Driven Architecture*, *MDA*) [6]. MDA vychází z existence několika typů modelů, které jsou používány na různých úrovních návrhu informačního systému, a lze ji přímočaře použít i při datovém modelování, čehož využíváme i v tomto článku. Z tohoto pohledu jsou důležité dva typy modelů: platformově nezávislý model (*Platform–Independent Model*, *PIM*) a platformově specifický model (*Platform–Specific Model*, *PSM*).

PIM je chápán jako konceptuální model. To znamená, že pomocí něj modelujeme data na úrovni, která je abstrahovaná od reprezentace dat v konkrétním datovém modelu jako je např. relační či XML datový model. PSM oproti tomu umožňuje navrhnout, jak jsou data reprezentovaná v konkrétním datovém modelu. Musí zachytit jeho specifika a proto jsou pro různé datové modely aplikovány různé PSM. Můžeme tak například navrhnout PSM diagram popisující reprezentaci dat v relační databázi a jiný PSM diagram popisující reprezentaci v nějakém typu XML dokumentů. Pro relační model dat ale zřejmě potřebujeme jiný PSM než pro popis reprezentace v XML.

V poslední době se objevují snahy o zavedení vhodného modelu pro modelování XML dat, který by byl založen na myšlence MDA. Existující přístupy rozšiřují buď ER model, který je používán především pro konceptuální modelování relačních databází, nebo model UML diagramů tříd. Jak ale ukazujeme v [7], žádný z těchto přístupů není pro modelování XML dat dostatečný. S tím souvisí i současný neuspokojivý stav v oblasti vhodných nástrojů pro takové modelování.

V tomto článku krátce popisujeme nový modelovací nástroj pro XML schémata zvaný *XCase*. Ten implementuje konceptuální model pro XML data zvaný *XSem* [7]. XSem se významně odlišuje od jiných přístupů ke konceptuálnímu modelování XML schémat především důslednou aplikací principu MDA. Nástroj je ve fázi vývoje a jedná se o prototyp, který má za cíl nejen ověřit možnosti modelu XSem, ale i konceptuálního modelování XML schémat obecně, neboť podobný nástroj ještě vyvinut nebyl.

Článek je strukturován následovně. V Sekci 2 popisujeme výhody aplikace principu MDA pro modelování XML schémat. V Sekci 3 stručně popisujeme základní principy modelu XSem. V Sekci 4 popisujeme vyvíjený nástroj XCase. Sekce 5 článek uzavírá.

2 Motivace

V současné době se návrháři při vytváření XML schémat potýkají s celou řadou problémů. Ty jsou způsobeny především tím, že je nutné modelovat každé XML schéma v daném systému samostatně, a to i přesto, že tato schémata mnohdy popisují jen odlišnou reprezentaci souvisejících nebo dokonce stejných dat. Přirozenou cestou pro řešení tohoto problému se zdá být aplikace principu MDA. Od PIM diagramů, které popisují data na úrovni abstrahované od jejich reprezentace v XML, může návrhář poloautomaticky přecházet k PSM diagramům. Ty pokrývají vždy jen určenou část modelované reality a popisují konkrétní XML reprezentaci. Z PSM diagramu je potom mnohem jednodušší vygenerovat výsledné XML schéma, než kdybychom se snažili odvodit jej přímo z PIM diagramu.

Pro praktické využití je nezbytné udržování konzistence mezi jednotlivými úrovněmi, tedy mezi PIM a PSM a ta-

ké mezi PSM a konkrétními odvozenými XML schématy. Tuto konzistenci nelze při správě rozsáhlého souboru XML schémat zajišťovat ručně, a proto je nutné příslušnou podporu zajistit pomocí vhodného softwarového nástroje. Jak ovšem podotýká autor [2], který se zabývá problematikou modelování dat pomocí MDA, a spoluvytváří standard ISO20022 [5], kvalitní nástroje poskytující tento komfort dnes v podstatě ještě neexistují: „*MDA pomáhá udržovat závislosti mezi jednotlivými úrovněmi, alespoň principiálně. Prakticky je to ale stále něco nového, nástroje ještě nejsou dostatečně vyspělé a obousměrné udržování vzájemných závislostí je velmi náročné*" [2].

Zmíněný standard ISO 20022 UNIversal Financial Industry message scheme (UNIFI) poskytuje konkrétní příklad pro reálné využití navrhovaného modelovacího nástroje XCase. UNIFI specifikuje platformu pro vývoj zpráv ve standardizované XML syntaxi. Cílem je vyvinout standard pro výměnu dat mezi finančními institucemi navzájem. Ve standardu ISO UNIFI je základem modelování dat založené na UML aplikovaném jako PIM. Dále pak existuje sada odvozujících pravidel pro přechod od UML diagramů ke XML schématům. Je tedy evidentní, že nástroj s možností tvorby PIM diagramů a odvozováním PSM diagramů jako mezikroků před vygenerováním výsledných XML schémat by zde bylo možno s výhodou použít.

Snahou o standardizaci XML dokumentů ze světa obchodu je také Universal Business Language (UBL) [10]. Cílem je sjednocení XML formátů používaných pro obchodní komunikaci. Pro návrh dat na konceptuální úrovni je opět využíván jazyk UML. K UML diagramům pak existují sady odvozovacích pravidel pro převod do XML schémat. Ani tento převod ale zatím není možné nijak podpořit pomocí vhodných nástrojů, což autorům komplikuje vývoj i aktualizaci knihovny. Jinou možností uplatnění MDA pro modelování XML dat je Health Level 7 [4], ANSI standard pro výměnu medicínských dat. Jednou z jeho standardizovaných syntaxí je také XML. Primárně jsou ovšem data opět modelována nezávisle na reprezentaci, pomocí tzv. Reference Information Model.

Ačkoliv myšlenku využití MDA přístupu pro návrh XML schémat přijala nezávisle již řada pracovních skupin, praktická realizace je bez vhodného nástroje stále poměrně obtížná. Přímé odvozování XML schémat z PIM diagramů je mnohem více náchylné k chybám a také prakticky znemožňuje jakékoliv automatické aktualizace. Pro oblasti, kde se standardy rychle mění, vyvíjejí a dotvářejí, je ovšem automatická aktualizační podpora téměř nezbytná.

3 Konceptuální model XSem

XSem je konceptuální model pro XML data, který aplikuje myšlenku MDA. Jako PIM používá model UML diagramů tříd. Příklad PIM diagramu vytvořeného pomocí editoru XCase je na obrázku 1. Diagram modeluje část domény společnosti, která vyrábí a prodává výrobky svým zákazníkům. Jak je vidět, diagram je nezávislý na reprezentaci dat v XML.

Jako PSM aplikuje model UML diagramů tříd rozšířený o sadu stereotypů, které umožňují zachytit specifika XML. Každý PSM diagram popisuje, jak jsou vybrané komponenty z PIM diagramu reprezentovány v daném typu XML dokumentů. XML dokumenty mají hierarchickou strukturu, která může být navíc nepravidelná. Je také možné mísit strukturovaná a nestrukturovaná (textová) data. Navíc je důležité pořadí v jakém se data v XML dokumentech vyskytují. Všechny tyto speciální vlastnosti XML jsou pokryty stereotypy nabízenými modelem XSem. Nebudeme je zde ale popisovat. Kompletní popis modelu lze nalézt v [7]. Příklad PSM diagramu odvozeného z předchozího PIM diagramu je na obrázku 2. PSM diagram popisuje strukturu XML dokumentů, ve kterých reprezentujeme objednávky výrobků. Z PSM diagramů lze automaticky odvozovat jejich reprezentaci ve vybraném jazyce pro popis XML schémat. V [7] popisujeme převod do reprezentace v jazyce XML Schema.

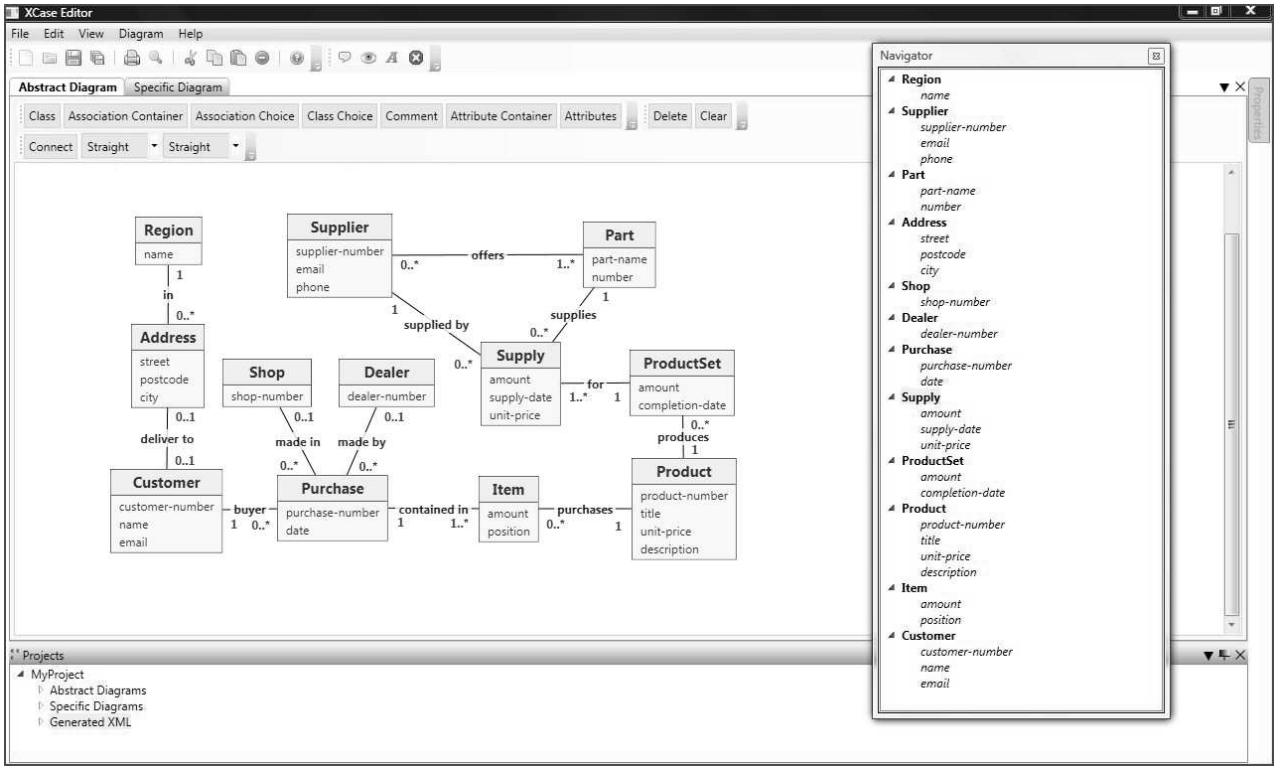
Důležitou vlastností modelu XSem, která jej odlišuje od jiných konceptuálních modelů pro XML data (viz [7]), je formální provázanost mezi PIM a PSM diagramy. To přináší řadu výhod. Pro komponenty XML schématu odvozeného z daného PSM diagramu je možné zjistit jejich sémantiku ve smyslu konceptů modelovaných PIM diagramem. Obráceně je pro daný koncept modelovaný PIM diagramem možné zjistit, v jakých XML schématech a jakým způsobem je reprezentován. To přináší zefektivnění nejen procesu návrhu XML schémat ale také jejich údržby. Výhody tohoto přístupu podrobně popisujeme v [7].

4 Modelovací nástroj XCase

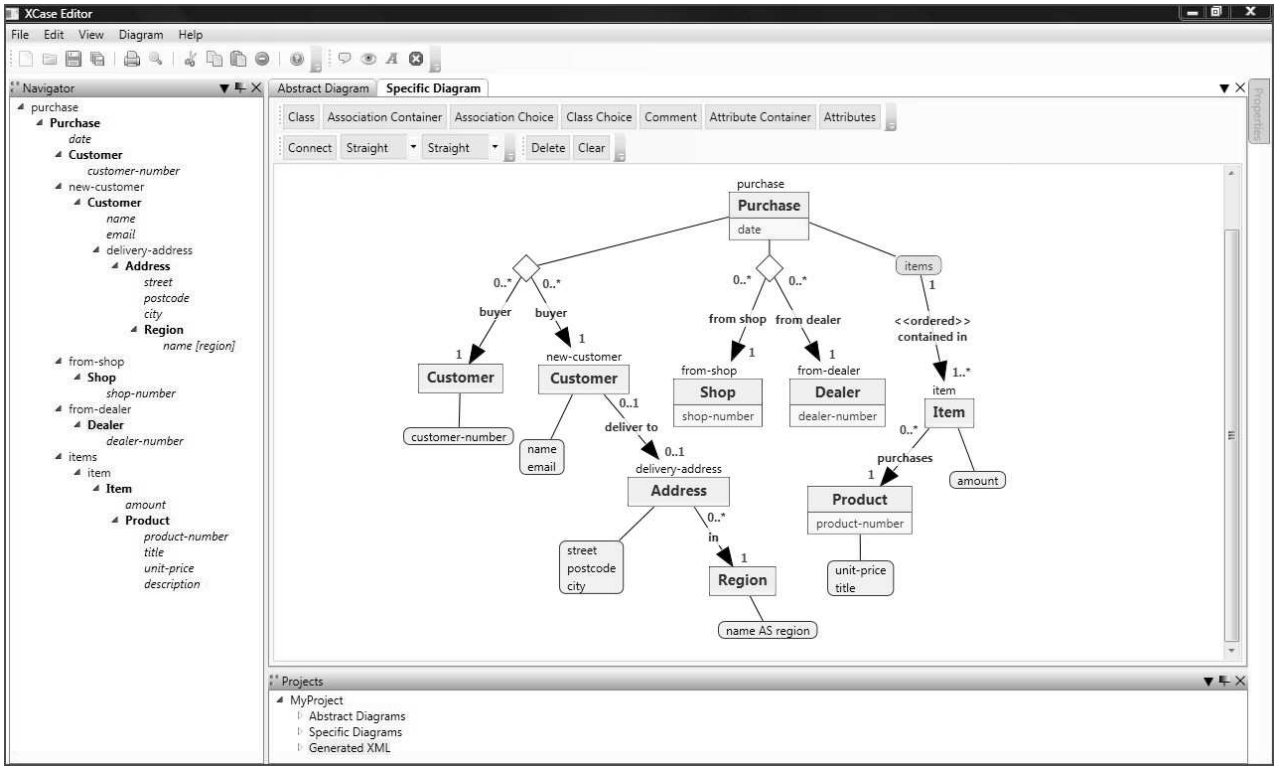
V současné době existuje několik softwarových nástrojů pro práci s XML schématy. Na jedné straně jsou to programy umožňující vizualizaci schémat jako Altova XML Spy [9]. Jejich přínosem je snazší a přehlednější editace schémat, než by bylo možné přímo v XML formátu. Navíc tyto nástroje typicky obsahují editory dalších XML technologií a integrují je tak v jednom nástroji.

Na straně druhé jsou to UML editory jako například Enterprise Architect [3]. Ty částečně aplikují MDA a z PIM diagramů umí odvozovat PSM diagramy popisující reprezentaci modelovaných dat v XML. Problém je, že odvozování probíhá automaticky a návrhář má pouze omezené možnosti, jak výsledný PSM diagram upravovat. Proces odvození ale nemůže být automatický, protože návrhář může vyžadovat stejná data reprezentovaná v různých typech XML dokumentů s rozdílnou strukturou.

Nástroj XCase se snaží vyřešit problémy existujících nástrojů aplikací modelu XSem, jehož hlavní principy byly nastíněny v předchozí sekci. Mezi hlavní funkce programu patří:



Obr. 1. PIM diagram.



Obr. 2. PSM diagram.

– vytváření a editace PIM a PSM diagramů a jejich import/export z/do XMI[1]
– poloautomatické odvození PSM diagramů z PIM diagramů
– udržování vazeb mezi PIM a PSM diagramy, udržování jejich konzistence
– automatický překlad PSM diagramů do jazyků pro popis XML schémat jako např. XML Schema

Editace diagramů probíhá v integrovaném UML editoru omezeném na práci s diagramy tříd. A rchitektonicky je UML editor založen na vzoru MVC[2]. Pro reprezentaci modelu je použita knihovna nUML [1], která implementuje jazyk UML ve verzi 2.0. Rozšíření UML do XSem je řešeno pomocí stereotypů a jejich atributů, tj. pouze pomocí standardních prostředků UML. To umožňuje používat knihovnu nUML bez dalších zásahů a zachovává i možnost přenosu PSM diagramů do jiných SW nástrojů prostřednictvím XMI.

Diagramy jsou organizovány do projektů. Každý projekt v sobě sdružuje jedno nebo více PIM diagramů, odvozené PSM diagramy a odpovídající XML schémata. Software zajišťuje udržení vazeb mezi závislými diagramy. Směrem ze shora dolů (od PIM) je propagace změn okamžitá, opačným směrem je ve většině případů návrhář nejprve dotázán, zda chce provedenou úpravu propagovat. Důvodem je, že zcela běžně je žádoucí, aby PSM diagramy neobsahovaly všechny atributy, třídy nebo asociace obsažené v PIM diagramu a naopak obsahovaly elementy, které v PIM diagramech nejsou.

Uživateli je třeba především co nejvíce usnadnit přechod od PIM k PSM diagramům a následnou práci s nimi. Přechod ale není možné automatizovat, pouze podporovat – struktura jednotlivých typů XML dokumentů vychází z uživatelských požadavků a může být dokonce nutné odvodit několik různých PSM diagramů ze stejné části PIM diagramu. Odvozování PSM diagramů probíhá za pomoci grafických nástrojů. Uživatel nejprve označí komponenty z PIM diagramu, které chce mít ve výsledném PSM diagramu. Editor vytvoří PSM diagram modelující základní XML reprezentaci vybraných komponent. Přidanou hodnotou editoru je možnost dále tento počáteční PSM diagram libovolně upravovat a navrhnout tak XML reprezentaci přesně podle uživatelských požadavků.

5 Závěr

V článku jsme představili konceptuální model pro XML data zvaný XSem. Představili jsme probíhající projekt implementující modelovací nástroj XCase. Ten umožní modelování XML dat na konceptuální úrovni pomocí modelu XSem. Nástroj má především ověřit možnosti v oblasti, neboť podobný nástroj ještě nebyl implementován. Aktuální volně dostupná verze je přístupná na adrese *http://www.necasky.net/xcase*.

[1] XMI (XML Metadata Interchange) je standard pro výměnu metadat a je využíván pro přenos UML diagramů mezi různými softwarovými nástroji.

[2] Model–View–Controller (MVC) je návrhový vzor oddělující model, aplikační logiku a uživatelské rozhraní.

Poděkování

Tento článek byl částečně podpořen Národním programem pro výzkum (projekt informační společnosti 1ET100300419).

Reference

1. R. Campero. *nUML library*. `http://numl.sourceforge.net`.
2. A. B. Coates. *Semantic Data Models and Business Context Modelling*. Talk at XML 2007. `http://2007.xmlconference.org/public/schedule/detail/359`.
3. *Enterprise Architect*. Sparx Systems. `http://www.sparxsystems.com/`.
4. *HL7* `Health Level 7, http://www.hl7.org/`
5. *ISO 20022 UNIversal Financial Industry message scheme (UNIFI)*. `http://www.iso20022.org/`.
6. J. Miller and J. Mukerji. *MDA Guide Version 1.0.1*. Object Management Group, 2003. `http://www.omg.org/docs/omg/03-06-01.pdf.`
7. M. Necasky. *Conceptual Modeling for XML*. Ph.D. thesis, 2008. Department of Software Engineering, Charles University. `http://www.necasky.net/thesis.pdf.`
8. H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. *XML Schema Part 1: Structures (Second Edition)*. W3C, October 2004. `http://www.w3.org/TR/xmlschema-1/`.
9. *XML Spy*. Altova. `http://www.altova.com/`.
10. *OASIS Universal Business Language (UBL)* `www.oasis-open.org/committees/ubl/`

A tool for simulation of the evolution

Marián Novotný

Institute of Computer Science, Pavol Jozef Šafárik University, Jesenná 5, 041 54 Košice, Slovakia
marian.novotny@upjs.sk

Abstract. *In this paper we briefly describe the program for simulation of Darwinian evolution by Richard Dawkins. Biomorph is a 2D-shape, which is defined by its parameters: genes. The visualization of a biomorph from genes is given deterministically by the program. We generalize this idea by using PDOL-systems [8] for constructing a language for defining biomorphs. We design and implement a tool for a simulation of the evolution and show some experimental results.*

1 Introduction

One of the most important result of *Biology*: *the evolution theory* [3] is a very interesting theory for Biology and also for other subjects including *Computer Science*. Many applications can be found in this area e.g. *evolution algorithms*, *genetic algorithms* etc. Deeper understanding of principles of the evolution is necessary for solving certain problems in various domains.

There are three major mechanisms driving evolution [5]. The first is *natural selection*, which is a process causing heritable traits that are helpful for survival and reproduction to become more common in a population, and harmful traits to become more rare. This occurs because individuals with advantageous traits are more likely to reproduce successfully, so that more individuals in the next generation inherit these traits.

In contrast, *genetic drift* produces *random* changes in the frequency of traits in a population. Genetic drift results from the role chance plays in whether a given individual will survive and reproduce. Though the changes produced in any one generation by drift and selection are small, differences accumulate with each subsequent generation and can, over time, cause substantial changes in the organisms.

Gene flow is the exchange of genes between populations, which are usually of the same species. Examples of gene flow within a species include the migration and then breeding of organisms, or the exchange of pollen.

We focus mainly on an abstraction of these mechanisms: the principle of *cumulative selection* [5] as a fundamentally *nonrandom* process, which performs gradual step by step transformation, where each improvement is used as a basis for a future building.

2 Dawkins biomorphs

Richard Dawkins in his book [5] described the program for simulation of the evolution of *biomorphs*. Biomorph is a 2D-shape, which is defined by its *parameters: genes*. The visualization of a biomorph from genes is given *deterministically* by the program.

In each step all possible direct descendants of the actual biomorph are displayed on the screen. User selects one of them as *the actual biomorph* for the next step in the evolution. This process of cumulative selection can be shown on Fig.1.

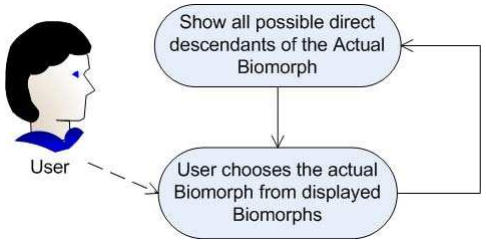


Fig. 1. The process of cummulative selection.

2.1 Parameters of biomorphs

Biomorph has a form of *a binary tree* - structure with recursive definition of tree-growing. This approach used Dawkins [5] as the analogy to the embryonic development of plants and animals in general. The binary tree is exactly defined by its parameters.

Values of each parameter are *uniformly* distributed along certain interval. For example the common parameter - *the depth of recursion* might be taken from the set of natural numbers from 1 to 15. Some parameters of Dawkins biomorphs are illustrated on Fig.2.

More formally we say that each parameter p has *minimal* value min_p, *maximal* value max_p, *the number of values* n_p and then *the difference* is $\Delta = \frac{max_p - min_p}{n_p - 1}$. Each parameter p can be taken from the set V_p,

$$V_p = \{min_p, min_p + \Delta, min_p + 2.\Delta, \ldots, max_p\}. \quad (1)$$

If we have k parameters, then the total numbers of biomorphs is $\prod_{i=1}^{k} n_i$. If we would like to find some particular biomorph, it seems to be hard to explore all possibilities, but the cumulative selection of the evolution process will be helpful.

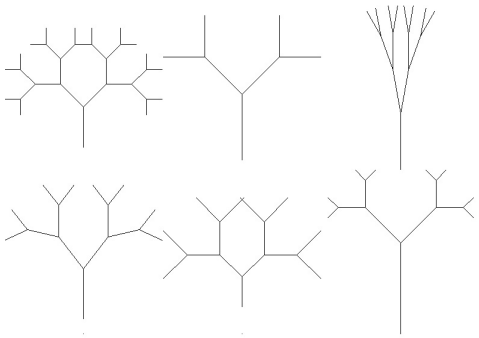


Fig. 2. Some parameters of Dawkins Biomorphs.

3 DOL systems

In the program by Richard Dawkins [5] it is not possible to define neither a parameter, a type, possible values of parameters, nor an interpretation of parameters. We would like to extend this program with this possibility. We consider *L-systems* [9] as a suitable *formal language* for describing growing structures like plants and the process of growing of organism.

The central concept of L-systems is that of *rewriting*. In general, rewriting is a technique for defining objects by replacing parts of *an initial object* using a set of *rewriting rules*. The theoretical computer science exhaustively studies rewriting systems which operate on character strings and the center of attention is focused on sets of strings called *formal languages*.

In 1968 a biologist Aristid Lindenmayer introduced a new type of string-rewriting mechanism called L-systems. The essential difference between *Chomsky grammars* and L-systems lies in the method of applying of rewriting rules. In Chomsky grammars there are applied *sequentially*, whereas in L-systems are applied in *parallel* and simultaneously replace all letters. This property reflects biological motivation of L-system to describe growing of organisms.

3.1 PDOL-systems

We use a simple class of L-systems, those which are *deterministic*, *context-free*, called *DOL-systems* [9]. Mo-reover the letters in PDOL-systems [8] are extended by *associated parameters*, which belongs to the set of real numbers $\mathcal{R}$.

More formally the letters belong to an *alphabet* V. A module with letter A and parameters $a_1, \ldots, a_n$ is denoted by $A(a_1, \ldots, a_n)$. Every module belongs to the set $M = V \times \mathcal{R}^*$, where $\mathcal{R}^*$ is the set of all finite sequences of parameters.

In rewriting rules[1] we need to have the set of *variables* Σ and over this set we denote the set of all corrected *arithmetic expressions* $\mathcal{E}(\Sigma)$ and the set of *logical expressions* $\mathcal{C}(\Sigma)$. An arithmetic expression consists of variables from Σ or real constants, which might be combined using *operators* such as $+, -, *, /$ etc. Logical expressions are created from arithmetical using the *relational operators* $<, >, =$ and also are combined using the *logical operators* $!, \&\&, ||$.

A PDOL-system is defined as an ordered quadruple $G = \langle V, \Sigma, \omega, P \rangle$, where

- V is the alphabet,
- Σ is the set of variables,
- $\omega \in (V \times \mathcal{R}^*)^+$ is nonempty parametric word called *the axiom*,
- $P \subseteq (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma)^*)^*$ is a finite set of rewriting rules.

We can write a rewriting rule in format:

$$predecessor : condition \rightarrow succesor \qquad (2)$$

If the *condition* is true implicitly, for example if a predecessor has no parameters, we can omit it altogether. A rewriting rule can be *applied* in a module in a parametric word if the following conditions are met:

- The letter in the module and the letter in the predecessor of the rule are the same,
- the number of actual parameters is equal to the number of variables in the rule,
- the condition of the rule *evaluates* to true, if the actual parameters from the module are *substituted* for variables in the rule.

When no rewriting rule matches a module in the re-writing string, we replace the module itself. We say that a parametric system is *deterministic* if and only if for each module is in the set of rewriting rules included *at most one* which matches the module.

*Example 1 (*A simple PDOL-system G*).*

Let $G = \langle V, \Sigma, \omega, P \rangle$, where the alphabet $V = \{F, X\}$ and ω=,(32768)X and the rules are:

1. F(t)→ F(1,5*t)
2. X→F(1)-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X

The three steps derivation from axiom ω in PDOL-system G is as follows:

$\omega \Rightarrow$,(32768)X⇒

,(32768)F(1)-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X⇒

,(32768)F(1,5)-(23)[[F(1)-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X]
+(23)F(1)-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X]+(23)F(1,5) [
+(23)F(1,5)F(1)-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X]-(23)F(1)
-(23)[[X]+(23)X]+(23)F(1)[+(23)F(1)X]-(23)X⇒ ...

3.2 The geometric interpretation of PDOL-systems

Strings generated by PDOL systems may be interpreted *geometrically* in many different ways. We use *the*

[1] productions in [9]

turtle [1] interpretation, introduced by [10] and extended by Prusinkiewicz et all [8, 9] with some minor modifications.

The turtle has *the state parameters* as a color, a depth of the pen, a position and an orientation. Basic commands for the turtle are those for moving and changing its parameters. The turtle has also a *stack* for storing its state during branching.

Before interpreting a word we have the turtle with implicit parameters in a certain initial position. After the string has been generated by applying rewriting rules, it is *scanned sequentially* from left to right and

- if a module is a command for the turtle, then this command is executed by the turtle in the 2D shape,
- else we skip this module and go to the next one.

After producing whole word the geometrical interpretation of the word has been drawn by the turtle. In the Tab.1 we enumerate the modules which we will be used in our program and briefly describe their meanings.

Module	Interpretation
$+(\varphi)$	Turn left by angle φ.
$-(\varphi)$	Turn right by angle φ.
|	Turn 180°.
[	Push the current state of the turtle into the stack.
]	Pop a state from the stack and change the current state with its.
{	Start saving the list of positions of the turtle as the vertices of a polygon.
}	Fill the saved polygon.
$f(l)$	Move forward a step of length l without drawing a line.
$F(l)$	Move forward a step of length l with drawing a line.
$G(l)$	Similar to F(l), but it does not save the position as the vertex for a polygon.
.	Saving the position for a polygon.
$\#(w)$	Set the line width to w.
$,(i)$	Set the index of the colormap to i.

Tab. 1. Letters and theirs interpretations by the turtle.

4 A program for simulation of the evolution

We designed and implemented the program Evolution, which has to be an *user-friendly* tool for simulation of the darwinian evolution. In this section we concisely describe the functionality of this tool. More details can be found in the user's manual [6].

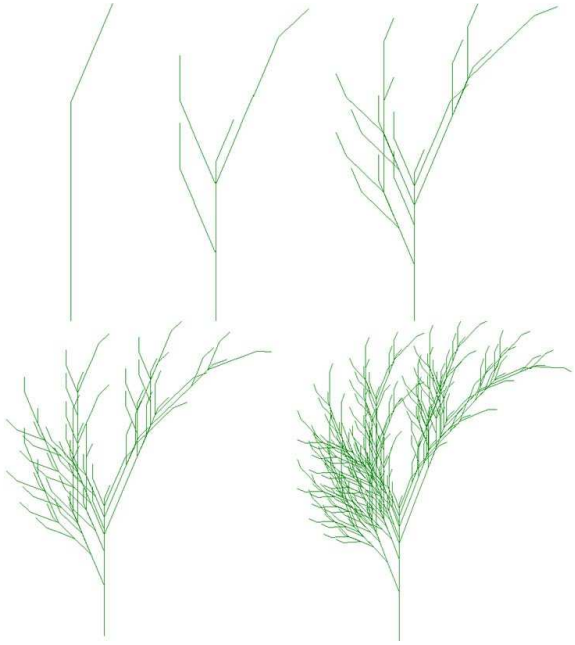


Fig. 3. The geometric interpretation of the PDOL-system G from example 1.

4.1 Language

We created a simple language for defining parameters and biomorph. The syntax of this language is defined and described in the user's manual [6]. We show in example 2 a simple commented source code of this language.

In the beginning of a code user has to define maximal number of iterations during derivation and has to set an initial value of the parameter of iteration. In the next part he defines uniform parameters with corresponding minimal, maximal, initial values and the number of values. We use variables in a rewriting rule with special symbol "$".

The source code continues in defining a starting word S^2 and rewriting rules in which are arithmetical and logical expressions in notation similar to the *lisp lists*.

*Example 2 (*A source code of a evolution*).*

```
Parameters
  Iterations
  //(max Value; initial Value)
    (13;3)
  Uniform
  //(variable;min Value;max Value; the numbers;initial)
    (u;10;50;50;45)
    (k;0,5;2;50;0,8)
    (n;1;9;8;1)
Rewriting rules
  S:P<360>
  P<x>:(> x 0)->[T<1>]+<(/ 360 $n)>P<(- 360(/ 360 $n))>
  T<x>:TRUE->F<x>[-<$u>T<(* x $k)>]+<$u>T<(* x $k)>
```

[2] axiom ω in the definition of PDOL-systems

4.2 The Program evolution

The program evolution consists of *the editor* and *the main window* of the application. After typing a code of a evolution in the editor, the evolution is created into main window. Thus the actual biomorph is a word, which is generated by PDOL system with initial parameters.

The main screen is divided into two times the number of parameters parts with equal size. In each one is a geometrical interpretation from all direct descendants of the actual biomorph. User can make a evolutionary selection as is shown on Fig.1.

The program supports different screen resolutions and allows saving pictures of biomorphs. It is possible to display a biomorph into the main window and show its word and parameters. The application records the history of the evolution and it can be displayed all ancestors of the actual biomorph in the chronological order.

4.3 Experimental results

In the tool Evolution we implemented the former Daw-kins biomorphs according to book [5]. Besides that, we built another structures similar to the former biomorphs. Some of them can be shown on Fig.4. We also created a database of *fractals* from the book [9] and some structures of trees and plants from [8], in which it is possible to finding suitable parameters using the process of cumulative selection of the evolution.

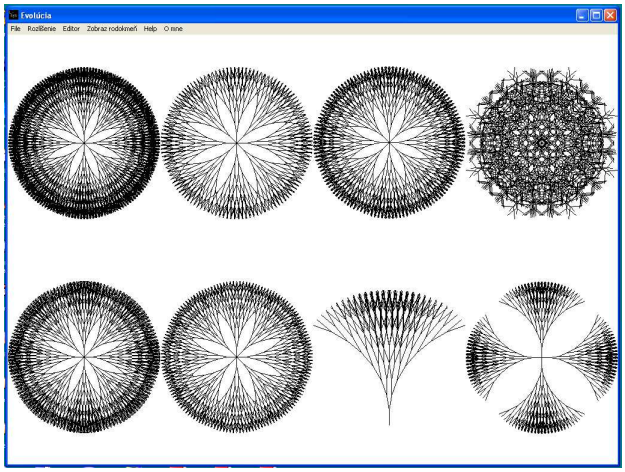


Fig. 4. The program Evolution.

5 Conclusions

In our approach we generalized the idea by Richard Dawkins by using PDOL-systems [8] for constructing a language for defining biomorphs. We designed and implemented a tool for a simulation of the evolution and showed some experimental results. The program Evolution can be used

- similar to the program by Richard Dawkins for better understanding of the principle of cumulative selection in the evolution,
- as a didactic tool in teaching of basic course of formal languages,
- for finding constants for geometrical structures, which are represented by PDOL-systems,
- for constructing fractals. In this case we have only one parameter – the number of iterations. This usability of the program is shown on Fig. 5. The relation between iterated function systems [2] and PDOL-systems is published in the paper [7]. A lot of examples of fractals represented by PDOL-systems is available in the book [9].

We would like to extend our tool by defining a type for parameters. The current version of the program allows a change of a parameter only by uniformly going trough the line. The other functions might be useful and *an enumerable type* seems to be suitable at least for changing of the index of a color from a defined list of color indices.

More realistic images can be possibly produced by a probabilistic interpretation of PDOL-systems by turtle. By this we mean that realization of commands of the probabilistic turtle will be randomized under certain distribution.

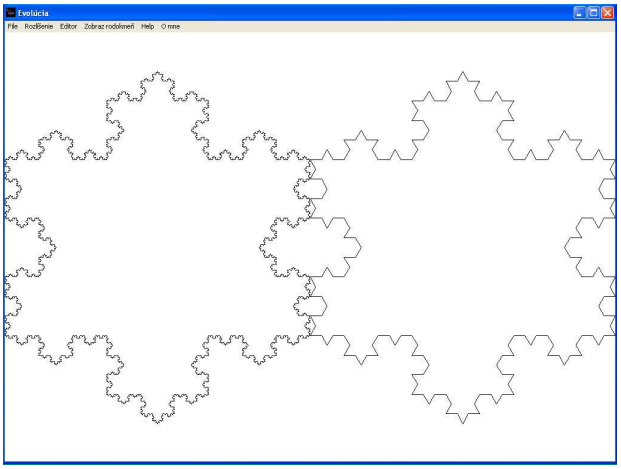


Fig. 5. The Koch snowflake.

References

1. Abelson, H. DiSessa, A. Turtle Geometry: The Computer as a Medium for Exploring Mathematics.MIT Press. 1981

2. Barnsley, M. Demko, S. Iterated function systems and the global construction of fractals. Proc. Roy. Soc. London, Ser A, 399. 1985
3. Darwin, C. On the Origin of Species by means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.Penguin Classics. 1982
4. Dawkins, R. The Selfish Gene: 30th Anniversary Edition. Oxford University Press. 2006
5. Dawkins, R. The Blind Watchmaker. New York: W. W. Norton & Company, Inc. 1996
6. Novotný, M. The program Evolution. user's manual. unpublished(in Slovak). 2008
7. Prusinkiewicz, P. and Hammel, M. Language restricted iterated function systems, Koch constructions and L-systems. In Hart, J. C., ed., New Directions for Fractal Modeling in Computer Graphics. SIGGRAPH '94 Course Notes, ACM Press. 1994
8. Prusinkiewicz, P. Hammel, M. Hanan, J. and Mech, R. L-systems: from the theory to visual models of plants. In M. T. Michalewicz, editor, Proceedings of the 2nd CSIRO Symposium on Computational Challanges in Life Sciences. CSIRO Publishing. 1996
9. Prusinkiewicz, P. and Lindenmayer, A. The Algorithmic Beauty of Plants. Springer-Verlag New York, Inc. 1996
10. Szilard, A. Quinton, R. An interpretation for DOL systems by computer graphics - The Science Terrapin. 1979

Daly by se použít robotické metody i v sémantickém webu?

David Obdržálek
Katedra softwarového inženýrství, Matematicko-fyzikální fakulta UK
Malostranské náměstí 25, 118 00 Praha 1
david.obdrzalek@mff.cuni.cz

Abstrakt. *V automatizovaném procesu získávání a zpracování informaci je používáno množství různých metod. V této práci se pokusíme naznačit, zda by tu mohly být využity také postupy a metody z domény reálné robotiky. Naším cílem bude prohlédnout některé základní běžně používané metody, které pomáhají reálným robotům v jejich práci, a zjistit zda existují paralely, které by pomohly tyto metody využít i pro virtuální roboty či agenty ve virtuálním světě informací. Typickým příkladem využití takových metod by mohla být například navigace v dokumentech propojených hyperlinky v prostředí internetu, anebo orientace v datech a metadatech používaných či navrhovaných k použití v prostředí Sémantického webu.*

1 Úvod

Jakkoli jsou robotika a obzvláště mobilní robotika ve světě vědy mladé obory, jeví se oproti oblasti sémantického webu jako obory již dávno zavedené a jsou také i podstatně více rozvinuté. Existuje již celá řada jak teoretických výsledků, tak prakticky ověřených postupů, které robotům umožňují (a usnadňují) plnit jejich práci v reálném světě. Domníváme se, že v oblasti sémantického webu je nejen možné nalézt teoretické použití pro některé z těchto dnes již běžně používaných robotických metod řešících navigaci, lokalizaci a mapování, ale že je možné je na rozdíl od jiných nápadů týkajících se přenesení teoretických postupů z jednoho oboru do druhého (viz např. [7]) v praxi i realizovat. Náš článek naznačí několik základních robotických algoritmů či postupů a pokusí se podnítit možné směry jejich využití pro potřeby sémantického webu.

V robotice se dnes zcela běžně a úspěšně používají metody pro navigaci, lokalizaci a mapování prostředí. Pro obecné zpracování informací (a v sémantickém webu zvláště) je orientace v dokumentech a ve vazbách mezi nimi také jednou ze základních potřeb. Proto se přímo nabízí myšlenka využít robotické metody pro orientaci jako metody pro obecné zpracování informací. Reální roboti v reálném světě získávají znalosti o svém okolí prostřednictvím množství různých typů senzorů, na základě těchto dat pak mohou například postupně budovat svou „mapu světa“, určovat svou polohu v takové mapě anebo dokonce oboje zároveň. Pokud bychom tedy uměli nalézt pro virtuální roboty vhodnou interpretaci vlastností prostředí ve kterém se pohybují a vhodně nadefinovat jejich senzoriku, pak by mělo být možné víceméně přímou implementací využít tyto reálné algoritmy i ve virtuálním prostředí. Vzhledem k tomu, že již existuje řada navigačních a lokalizačních algoritmů, jejichž úspěšnost je reálně prověřena, mohlo by to znamenat zajímavý impuls pro zpracování dat v sémantickém webu.

2 Základní náměty

Jednou z nutných podmínek pro tvorbu úspěšných mobilních robotů je kvalitní implementace jejich orientace v prostředí, kde se mají pohybovat. V tomto směru robotika řeší několik základních úloh:

- Orientace ve známém prostředí
- Poznávání neznámého prostředí
- Plánování postupu pro dosažení cíle

V prvním typu úloh má robot znalost o prostředí, ve kterém je umístěn a kde se pohybuje. V rámci svého pohybu se snaží co nejpřesněji určit, kde se v daném prostředí nachází – lokalizuje se. Ve druhém typu úloh se robot pohybuje v neznámém prostředí a postupně toto prostředí poznává – mapuje. Třetí typ úloh se zabývá problémem, jak dosáhnout určeného cíle, ať už je předem známý cíl konkrétní nebo abstraktní – plánování.

První dva typy úloh jsou spolu velmi často provázány – robot se může pohybovat v neznámém prostředí, vytvářet jeho mapu a zároveň se podle této mapy také i orientovat. Podle toho, jak jsou obě metody (orientace i poznávání) přesné, vzniká méně či více přesná mapa prostředí a robot má méně či více přesnou informaci o tom, kde se nachází, přičemž chyby v jedné části se promítají do části druhé. Tuto problematiku řeší úlohy nazývané SLAM (Simultaneous Localization and Mapping – simultánní lokalizace a mapování, viz dále, příp.[1], [4]).

Plánování dalšího postupu robota je velmi silně závislé na tom, jak je stanoven cíl (cíle) a jaké jsou „povolené“ prostředky k jeho (jejich) dosažení. Na příkladu průchodu bludištěm můžeme ukázat některé možnosti: Cíl může být stanoven například jako dosažení konkrétní lokace („najdi východ z bludiště“), splnění složitější podmínky závislé na konkrétním prostředí („najdi nejkratší cestu mezi vchodem a východem z bludiště“) anebo i jako splnění nějaké lokačně ne nutně určené nebo pevné podmínky („najdi v bludišti pohyblivý cíl“, „v bludišti s predátory přežij co nejdéle“ apod.). Cíl může být předem daný s možností naplánování cesty, stejně jako může být předem blíže neurčený s tím, že jeho dosažení zjistíme teprve až jej dosáhneme, a tedy není možné cestu předem přesně naplánovat.

3 Senzory

V robotice je dostupná a používá se velká šíře typů senzorů. Složitost poskytovaných dat je také velmi různá, od jednoduchých logických údajů ano/ne (např. nárazníky rozlišující pouze stavy narazil / nenarazil), přes diskrétní číselné údaje (např. počítadlo otáček), analogové údaje

(např. senzor měřící jasovou úroveň osvětlení, senzor měřící vzdálenost od překážky), až po komplexní údaje, které byly před předáním k dalšímu zpracování již nějakým způsobem předzpracovány (např. spektrální analýza obrazu). To vše je možné získat jako jedinou hodnotu nebo i s časosběrnými údaji, v jednom, dvou i třech prostorových rozměrech a v ne-prostorových doménách pokrývajících všechny lidské smysly (a nejen je).

Pro další zpracování jsou výstupy senzorů obvykle převáděny do digitální podoby. Algoritmy, které je následně používají, už pak používají číselné hodnoty a nepracují s původními skutečně zkoumanými vlastnostmi okolí. Proto při využití těchto algoritmů ve virtuálním prostředí sémantického webu nový problém týkající se adaptace vstupů pro tyto algoritmy nevznikne.

4 Zpracování dat pro tvorbu mapy

Při procesu zpracování vstupních dat pro účely mapování je třeba v těchto datech umět vyhledávat podstatné informace a odlišit je od informací sice jinak třeba zajímavých, ale pro tvorbu map nepodstatných. V dalším textu budeme hovořit o zpracování obrazu; v tomto procesu chápeme obraz nikoli jen jako informaci o barvě jednotlivých bodů scény, ale ve zobecněném smyslu jako informaci, kterou poskytují vstupní senzory o bezprostředním okolí robota, tj. to, co robot „vidí“ z místa, kde se právě nachází. Mohou to tedy být jak klasická obrazová data z barevných senzorů (např. z kamery či fotoaparátu), stejně tak to ale mohou být data získaná pomocí laserového dálkoměru anebo ultrazvukového čidla, která představují vzdálenost k překážce „viditelné“ pomocí zvolené technologie. Z takových vstupů potom získáme množinu dat, kterou si lze představit jako obraz, jehož jednotlivé body tvoří místo obvyklé barvy pozorovaného bodu například údaj o vzdálenosti k nejbližší překážce od pozorovatele v tomto zvoleném směru.

Při zpracování obrazu jsou při tvorbě mapy nebo při lokalizaci pomocí již existující mapy vyhledávány takové jeho části, které definují možnosti pohybu robota. Základní možností je vyhledávání důležitých hran a ploch, stále častěji se ale používá komplexnější vyhledávání takzvaných artefaktů či „landmarků“ a tyto objekty jsou pak použity jako další důležitý zdroj informací o scéně. Hrany jako takové slouží při zpracování obvykle jako stavební prvky pro konstrukci celé scény, přičemž jsou vybírány takové vstupní údaje, kde jsou hrany pro tvorbu mapy důležité. Například pro generování prostorové mapy budou podstatné hrany ve vzdálenostním typu obrazu, zatímco barevné informace by mohly být zcela pominuty. Se znalostí prostředí bychom ale i z barevných hran mohli umět získat relevantní data. Například ostrá změna jasu nebo odstínu spolu s informací, že se pohybujeme v prostředí s velkými jednobarevnými plochami bude možná znamenat zlom plochy a tedy informaci podstatnou pro tvorbu mapy.

Artefakty budou v mapě reprezentovány objekty, které kromě svých fyzických vlastností (rozměry, barva, povrch a podobně) nesou i dodatečnou informací týkající se jejich typu a atributů konkrétního typu (dveře, stůl, schod a podobně). Mapa vytvořená s pomocí artefaktů pak může přinášet kromě své základní vlastnosti – popisu prostředí i právě takovéto důležité dodatečné informace.

Z hlediska sémantického webu odpovídá tvorba mapy zjišťování vnitřní struktury dokumentu a/nebo zjišťování vztahů mezi provázanými dokumenty. V obou případech půjde pravděpodobně především o zjištění topologické mapy, v případě vnitřní struktury dokumentu s konkrétní grafickou reprezentací by mohlo být užitečné tento dokument popsat i mapou geometricky přesnou.

5 Lokalizace

V základní lokalizační úloze je k dispozici mapa prostředí, ve kterém se robot má pohybovat. Na začátku svého pohybu robot může (ale nemusí) mít informaci o své poloze. Pokud ji má, pak se ji bude při pohybu stále snažit udržovat a korigovat podle svého pohybu a dat z jeho senzorů. Pokud informaci o své poloze nemá, pak se ji bude snažit opět na základě mapy a svých vstupních dat zjistit. Taková situace odpovídá vysazení robota na neznámém místě, anebo ztráta orientace přenesením na jiné místo, aniž by to robot svými senzory mohl postihnout (tzv. „kidnap problem“).

Pokud je známa počáteční poloha robota a robot se pohybuje, bude jeho aktuální pozice zjišťována na základě těchto čtyř vstupů: předchozí pozice, představa o vlastním pohybu z předchozí pozice, vstupy ze senzorů, mapa prostředí. Skutečný pohyb robota se ale může od předpokládaného pohybu lišit kvůli vlastní mechanice pohybu (například vlivem proklouznutí kol) nebo kvůli vlastnostem a reakci vnějšího prostředí (například posunutí robota jinou pohybující se entitou). Proto i s přesnou znalostí předchozí pozice není možné jen na základě vlastní představy o pohybu přesně určit aktuální pozici, navíc jsou i vstupy ze senzorů zatíženy chybami. Proto je jedním z důležitých problémů v robotice vyrovnání se s nepřesnostmi všech vstupů a s nesouladem mezi představou a skutečností. Kromě vlastního zpřesňování dat poskytovaných senzory se používají například i metody založené na statistických postupech. Jednou z dnes velmi populárních – a velmi úspěšných – tříd takových metod jsou tzv. „metody Monte Carlo lokalizace“ (MCL metody, např. viz [3],[8]).

Robotická lokalizace by v sémantickém webu mohla mít použití pro dohledání konkrétního dokumentu (či jeho části) v existujícím datovém skladu na základě popsání jeho vlastností. Zadaný popis by pak byl analogií aktuálnímu zjištění, co robot po vysazení na neznámé místo vidí kolem sebe. Zdá se však, že v této oblasti budou mít robotické algoritmy spíše minoritní použití vzhledem k poměrně silně propracovaným metodám pro dolování dat. Přesto by mohlo být zajímavé využít právě představu, že virtuální robot je vysazen na neznámé místo, kde kolem sebe něco vidí, tato informace však není dostatečná pro přesné určení jeho polohy (tj. nalezení konkrétního dokumentu nebo jeho části). Proto se virtuální robot začne pohybovat a zpřesňuje svou pozici na základě porovnání toho, co má vidět (vyhledávané údaje) s tím, co skutečně vidí (obsah dostupných dokumentů). Tento postup by mohl poskytnout výsledky i v případě, že uživatelovo zadání nebylo přesné

nebo bylo zčásti chybné, protože výsledkem Monte Carlo lokalizace není konkrétní jedna lokace ale pravděpodobnostní údaj o možném umístění.

6 Mapování

Pro tuto chvíli předpokládejme, že nemáme k dispozici mapu prostředí a jedním z úkolů robota je takovou mapu vytvořit.

Při budování mapy velmi pomáhá znalost typu prostředí, jakkoli to je ve zcela obecném případě nedosažitelný požadavek. Se znalostí alespoň přibližných vlastností ale můžeme tvořit mapu snadněji či přesněji – například informace, že se robot pohybuje v bludišti ve čtvercové síti napomůže výrazně jednoduššímu postupu pro tvorbu takové mapy. Pokud se robot vždy pohybuje ve vnitřním prostředí, budou například i podmínky osvětlení definovány lépe, než pro venkovní roboty. Znalost prostředí umožní lépe zpracovávat data a pomůže při odstraňování chyb měření, ať už při pořizování vstupních dat pro účely tvorby mapy nebo dat z měření pohybu robota.

Mapa, kterou robot tvoří nebo podle které se orientuje, může být různých typů. Obvykle se tvoří mapa topologická nebo mapa geometricky přesná. V topologické mapě jsou důležité především jednotlivé uzly, jejich vlastnosti a vazby mezi nimi, v geometricky přesné mapě jsou důležitá absolutní umístění jednotlivých elementů, ale informace o objektech naopak často nejsou vůbec zachyceny. Zobecněná geometricky přesná mapa samozřejmě může obsahovat i informace o objektech, které pak mohou být z hlediska lokalizace i plánování velmi závažné.

Tvorba mapy je v robotice velmi obvyklým tématem a existuje řada implementací pro různé účely. Mnohé z implementací uvažují i dynamicky se měnící prostředí (viz např. [9]), což je vhodné uvažovat i v prostředí sémantického webu. Analogii mapy reálného prostředí ve virtuálním prostoru blíže zmíníme v kapitole 8.

7 SLAM – Simultánní lokalizace a mapování

Problém současné lokalizace robota v nějakém prostředí a zároveň tvorby mapy tohoto prostředí je v mobilní robotice velmi aktuální, neboť přesná mapa prostředí je zřídka předem k dispozici a navíc prostředí je často dynamické, takže dříve pořízená mapa již neodpovídá skutečnosti. Obvykle je tedy potřeba jak tvořit mapu prostředí, tak se zároveň přesně v prostředí pohybovat. To ale je spojeno s paradoxem: aby se robot mohl pohybovat přesně, potřebuje přesnou znalost o prostředí (mapu). Aby vytvořil přesnou mapu, potřebuje přesné informace o svém pohybu. Vzniká tedy klasický problém „Co bylo dřív, slepice nebo vejce?“. V uplynulých 15-20 letech je tento problém v robotice stále více zkoumán, neboť jeho praktické řešení by umožnilo vytvářet skutečně autonomní roboty, které je možno umístit do neznámého prostředí, kde se pak budou úspěšně pohybovat a orientovat.

Dnes může být „SLAM problem“ považován za vyřešený po teoretické stránce (viz např. [4]). Jeho praktické částečné implementace existují již v obrovském množství realizací pro nejrůznější roboty (vnitřní i venkovní, pro podvodní prostředí i pro létající systémy) a to s velmi nadějnými výsledky (viz např.[2], [11], celé [5] a další). K řešení sice zůstávají podstatné zásadní problémy spojené s obecností postupů (např. [10]), přesto jsou konkrétní implementované postupy velmi nadějné.

8 Použití SLAM pro Sémantický web

Jednou z úloh v oblasti Sémantického webu je získávání instancí ontologií a následně dolování ontologií jako takových a jejich mapování. Jako zdroj dat může například sloužit prostředí webových dokumentů.

Pro tuto úlohu bychom mohli SLAM metody použít takto: webové dokumenty obsahují různé textové a grafické informace a jsou mezi sebou svázány množstvím odkazů. Pro virtuálního robota budou představovat jeho „svět“. Pokud by se nám podařilo zajistit, aby se robot v tomto prostředí uměl pohybovat a vytvářet jeho mapu, pak bychom jako výstup mohli získat právě požadované údaje – ontologie a jejich mapování.

V práci [6] se například autor zabývá OCR zpracováním webových stránek a na základě takto získaných dat vytvářením struktury jednotlivých stránek. Pokud bychom robota „vypustili“ do stránky s úkolem zmapovat ji, pak by výsledná mapa mohla popisovat právě tyto oblasti. V prvním přiblížení bychom si mohli přímo představit jednu grafickou stránku zobrazenou v počítači a malého robota, který se po ní bude pohybovat. Okno prohlížeče by pak podávalo „pohled shora“ na celou scénu, která by byla tvořena barevným obrazem. Podle toho, jakým způsobem by byla data z tohoto obrazu zpracovávána, může robot postupně nalézat množství různých informací: oblasti souvislého textu, obrázky, odlišit pozadí, v obrázcích vyhledávat zajímavé struktury atd.

Na rozdíl od zpracování jednotlivých stránek OCR metodami může robot procházející stránku dále pozorovat a mapovat vztahy mezi stránkami stejným způsobem, jako prochází jednotlivé stránky. Ve spojení s interpretací webové stránky v prohlížeči, zejména používání odkazů, může robot mezi jednotlivými částmi stránek i mezi různými stránkami přecházet tak, jako kdyby v reálném světě prostupoval spojeními mezi různými místnostmi. Přestože je na první pohled zřejmé, že výsledná mapa by byla pro člověka jen těžko představitelná v běžných třírozměrných dimenzích, ničemu to nevadí. Algoritmy použité pro mapování i plánování na „lidské představitelnosti“ založeny nejsou.

9 Paralely - shrnutí

Oblasti, kde by se mohly robotické postupy pro účely Sémantického webu uplatnit, zahrnují například:

- Zjišťování vztahů mezi provázanými dokumenty – tato úloha by odpovídala problému vytváření topologické mapy neznámého prostředí a budování grafu.

- Analýza dokumentu, například stránky zobrazené prohlížečem – tato úloha by odpovídala problému vytváření obecné mapy prostředí.
- Vyhledávání údajů v dokumentu (dokumentech) – tato úloha by odpovídala problému lokalizace robota na základě vstupních dat z jeho senzorů.

10 Závěr

V tomto článku jsme se pokusili naznačit, že některé úkoly nově objevované a zkoumané v oblasti sémantického webu by vlastně mohly být řešeny pomocí postupů z jiného oboru – robotiky. Přestože jsou oba obory na pohled velmi vzdálené, na příkladech jsme ukázali, že by mohly používat velmi podobné metodiky. Zda se jedná či nejedná o slepou uličku, zatím není možné určit; přesto se nyní zdá, že je-li to slepá ulička, pak je alespoň dlouhá a naučná natolik, že poutník, který by se po ní vydal, nezhyne a přinejmenším ze své cesty přinese zajímavé podněty pro další postup.

Poděkování

Práce byla částečně podporovaná projektem 1ET100300419 „Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu" programu Informační společnost tématického programu II Národního programu výzkumu v České republice.

Literatura

[1] Z. Chen, J. Samarabandu, R. Rodrigo, "Recent advances in simultaneous localization and map-building using computer vision," *Advanced Robotics*, Brill, Leiden, ISSN 0169-1864, Vol. 21, No. 3–4, pp. 233–265, 2007

[2] A.J. Davison, N. Kita, "3D Simultaneous Localisation and Map-Building Using Active Vision for a Robot Moving on Undulating Terrain," *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2001*, ISBN 0-7695-1272-0, pp. 384-391, 2001

[3] A. Doucet, N. de Freitas, N. Gordon (Eds.), *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, Berlin / Heidelberg, ISBN 978-0-387-95146-1, 2001

[4] H. Durrant-Whyte, T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics and Automation Magazine*, ISSN: 1070-9932, Vol. 13, No. 3, pp. 99-110, September 2006

[5] C. Laugier, R. Chatila (Eds.), *Autonomous Navigation in Dynamic Environments*, Springer Tracts in Advanced Robotics, Springer-Verlag, Berlin / Heidelberg, STAR 35, ISBN 978-3-540-73421-5, October 2007

[6] D. Maruščák, "Mapovanie a dolovanie ontológií s užívateľskou preferenciou," diplomová práce, MFF UK Praha, 2007

[7] J. Štanclová, F. Zavoral, "Hierarchical Associative Memories: The Neural Network for Prediction in Spatial Maps," *Proceedings of ICIAP 2005*, Cagliary, Italy, Springer-Verlag, Berlin, LNCS3617, ISSN-0302-9743, ISBN 3-540-28869-4, pp. 786-793, September 2005

[8] S. Thrun, W. Burgard, D. Fox, „Probabilistic Robotics," MIT Press, Cambridge, USA, ISBN 0-262-20162-3, September 2005

[9] J. Vandorpe, H.V. Brussel, H. Xu, "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range fnder," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 901-908, 1996

[10] M. Walter, R. Eustice, J. Leonard, "A Provably Consistent Method for Imposing Sparsity in Feature-Based SLAM Information Filters," *Robotics Research*, STAR 28, ISBN 978-3-540-48110-2, pp May 2007

[11] Z. Xiang, W. Zhou, "3D Map Building for Mobile Robots Using a 3D Laser Range Finder," *Proceedings of ICIC 2006*, China, Springer-Verlag, Berlin/Heidelberg, LNCIS 345, ISSN 0170-8643, ISBN 978-3-540-37257-8, pp. 785 – 790, September 2006

Mám hlad: pomůže mi Sémantický web?

Michal Podzimek, Jiří Dokulil, Jakub Yaghob, and Filip Zavoral

Katedra softwarového inženýrství, MFF UK Praha,
michalpodzimek@volny.cz, (dokulil,yaghob,zavoral)@ksi.mff.cuni.cz

Abstrakt *Sémantické úložiště Trisolda slouží jako platforma pro získávání a uchovávání sémantických dat a pro dotazování nad nimi. Pouhé dotazování však neumožňuje plně využít potenciálu platformy. Technika exekutorů rozšiřuje infrastrukturu o procesní modely. Atomické exekutory, jejichž úkolem je provést sémantickou akci, lze pomocí dirigentů složit a vytvořit tak exekutory složené. Článek popisuje implementaci exekutorů a dirigentů v prostředí Trisoldy a analyzuje praktickou použitelnost stávajících servisně orientovaných frameworků pro sémantický web.*

1 Úvod

Oblast Sémantického webu prodělává v posledních letech bouřlivý vývoj [**?**,**?**]. Většina výzkumu se však soustřeďuje na teoretické aspekty Sémantického webu, praktických implementací je jako šafránu a ve většině případů se navíc jedná pouze buď o knihovní systémy s úzkým oborem nebo o jednoduché mesh-upy – služby vzniklé spojením několika jiných služeb, zatím vždy však opět z jednoho oboru. Většina výzkumu se navíc zabývá pouze dotazovací částí Sémantického webu. Pokud však chceme dosáhnout skutečného využití potenciálu Sémantického webu, na část dotazovací musí navazovat část výkonná, schopná manipulace s reálným světem.

V současnosti je pro vědeckou veřejnost volně dostupných několik výzkumných infrastruktur pro Sémantický web [**?**,**?**]. Tyto se však opět soustřeďují pouze na část dotazovací. Na rozdíl od těchto v akademické sféře často citovaných projektů je Trisolda [**?**,**?**] navržena, tak aby mohla fungovat jako výkonná infrastruktura pro Sémantický web podporující výzkum i v dalších oblastech Sémantického webu, včetně možnosti využít výsledků nějakého dotazu k řízení nějakých procesů.

Ukažme si důležitost na jednoduchém příkladě: Známý badatel FZ je večer v práci a bádá nad Sémantickým webem. Náhle si uvědomí, že má hlad a rozhodne se, že si nechá dovézt pizzu. Pokud použije pouze dotazovací část Sémantického webu, dostane se mu seznamu pizzerií, kde si takovou pizzu může objednat. Pokud však navíc použije i výkonnou část, pizza se "sama" objedná a po čase bude dovezena na pracoviště, kde ji FZ již netrpělivě očekává. Při tom budou využity i další důležité aspekty vztahující se přímo k vyhledávání, např. uživatelské preference [**?**].

1.1 Trisolda

Trisolda je infrastruktura pro sémantický web, která si již při svém vzniku (na rozdíl od mnoha jiných akademických projektů) kladla za cíl stabilitu a výkon. Již při návrhu Trisoldy se počítalo s jejím rozvojem a s podporou výkonné části, kterou tvoří dirigenti a exekutoři.

Dirigent je služba, která je zodpovědná za provedení procesu na základě dat dodaných dotazem na dotazovací část Trisoldy. A to bez ohledu na to, zda jde o reálný svět nebo virtuální svět internetu. Dirigenti však neprovádí interakce přímo, nýbrž provádějí orchestraci exekutorů. Exekutor je specializovaný modul Trisoldy zodpovědný za atomickou interakci s reálným nebo virtuálním světem (např. objednání kurýrní služby).

Před započetím této práce nebylo příliš zřejmé, jaké přesně prostředky bude potřeba využít pro realizaci dirigentů a exekutorů. Rovněž tak nebylo předem určeno, jakou podporu infrastruktura musí poskytovat konkrétním implementacím dirigentů a exekutorů.

Jedním z hlavních cílů našeho projektu je pilotní implementace výkonné části pomocí stávajících prostředků. To nám mělo vyjasnit výše zmíněné problémy.

V článku dále popíšeme nejprve návrh konkrétního chování orchestrace a poté způsob jeho implementace stávajícími dostupnými nástroji včetně získaných praktických zkušeností s jejich kombinovatelností.

2 Úkoly a chování výkonné části orchestrace

2.1 Možnosti implementace

Výkonnou část orchestrace lze rozdělit na dva hlavní moduly a další menší pomocné části. První modul má na starosti vykonání procesu. K tomuto mu pomáhá modul druhý, který vykonává relativně jednoduché dotazy a získává tak potřebné informace. Druhý modul je nejlepší rozdělit na více malých částí, které se specializují na jednu úzce definovanou činnost. Tyto části samostatně zpracovávají nějaký dotaz, nazvěme je atomickými exekutory. První modul postupně volá jednotlivé exekutory, řídí neboli orchestruje jejich činnost. Dále bude tedy nazýván dirigent. Další pomocné části mohou mít na starosti například uložení nějakých informací pro statistické účely. Tyto části by měly být volány dirigentem, ale nepodílí se přímo na získávání potřebných informací, a proto není důvod nazývat je exekutory.

Nejpodstatnější částí je samotný exekutor. Je zřejmé, že pro různé činnosti bude třeba používat různé exekutory. Některá činnosti mohou mít exekutory společné. V prvním kroku dirigenta je třeba určit, který exekutor bude pro

aktuální dotaz použit. O výběr exekutora podle dotazu se může starat nezávislá komponenta systému.

Druhý možný přístup je takový, že dirigent jen vybere, který exekutor se má pro daný dotaz použít a zavolá první exekutor z řetězu. Ten vykoná nějakou činnost a její výsledek pošle dalšímu exekutoru. Toto řešení vyžaduje složitější a chytřejší exekutory, na druhou stranu dirigent je jednodušší.

Přístup s chytrým dirigentem a hloupými exekutory najde popis procesu a začne podle něj volat jednotlivé exekutory a skládat jejich výstupy. Exekutor zpravidla vykoná jen jednu malou činnost a nevolá další exekutory (vyloučeno to ale není). V případě druhého možného přístupu dirigent najde jednoho či více exekutorů a ty postupně volá.

Verze s chytrým dirigentem a hloupými exekutory se po zevrubnější analýze jeví vhodnější. Umožňuje snadno vytvořit nezávislé exekutory, které se soustředí jen na získání požadované informace. Tyto exekutory lze pak snáze znovu využít, protože nejsou zatíženi žádnou další funkcionalitou. U dirigenta se soustředí veškerá orchestrace, takže je snazší navrhnout nějaký univerzální popis procesu. Výhodou chytrých exekutorů je, že jejich vzájemné navazování může ušetřit nějakou práci při vytváření popisu procesu.

2.2 Popis dirigenta

V našem návrhu je zvolena varianta chytrého dirigenta a hloupých exekutorů. Nejprve proto popíšeme návrh dirigenta.

V první fázi dirigent zjistí, do jaké třídy patří dotaz. Vybere takovou nejbližší nadtřídu, ke které existuje procesní model. Pokud takovou nelze nalézt, dirigent skončí neúspěchem.

Pokud dirigent k dotazu zná jeho třídu s procesním modelem, může spustit instanci tohoto modelu. Procesu je předán dotaz, třída a preferenční informace o uživateli. Detaily procesu již záleží na konkrétní realizaci modelu. Primárně se předpokládá využití exekutorů, ale nemusí to být podmínkou. V průběhu vykonávání procesu se sbírají potřebná data a jsou opět sestavována podle předpisu v procesu. Proces může data setřídit, filtrovat a i jinak s nimi manipulovat.

2.3 Popis exekutora

Exekutor je relativně malý modul, který se specializuje na jednu konkrétní činnost. Vstupní a výstupní hodnoty exekutora nejsou nijak speciálně určené, protože pro každý úkol potřebuje jiné údaje. O využití exekutorů rozhoduje dirigent. Není vyloučeno, aby jeden exekutor využíval služby jiného exekutora.

2.4 Realizace

Možnosti, jak realizovat navrhovaný systém, jsou různé. Buď lze vytvořit monolitický systém, kde již popsané části budou představovány třídami a celý projekt bude vytvořen víceméně v jednom programovacím jazyku.

Zajímavější možností je vytvořit systém webových služeb, kde každá komponenta bude samostatná webová služba. To umožňuje vytvořit distribuovanou aplikaci, která nebude závislá na jednom programovacím jazyku či platformě. Je třeba jen nadefinovat komunikační protokoly a rozhraní pro jednotlivé služby a konkrétní realizace služby může být libovolná, v případě potřeby i distribuovaná.

Hlavní částí celého systému je procesní model. Nástrojů pro modelování procesů je několik - Business Process Modeling Notation (BPMN), Business Process Execution Language (BPEL), Unified Modeling Language (UML), Web Services Choreography Description Language (WS-CDL) a další. Každý z jazyků má trochu jiný přístup a je vhodný pro odlišné použití. Pro tuto práci jsou nejzajímavější BPEL a WS-CDL, jejich vzájemný rozdíl odpovídá rozdílu mezi choreografií a orchestrací webových služeb.

Oba pojmy souvisí s modelováním spolupráce mezi službami. Jejich rozdíl je v přístupu k umístění logiky, která řídí jejich spolupráci. V případě choreografie je logika rozdělena mezi všechny zúčastněné služby a neexistuje žádný centrální řídící prvek. Choreografie popisuje interakce mezi jednotlivými spolupracujícími službami, ale nezabývá se tím, jak poskytovatel celé služby zajistí její korektní provedení, a proto nedefinuje konkrétní spustitelný proces.

Naopak u orchestrace je logika soustředěna do jednoho místa a popisuje jaké úkony provádí poskytovatel služby pro to, aby zajistil její správné provedení. Orchestrace popisuje interakce mezi poskytovatelem a jednotlivými zúčastněnými službami. Popisuje jí z pohledu poskytovatele, neboli z pohledu strany, která celý proces ovládá. Výsledkem je pak opět nová složitější webová služba.

Řešením tedy bude systém služeb, jehož základem je proces definovaný v jazyce BPEL. Proces bude vykonáván dirigentem a podle popisu bude volat jednotlivé exekutory. Exekutor bude opět webová služba a její realizace může být libovolná.

3 Pilotní implementace

Cílem pilotní implementace je demonstrovat základ funkčnosti dirigenta a procesu pro třídu. Je třeba mít nástroj pro vytváření webových služeb a pro jejich spuštění a dále nástroj pro vývoj BPEL procesů a pro jejich zveřejnění a vykonání.

Vytvářet webové služby lze v dnešní době pomocí téměř všech moderních programovacích jazyků. Pro tuto práci byly vyzkoušeny jazyky Java a C++.

V C++ lze pomocí nástroje gSOAP [**?**] vytvořit samostatný program, který má v sobě implementovánu komunikaci pomocí SOAP. gSOAP vytvoří základní strukturu (skeleton a stub) programu buď z WSDL popisu služby nebo z hlavičkového souboru. Velkou výhodou gSOAP je jeho rychlost a jeho použití zdarma pro akademické projekty.

V Javě lze webové služby vytvářet pomocí nástroje Apache Axis [**?**]. Pro spuštění webové služby je třeba nainstalovat server Tomcat a na něm zveřejnit vytvořenou službu pomocí Axis. Výhodou Axis je, že na jednom serveru může být spuštěno více služeb, kdežto u gSOAP jen jedna. Zásadní nevýhodou ale je rychlost, která je oproti gSOAP pomalejší 10x – 15x.

Druhým problémem je výběr nástrojů pro vývoj a vykonávání BPEL procesů. Jako nejpoužitelnější se ukázal produkt ActiveBPEL, který je dnes součástí většího celku ActiveVOS [**?**]. ActiveBPEL se skládá ze dvou nezávislých částí. ActiveBPEL designer slouží pro vývoj BPEL procesu a k jeho ladění. ActiveBPEL server pak umožňuje proces zveřejnit a používat jako webovou službu.

3.1 Dirigent

Dirigent je implementován pomocí následujících BPEL aktivit. Základem je dvojice aktivit *receive* a *reply*. Aktivita *receive* umožňuje procesu čekat na příchod zprávy a je ukončena jejím příchodem. Aktivita *reply* naopak zasílá zprávu jako odpověď na zprávu přijatou pomocí *receive*. Aktivita *assign* slouží k manipulaci s proměnnými a může obsahovat několik elementárních příkazů přiřazení. Podstatná je také aktivita *invoke*, která se používá k volání externí služby definované pomocí “partner link”.

Jednotlivé aktivity je třeba zřetězit, k tomu slouží buď kontejner *sequence* nebo prvek *link*, který umožňuje definovat přechody mezi jednotlivými aktivitami a případně k nim definovat podmínku přechodu.

Pro volání jiné služby pomocí aktivity *invoke* je třeba znát některé parametry volané služby, zejména adresu, na které je služba přístupná. Tato adresa může být zadána buď staticky, nebo dynamicky. To musí být známo ještě před zveřejněním služby a v ActiveBPEL je to součástí souboru Deployment Descriptor. V případě statického volání je zde určena reference na koncový bod, která obsahuje pevně zadanou adresu a další parametry. V dynamickém případě je reference určena v průběhu vykonávání procesu a k volání partnerské služby je přiřazena pomocí aktivity *assign*. Tímto je umožněno získat adresu služby z návratové hodnoty jiné volané služby.

Služba *dirigent* obsahuje pouze jednu operaci nazvanou *ziskej_informace*. Tato operace na vstupu získá dotaz a preferenční informace.

Jako první operaci dirigent provede zjištění třídy dotazu. To zajistí zavoláním služby *najdi_tridu*. Adresa této služby je pevně zadána v Deployment Descriptoru. Podle návratové hodnoty se pomocí podmínky u prvku link určí, zda má ve zjišťování informací pokračovat, nebo zda má být ukončeno, protože návratová hodnota je *trida_nenalezena*.

V druhém kroku se identifikace nalezené třídy použije pro volání *najdi_proces*, která vrátí název, adresy a popis procesu. V této pilotní implementaci se používá pouze adresa procesu. Ostatní hodnoty se ignorují a slouží pro případná rozšíření, kde by bylo možné vybírat z více procesů podle preferencí uživatele.

Pokud je adresa procesu nalezena, je použita pro aktivitu *invoke*, která má na starosti zavolání procesu. Všechny zde volané služby by měly vycházet z jednoho WSDL souboru, kde by jediné rozdíly měly být v části service a to hlavně v adrese služby. Všechny tyto služby musí být definovány ve stejném jmenném prostoru.

3.2 Problémy s nástroji

Při zkoušení různých nástrojů se objevila spousta problémů. Naprosto zásadním problémem bylo velké množství různých verzí a možných konfigurací jednotlivých nástrojů. Bylo tedy potřeba dlouho zkoušet různé kombinace všech těchto nástrojů, než se podařilo najít plně funkční sadu. Důležité bylo později již neměnit verzi nástroje. Takto se jako velký problém ukázala snaha zkombinovat použití Axis 1.1 a Axis 2. Po zjištění, že Axis 2 způsobuje problémy, bylo třeba přeinstalovat celý server Tomcat, protože přestaly fungovat služby původně vytvořené pomocí Axis 1.1.

Problém také nastává, pokud se nainstaluje server Tomcat a na něm se třeba používá jUDDI server. Potom při spuštění Tomcatu z Eclipse jUDDI server nefunguje. Je to způsobeno tím, že při spuštění z Eclipse se používá jiná konfigurace a jiný domovský adresář pro Tomcat a tudíž i jiný adresář pro zveřejnění webových služeb. Je tedy potřeba najít tento adresář a tam jUDDI také nainstalovat. Většinou bývá někde v domovském adresáři pro projekty.

Obtížné je i ladění webových procesů. V případě samotných služeb ještě občas lze z konzole Tomcatu v Eclipse vyvolanou chybu nějak prozkoumat a pokusit se jí odstranit. V případě procesů se již většinou objevuje jen chyba Internal Server Error, která sama o sobě toho již moc neříká, je třeba postupnými úpravami aplikace a testováním kdy se chyba projeví najít její příčinu. Jednou z těchto chyb může být například špatně napsaná adresa partnerské služby nebo její nekorektní chování.

Dalším problémy způsobuje ActiveBPEL. Na serveru se ukládají zdroje (Resources), jako jsou WSDL popisy, XSD schémata apod. Tyto zdroje se při zveřejnění nové verze procesu, a tedy i služby, na serveru neaktualizují. Dochází tedy například ke špatnému popisu celé služby a při pokusu o zveřejnění nebo použití služby se to může projevit různými chybami. Zajímavé je, že při zveřejnění

procesu se některé poznámky o změně verze zdroje objevují a některé ne. Po důkladném prozkoumání přesné SOAP komunikace mezi službou starající se o zveřejnění a ActiveBPEL designer lze nalézt informace o tom, které zdroje byly změněny, a které zůstaly v původních verzích.

Problémy se mohou objevit i při používání knihovny UDDI4J. Je třeba dát pozor na to, že využívá konfigurační soubor pro nastavení parametrů pro získání dat z UDDI registrů. K tomuto konfiguračnímu souboru je třeba správně nastavit cestu. Problémem je opět různá konfigurace Tomcatu při spouštění z Eclipse a standardně mimo Eclipse. Různý je totiž i aktuální adresář, od kterého se vyhodnocují relativní cesty. Navíc se u UDDI4J objevily problémy s připojením k UDDI registrům. Nedařilo se připojení k registrům, ze kterých se snadno získávala data pomocí Web Services Exploreru v Eclipse.

4 Závěr

Během pilotní implementace jsme narazili na různé obtíže s použitím volně dostupných nástrojů. To jen potvrzuje naši tezi, že současné nástroje a prostředky používané v oblasti Sémantického webu jsou vyvíjeny jako specializované akademické prototypy, které jsou vhodné pouze pro původní účely a pro reálné nasazení jsou zcela nevhodné, včetně rychlosti a stability produktů.

Bez ohledu na tyto potíže však pilotní implementace v omezené míře fungovala a poskytla nám možnost provádět s ní různé experimenty.

Získané zkušenosti využijeme v dalším rozšiřování a obohacování aplikačního rozhraní infrastruktury Trisolda. Zvláštní pozornost věnujeme návrhu části rozhraní přímo podporující výkonné procesy v reálném světě, neboť jejich podpora ve stávajících nástrojích a technologiích buď vůbec neexistuje nebo je pro potřeby sémantické orchestrace zcela nedostatečná.

Reference

1. ActiveVOS. http://www.activevos.com/.
2. Apache Axis. http://ws.apache.org/axis/.
3. J. Broekstra, A. Kampman, and F. Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proceedings of the First International Semantic Web Conference*, pages 54–68, Italy, 2002.
4. John Davies, Miltiadis D. Lytras, and Amit P. Sheth. Guest editors' introduction: Semantic-web-based knowledge management. *IEEE Internet Computing*, 11(5):14–16, 2007.
5. Jiří Dokulil, Jaroslav Tykal, Jakub Yaghob, and Filip Zavoral. Semantic web infrastructure. In Patrick Kellenberger, editor, *First IEEE International Conference on Semantic Computing*, pages 209–215, Los Alamitos, California, 2007. IEEE Computer Society.
6. Alan Eckhardt, T. Horváth, and Peter Vojtáš. PHASES: A user profile learning approach for web search. In Tsau Lin, Laura Haas, R. Motwani, A. Broder, and H. Ho, editors, *2007 IEEE/WIC/ACM International Conference on Web Intelligence - WI 2007*, pages 780–783. IEEE, 2007.
7. Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/.
8. Rudi Studer. The semantic web: Suppliers and customers. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 995–996. Springer, 2006.
9. Robert van Engelen and Kyle Gallivan. The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks. In *CCGRID*, pages 128–135. IEEE Computer Society, 2002.
10. J. Yaghob and F. Zavoral. Semantic Web Infrastructure using DataPile. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Itelligent Agent Technology*, pages 630–633, Los Alamitos, California, 2006. IEEE. ISBN 0-7695-2749-3.

Framework for mining of association rules from data warehouse

Lukáš Stryka[1] and Petr Chmelař[2]

[1]Department of Information Systems, Faculty of Information Technology, Brno University of Technology
Božetěchova 2, Brno, 612 00, Czech Republic

[2]Department of Information Systems

Abstract. *In this paper, we propose a framework for association rules mining from data warehouses. This framework presents alliance between two business intelligence areas. First area is represented by data warehouse and data cube providing high quality data. The second area is represented by data mining, especially association rules mining providing an additional knowledge.*
Association rules mining on data warehouses is different from mining on relational or transactional databases, because it deals with couple of dimensions, which form conceptual hierarchies. Thus we mine multi- and inter-dimensional association rules. There are several approaches how to mine such association rules described in literature. This framework presents a novel combination of the data cube processing - top-down (on product dimensions) and bottom-up (on domain dimensions). We presume division of dimensions on domain and product dimensions.
The framework works in the following steps. The first one represents obtaining frequent leaf 1-itemsets, which means obtaining frequent itemsets from domains represented by items from domain dimensions on leaf level. In the second step we obtain all frequent 1-itemset. Following step represents iterative mining of frequent k-itemset from frequent (k-1)-itemsets. In the final step we process all k-itemsets and obtain association rules from them.

1 Introduction

There are huge amounts of data stored in databases. Thus, it is very difficult to make decisions based on this data. Decision support problems have been motivating a development of sophisticated tools which provide a new view on data for better data understanding. These tools are used for business analysis, scientific research, medical research and many other areas. These tools can be based on data mining techniques, OLAP (On-Line Analytical Processing), data warehouses, etc.

There are many algorithms and methods for data mining on transactional and relational data. But following the requirements of science or commercial sphere the expansion of storing structured or semi-structured data has been coming up.

Nowadays, the most of big companies uses data warehouses for data processing and analyzing. Operational data are stored in the classic OLTP databases, but this data are cleaned and stored in data warehouses at regular intervals.

We develop a system for multidimensional and multilevel data mining of association rules. This method is developed for shopping basket analysis task. It should be able to provide useful information and knowledge for target marketing for special marketing strategies preparation.

My approach enables to mine the association rules from the data stored in data warehouses with given conceptual hierarchy. This approach divides the dimensions into two main groups. The first group contains product dimension, which represents a conceptual hierarchy of sold goods. The second group consists of domain dimensions, which represents other dimensions like geographical or time dimension hierarchies.

The system will use XMLA (XML for Analysis) for communication with data warehouse. This way of communication provides portability between many data warehouse solutions from different companies.

Motivation example

We can presume a data warehouse storing sales data from branches in several countries or regions. It's obvious, that there will be differences in sales of some commodities. Of course there are also some differences in such sales during the calendar year because of seasonal character of some commodities.

Now we can illustrate a situation demonstrating advantage of mining method working in special context domain. We can presume big international store company with widespread commerce representation. It's obvious that there are some kinds of commodities which are soled typically in certain regions. So we can prepare target marketing campaigns or strategies focused on concrete types of goods or customers.

We have sales data of mountaineering boots and climbing ropes. If we will count a support value of these goods in whole sales amount of company, we find out, that the minimum support threshold is not satisfied. But if we divide the whole sales area into smaller parts, we can find regions in which the minimum support of these goods is satisfied. We can identify a region with mountains where most of people wear mountaineering boots whole year. In contrast to this we can identify the same region for satisfying of minimum support of ropes, but we can also identify a region with big sandstone rocks. If we compute a confidence of the (mountain) rule rope $\rightarrow$ mountaineering boots or (mountain) mountaineering boots $\rightarrow$ rope, we discover that the minimum confidence threshold is not satisfied. So we divide the mountain region sales by the season of the year and then we can identify the rule (mountain, spring) rope $\rightarrow$ mountaineering boots satisfying the minimum confidence threshold.

Ergo this example shows that we can find interesting rules in smaller contexts. These rules provide us the potentially interesting information for target marketing.

2 State of the art

In this chapter the state of the art will be introduced. There are three kinds of data warehouse applications: information processing, analytical processing, and data mining.

2.1 Data mining

A generally accepted definition of data mining and knowledge discovery is given by Fayyad et al. (1996) as "the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data". Data mining is not a straightforward analysis nor does it necessarily equate with machine learning. It is non-trivial, usually in the sense that the dataset under consideration is large.

2.2 Association analysis

Association analysis [4] is one of the data mining methods. It provides us association rules as a knowledge mined from different types of data. An association rule is a rule in the form of :

$$A1, A2,\ldots, Am \rightarrow B1, B2,\ldots, Bn$$

where Ai and Bj are predicates or items. It means, when left side of rule occurs in transaction, then, there is high probability that the right side of rule occurs in this transaction as well. Result rules are characterized by two main parameters: the support and the confidence. The support determines probability of reoccurrence transaction including both sides of rule in transaction set. The confidence specifies probability of re-occurrence of the right side of rule in only those transactions, where the left side of rule is included.

2.3 Multilevel association rules

Multilevel association rules can be mined using several strategies, based on how minimum support thresholds are defined at each level of abstraction, such as uniform support, reduced support, and group-based support. Redundant multilevel association rules can be eliminated if their support and confidence are close to their expected values, based on their corresponding ancestor rules. In our case we determine specific value of Support for each level specially.

Nowadays, some researches have applied the association rule mining on the data cubes and other warehouse techniques, originated by Han et. al. in [3]. The generalized association rules mining algorithm and a survey of other approaches can be found in [6], however the disadvantage of all cited techniques is the need for specification of the support and confidence thresholds on each level of the concept hierarchy or the cube granularity.

2.4 Dynamic minimum support determining

We established the new parameter called cover [1]. It means a percentage cover of examined data. It is based on the Paretto analysis -- minor part of classes covers major part of data. So the Cover parameter determines which data are significant and essential to cover required part of data.

We have used ideas of information theory. It is similar to the coding of entropy. It works similarly (reversely) to the Huffman coding algorithm. The difference is that Huffman is merging the least significant values (with lowest information value or support) while learning the code, our Inverse Huffman algorithm is merging the most significant values and the least significant doesn't take into consideration at all. So we employed a lossy compression of analyzed data. We describe the algorithm rather than formalizing the problem.

2.5 OLAM

OLAP is a data summarization/aggregation tool that helps to simplify data analysis, while data mining allows the automated discovery of implicit patterns and interesting knowledge hidden in large amounts of data. The main difference between OLAP and DM is that OLAP is based on interactive user-defined hypothesis testing while DM is relatively slow generation of such hypotheses.

As it was outlined, our method is similar to principle of Han's OLAM methods. On-Line Analytical Mining (OLAM) [5] provides tools for data mining on different granularities. It means on different subsets or different levels of abstraction by drilling, pivoting, filtering, dicing and slicing on a data cube and on some intermediate data/knowledge results.

OLAM represents methods implemented in DBMiner system [2] included in Intelligent Database System Research Laboratory. These methods integrate On-Line Analytical Processing (OLAP) principles and data mining methods for multidimensional data mining in large databases and data warehouses. An OLAM engine performs analytical mining in data cubes in a similar manner as an OLAP engine performs on-line analytical processing. This engine accepts user's on-line queries and work with the data cube in the analysis. So the engine may perform multiple data mining tasks, such as concept description, association, classification, prediction, clustering, etc. OLAM uses more powerful data cube construction than OLAP because OLAM analysis often involves the analysis of large number of dimensions with inner granularities. Construction of data cube is following: if data cube contains a small number of dimensions, or if it is generalized to high level, the cube is constructed as compressed sparse array but is still stored in a relational database to reduce costs of construction and indexing of different data structures.

3 Algorithm

The mining algorithm consists from few steps. First step deals with generating of the frequent 1-leaf-itemsets. These mean such frequent itemsets which contain product item on given level of conceptual hierarchy and set of items on lowest level of each domain dimension. These items represent a domain of such itemsets. Firstly the support value for initial itemset containing product item on highest level of conceptual hierarchy is calculated. After that this support value is compared to domain minimum support value. If itemset support value is smaller than given minimum support value, the dissatisfactory value of itemset is set at true. Dissatisfactory value represents a fact that in give domain is occurrence of product item smaller than required threshold. This value helps with decreasing of computational complexity.

For computation of minimum support can be used one of two strategies. First strategy calculates with fix minimum support value. The second strategy uses the cover value to determine domain minimum support value dynamically. This step uses the Inverse Huffman algorithm joining the items with the highest occurrence up to achieve a cover threshold.

3.1 Leaf items generation

When the all itemsets with the top level product items are processed, we move to lower level of the conceptual hierarchy. Firstly, the existence of predecessor itemset with dissatisfactory value set at false in given domain is tested. If such itemset is not found, we can skip tested itemset and we set its dissatisfactory value to true. Else we check if the support value of the tested itemset is up to the domain minimum support value. This is repeated until we reach the top level of product conceptual hiearchy.

```
Function generate_leaf_1_itemsets(){
  For each leaf_domain domain{
    For product_level=0 to
    Product_dimension.Levels.length{
      Domain.min_supp = gen_min_supp(domain, cover);
       For each item in
       Product_dimension.Levels(product_level){
         If product_level &
         pred(item).dissatisfactory(domain)==false{
              Supp = gen_sup(item, domain);
              If supp<domain.min_supp
                 Item.dissatisfactory(domain) = true;
              }
         }
       }
    }
}
```

3.2 Generating of more generalized 1-itemsets

Now we have a set of frequent leaf 1-itemset. So we try to joining the domains of frequent itemsets in compliance with conceptual hierarchies of single domain dimensions. For each domain dimension we try to move to top level of this dimension. Firstly we move to higher level of conceptual hierarchy of given dimension and we try to find any successor itemset (all items except item from the given dimension are same) with dissatisfactory parameter set at false. If there is no such successor, it indicates, that the given product item can't be in given domain. So we set a dissatisfactory parameter to false and we move to another dimension.

```
Function gen_all_1_itemsets(){
  Foreach leaf_1_itemset basic_itemset{
     Foreach Dimensions Di{
        Exist_on_level = null;
        For j = (Di.Levels.length-1) downto 0 {
           dij = basic_itemset.domain;
           dij(i)=Di.Levels(j);
          If not exist any
           (item.dissatisfactory(succ(dij))==false){
              Item(dij).dissafisfactory = true;
           }
           else{
             item.supp(dij) = count_supp(dij, item);
             if item.supp(dij) < gen_min_supp(dij, cover){
               item.dissatisfactory(dij) =true;
               Itemsets.add(new Itemset(item, dij);
             }
           }
        }
       }
     }
}
```

After this step we have a set of frequent 1-itemsets. In the next step we find frequent all-itemsets which contain k items from a product dimension (there is no product item which is ancestor or predecessor of any other product item). For such k-itemset, the support value is computed and compared with the given domain the minimum support threshold. If the condition of minimum support threshold is met, we add given k-itemset into result set of the frequent itemset.

```
Function gen_all_k_itemsets(){
  Itemsets = one_itemsets;
  While(true){
    Foreach itemseti in itemsets{
       For each itemsetj in one_itemsets{
        If itemseti.domain==itemsetj,domain &
        not itemseti.contains(itemsetj) {
          Itemset new itemset(itemseti);
          Itemset.add(itemsetj);
          Itemset.supp =
          count_supp (itemset.domain, itemset);
          If Itemset.supp ≥ gen_min_supp(itemset.domain,
          cover){
            Itemsets.add(itemset);
            any_added = true;
          }
        }
       }
    }
    If any_added == false
       Break;
    Else
       any_added=false;
  }
}
```

When we have all frequent itemsets we generate association rules with similar way to classic association rules generating from transactional databases.

4 Mining example

We have used sample database from [6] for the multidimensional association rules demonstration in Table 1.

The sample association rules may be then as in the Table 2.

Region	Basket		
North	{ Mtn.Bike Cube, Camel bag }		
South	{ Road bike Cube , Hydra Pack bag }		
West	{ Road Bike Cube, Merida road bike }		
East	Merida road bike , crankset Gebhardtr }		
Central	{ Mtn. bike Cube}		
Borderland	{ Road bike Merida }		

Tab. 1. Sample transaction database .

Multi-layer Association Rule
(buys, Water bags) => (age, [20,29])
(income, [jih => (buys, Biker)
(buys, Bike) => (income, [40 k,49 k])
(buys, Road Bikes) => (age, [20,29])
(age, [30,39]) => (buys, ,tn bike)
(buys, Laptop) => (age, [30,39])
(buys, Mtn. bike) => (income, [40 k,49 k])
(buys,Mtn BIKE) => (income, [40 k,49 k])
(age, [20,29]) => (buys, Road bike Merida
(buys, Road bike) => (age, [20,29])
(buys, Road bike Meridda => (income, [40 k,49 k])
(buys, mtb. Bike => (income, [40 k,49 k])
(age, [20,29]) => (income, [40 k,49 k]) ^ (buys, Biker)
(income, [40 k,49 k]) ^ (buys, Crankset) => (age, [20,29])
(age, [20,29]) => (income, [40 k,49 k]) ^ (buys, Carbon road l
(buys, Carbon road fbke) => (income, [40 k,49 k]) ^ (age, [20

Tab.2. Association rules.

5 Communication between mining engine and data warehouse

Proposal framework consists from two main parts. First part is represented by data warehouse server engine. The second is represented by data mining engine. Data warehouse provides high quality cleaned data for data mining engine. It can also provide some auxiliary computation. For example it can provide support value determination. Data mining engine communicates with data warehouse engine via XMLA documents which performs required data specification and transfer.

XMLA (XML for Analysis) is the most recent attempt at a standardized Application Programming Interface (API) in the Online Analytical Processing (OLAP) and Business Intelligence (BI) space. It has already gained broad support with companies like Hyperion, Microsoft, SAP, and SAS supporting it.

XML for Analysis is a standard that allows client applications to talk to multi-dimensional or OLAP data sources. The communication of messages back and forth is done using web standards – HTTP, SOAP, and XML. The query language used is MDX, which is the most commonly used multi-dimensional expression language today. Hyperion's Essbase, Microsoft's Analysis Services, and SAP's Business Warehouse all support the MDX language and the XMLA specification.

Technically speaking, XML for Analysis (XMLA) is a specification for a set of XML message interfaces that use the industry standard Simple Object Access Protocol (SOAP) to define data access interaction between a client application and an analytical data provider working over the Internet. Using a standard API, XMLA provides open access to multi-dimensional data from varied data sources – any client platform to any server platform – through web services that are supported by multiple vendors.

6 Conclusion

We have designed a new algorithm for mining of association rules over data warehouses. The algorithm uses a division of dimensions into domain and product dimensions. Domain dimensions create a context of mined frequent itemsets or association rules.

The algorithm works in few steps. In the first step it generates frequent 1-itemsets in the context of the domain dimensions items on the lowest level of conceptual hierarchy. In the second step, all more generalized 1-itemsets are generated. In the third step, the all frequent itemsets are found. Finally, the association rules are mined from the frequent itemsets.

We have also designed the architecture of mining system. It consists from few basic layers. On the top, the presentation layer is located. It presents the result provided by the data mining engine layer. n the top, the presentation layer is located. It presents the result provided by the data mining engine layer his data mining works on the data warehouse. The communication between the data warehouse and the data mining engine is solved via XMLA channel.

References

1. Chmelař P., Stryka L.: Simplified Progressive Data Mining, In: Proceedings of the 16th International Conference on Systems Science, Wroclaw, PL, PWR WROC, 2007, s. 1-10, ISBN 978-83-7493-340-7
2. Han J. and et al.: DBMiner: A system for mining knowledge in large relational databases. In Proc. 1996 Int'l Conf. on Data Mining and Knowledge´Discovery (KDD'96), pages 250-255, Portland, Oregon, 1996.
3. Han J.: Towards on-line analytical mining in large databases. ACM Special Interest Group on Management of Data, 1998.
4. Stryka L.: Association rules mining modul. Master's thesis, BUT Brno, Brno, 2003.
5. Zhu H.: On-line analytical mining of association rules. Master's thesis, Burnaby University, Burnaby, British Columbia V5A 1S6, Canada, 1998.
6. Zhang H. et. al.: Generalized Association Rule Mining Algorithms based on Data Cube. In: Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, Eighth ACIS International Conference on , volume: 2, pages: 803-808, 2007, ISBN: 978-0-7695-2909-7.

Algoritmus na hľadanie množiny izotopizmov medzi Latinskými štvorcami

Marek Sýs

Katedra aplikovanej informatiky a výpočtovej techniky, FEI STU Bratislava,
marek.sys@stuba.sk

Abstrakt *V článku popisujeme algoritmus, ktorý pre dané štvorce nájde všetky izotopizmi medzi nimi efektívnejšie ako bolo doteraz známe. Tento algoritmus pracuje so špeciálnou reprezentáciu štvorcov, kde je každý riadok interpretovaný ako permutácia. Hlavný algoritmus využíva ďalší, ktorý dokáže pre dve množiny nájsť všetky také permutácie,ktoré pomocou konjugácie zobrazia jednu z nich na druhú. V článku ďalej analyzujeme zložitosť hlavného algoritmu, pomocou ktoreho sme schopní lepšie určiť hornú hranicu počtu izotopizmov aká bola doteraz známa.*

1 Základné pojmy

Latinské štvorce sa často používajú, ako funkcie dvoch premenných v rôznych šifrovacích algoritmoch. Poznanie, ku ktorej izotópnej triede príslušný štvorec patrí, a ako vyzerajú jednotlivé izotopizmi medzi danými dvoma štvorcami, je jedným z viacerých nástrojom ako analyzovať ich bezpečnosť. Tento proces je však z hľadiska časovej zložitosti vysoko náročný. Naším cieľom je preto popísať nový algoritmus, ktorý nájde všetky izotopizmy medzi dvoma Latinskými štvorcami a určiť jeho zložitosť. Na začiatok si ozrejmíme základné pojmy, s ktorými v článku narábame.

Latinský štvorec je tvorený tabuľkou rozmerov $n \times n$ nad n rôznymi symbolmi, pričom sa každý symbol nachádza v každom riadku a stĺpci práve raz. Pre jednoduchosť budeme ďalej predpokladať, že každý Latinský štvorec $L = (l_{ij})$ je tvorený symbolmi $\{1, 2, \ldots, n\}$.

Hovoríme, že dva Latinské štvorce sú izotópne, ak permutovaním riadkov, stĺpcov a symbolov jedného štvorca získame druhý. Aplikáciu trojice riadkových, stĺpcových a symbolových permutácií (r, c, s) na štvorec L, budeme označovať v zhode s [1] ako $L^{(r,c,s)}$. Naším cieľom teda je nájsť pre dané štvorce L_1, L_2 rádu n, všetky trojice $(r, c, s) \in S_n \times S_n \times S_n$ také, že platí $L_1^{(r,c,s)} = L_2$. Pozrime sa teraz na základné črty algoritmu.

1.1 Reprezentácia Latinských štvorcov

Náš algoritmus využíva riadkový zápis Latinského štvorca. V ňom je riadok $r = (r_1, \ldots, r_n)$ interpretovaný ako permutácia

$$r = \begin{pmatrix} 12 \cdots n \\ r_1 \ldots r_n \end{pmatrix}.$$

Latinský štvorec L rádu n je tak zapísaný n prvkovou množinou permutácií z S_n (symetrická grupa stupňa n). Túto množinu budeme pre štvorec L ďalej označovať P_L. V akom vzťahu sú množiny P_{L_1}, P_{L_2} izotópnych štvorcov možno vidieť z vety .

Veta 1 *Latinské štvorce $L_1, L_2 \in \mathcal{L}(n)$ sú izotópne práve vtedy, keď existujú permutácie $\varphi, \psi \in S_n$ také, že platí*

$$P_{L_2} = \varphi P_{L_1} \psi. \tag{1}$$

Navyše platí, že ak také $\varphi, \psi \in S_n$ existuje, potom tiež existuje izotopizmus (θ, φ, ψ) štvorcov L_1, L_2 pre nejaké $\theta \in S_n$.

Z vety teda máme, že stačí nájsť všetky také dvojice permutácií φ, ψ, pre ktoré je splnená rovnica (1) a na základe Poznámky (1), už ľahko získame množinu izotopií.

Poznámka 1 *Riadkovú permutáciu θ izotopizmu možno nájsť aplikáciou (id, φ, ψ) na štvorec L_1 a porovnaním riadkov novovzniknutého štvorca $L_1^{(id,\varphi,\psi)}$ a L_2.*

Hľadať príslušné dvojice permutácií φ, ψ možno testovaním všetkých $(n!)^2$ možných dvojíc. Toto je však značne nepraktické. Preto sme hľadali ďalší spôsob, akým sa k nim dá dopracovať. Výsledkom našej snahy je algoritmus, ktorého základnú kostru teraz popíšeme.

2 Základný algoritmus

Hlavná myšlienka algoritmu vychádza z nasledujúcej vety a jej dôsledku.

Veta 2 *Majme Latinské štvorce $L_1, L_2 \in \mathcal{L}(n)$ a ľubovoľné $p_1 \in P_{L_1}$. Štvorce L_1, L_2 sú izotópne práve vtedy, keď existujú permutácie $p_2 \in P_{L_2}, p \in S_n$ také, že platí*

$$p P_{L_1} p_1^{-1} p^{-1} = P_{L_2} p_2^{-1}. \tag{2}$$

Dôsledok 1 *Ak sú Latinské štvorce L_1, L_2 izotópne, tak existuje izotopizmus $(\theta, p, p_1^{-1} p^{-1} p_2)$, kde θ je nejaká permutácia z S_n a permutácie p, p_1, p_2 sú práve tie, ktoré vystupujú v rovnici (2) .*

Poznámka 2 *Dôkazy všetkých viet a tvrdení možno nájsť v [7].*

Ich použitím možno skonštruovať algoritmus, ktorý nájde všetky dvojice φ, ψ, pre ktoré je splnená rovnica (2).

Algoritmus 1 *Vstup: Latinské štvorce L_1, L_2.*
Výstup: Množina Is dvojíc (φ_i, ψ_i)

1. *Zvoľ si ľubovoľnú permutáciu $p_1 \in P_{L_1}$.*
2. *Vezmi v cykle každú permutáciu $p_2 \in P_{L_2}$ rob pre ňu nasledovné:*
   (a) *generuj postupne v cykle všetky permutácie p z S_n.*
   (b) *Ak pre nejaké p platí $pP_{L_1}p_1^{-1}p^{-1} = P_{L_2}p_2^{-1}$ ulož novú dvojicu $\varphi_i = p, \psi_i = p_1^{-1}p^{-1}p_2$ do Is.*
3. *Vráť Is.*

Ako vidno algoritmus (1) vlastne testuje, či je splnená rovnica (2) pre ľubovoľné fixné p_1 a všetky možné páry $p_2 \in P_{L_2}, p \in S_n$. To znamená, že jeho výpočtová zložitosť je $n(n!)$. Napriek tomu, že sme podstatne znížili výpočtovú zložitosť z $(n!)^2$ na $n(n!)$, dá sa táto ešte ďalej znížiť.

Treba si uvedomiť, že množiny $P_{L_1}p_1^{-1}, P_{L_2}p_2^{-1}$ vystupujúce v rovnici (2) sú konjugované (konjugovanosť pozri v [2]), a teda tento fakt sa dá využiť pri hľadaní permutácií p.

3 Optimalizácia algoritmu

V tomto odseku sa zameriame na to, ako čo najviac zúžiť množinu S_n testovaných kandidátov na permutáciu p. Ako sme už povedali, skôr chceme využiť fakt, že množiny $P_{L_1}p_1^{-1}, P_{L_2}p_2^{-1}$ sú konjugované. Konjugované prvky (množiny) vykazujú vo všeobecnosti veľa "rovnakých" vlastností. Pre nás je z nich najdôležitejšou, že konjugované permutácie majú rovnakú cyklovú štruktúru [3].

Definícia 1 *Permutácia p je cyklická (tvorí cyklus), ak sa dá zapísať ako*

$$p = \begin{pmatrix} a_1 a_2 \cdots a_{n-1} a_n \\ a_2 a_3 \cdots a_n a_1 \end{pmatrix}.$$

Cyklickú permutáciu p budeme ďalej zapisovať jednoriadkovo ako $p = (a_1 a_2 \cdots a_{n-1} a_n)$.

Táto vlastnosť sa dá využiť pri optimalizácii algoritmu (1). Vychádzame z vety (3) (Dôkaz pozri v [3]).

Veta 3 *Ak permutácia $q = (q_1, q_2, \cdots, q_n) \in S_n$ tvorí cyklus dĺžky $m \leq n$,, potom aj konjugovaná permutácia pqp^{-1} tvorí cyklus dĺžky m. Naviac sa dá tento cyklus vyjadriť ako*

$$pqp^{-1} = (p^{-1}(q_1), p^{-1}(q_2), \cdots, p^{-1}(q_m)). \qquad (3)$$

Z vety je zrejmé, že každá permutácia p, ktorá konjugáciou zobrazí cyklus $\gamma = (\gamma_1, \gamma_2, \cdots, \gamma_m)$ na cyklus $\delta = (\delta_1, \delta_2, \cdots, \delta_m)$ sa dá zapísať ako spojenie bijekcie $\omega_{\gamma\delta} : \{\gamma_1, \gamma_2, \cdots, \gamma_m\} \mapsto \{\delta_1, \delta_2, \cdots, \delta_m\}$ a bijekcie $\bar{\omega}_{\gamma\delta} : I_n - \{\delta_1, \delta_2, \cdots, \delta_m\} \mapsto I_n - \{\gamma_1, \gamma_2, \cdots, \gamma_m\}$.

Poznámka 3 *Spojenie $\omega_1 \vee \omega_2$ zobrazení $\omega_1 : D_1 \to H_1$ a $\omega_2 : D_2 \to H_2$ je pre $D_1 \cap D_2 = \emptyset$ také zobrazenie $\omega_3 : D_1 \cup D_2 \to H_1 \cup H_2$, že jeho funkčné hodnoty sa dajú vyjadriť ako $\omega_3(x) = \omega_i(x)$ pre $x \in D_i$.*

Keďže cyklus q dĺžky m má m možných ekvivalentných zápisov, je každá $\omega_{\gamma\delta}$ tvaru

$$\omega_{\gamma\delta} = \begin{pmatrix} \delta_1 \delta_2 \ldots \delta_{n-i+1} \delta_{n-i+2} \ldots \delta_n \\ \gamma_i \gamma_{i+1} \ldots \gamma_n \gamma_1 \ldots \gamma_{i-1} \end{pmatrix}. \qquad (4)$$

Poznámka 4 *Bijekciu f zapísanú ako*

$$f = \begin{pmatrix} a_1 a_2 a_3 \ldots a_m \\ b_1 b_2 b_3 \ldots b_m \end{pmatrix}$$

chápeme ako funkciu z definičného oboru D=$\{a_1,\ldots,a_m\}$ do oboru hodnôt $H = \{b_1, \ldots, b_m\}$, kde pre funkčné hodnoty platí predpis $f(a_i) = b_i$, pre všetky $i \in \{1, \ldots, n\}$.

V nasledujúcom odseku popíšme, ako nájsť pomocou jednotlivých bijekcií $\omega_{\gamma\delta}$ množinu všetkých takých p, pre ktoré platí $pqp^{-1} = r$. Hľadanú množinu permutácií $p \in S_n$ budeme ďalej označovať q_r.

3.1 Algoritmus na hľadanie q_r

Vychádzajúc z vety (3) vieme, že každý m cyklus (cyklus dĺžky m) sa konjugáciou zobrazí opäť na m cyklus.

Vieme teda, že všetky m cykly γ_i z q sa pomocou konjugácie zobrazia na m cykly δ_i z r. Označme zobrazenie, ktoré mapuje m cykly vyššie uvedeným spôsobom, symbolom ω_m(také nebude jediné). Keďže všetky cykly sú disjunktné, dá sa každá permutácia $p \in q_r$ zapísať ako spojenie $p = \bigvee_{m=1}^{n} \omega_m$. Ako takéto ω_m hľadať, si ukážeme v nasledujúcom odstavci.

Algoritmus na hľadanie bijekcií ω_m Prepokladajme teda, že máme množiny m-cyklov $\Gamma_m = \cup_i \gamma_i, \Delta_m = \cup_i \delta_i$ po rade z permutácií q, r. Z toho, že konjugované permutácie majú rovnakú cyklovú štruktúru sú množiny Γ_m, Δ_m rovnako mohutné, alebo q, r nie sú konjugované. Predpokladajme teda, že konjugované sú a pre mohutnosť platí $|\Gamma_m| = |\Delta_m| = k$.

Z toho, že cykly γ_i a δ_i sú disjunktné a z vety (3) plynie, že každá bijekcia ω_m, ktorá zobrazí cykly $\Gamma_m = \cup_i^k \gamma_i$ na $\Delta_m = \cup_i^k \delta_i$ je tvaru $\bigvee_{i=1}^{k} \omega_{\gamma_i, \delta_{\chi(i)}}$, kde χ je nejaká permutácia z S_k.

Naopak platí, že pre ľubovoľnú permutáciu $\chi \in S_k$ zobrazí bijekcia $\bigvee_{i=1}^{k} \omega_{\gamma_i, \delta_{\chi(i)}}$ každý z cyklov γ_i na $\delta_{\chi(i)}$. Vieme teda, ako nájsť všetky také ω_m. Pseudokód algoritmu hľadajúceho množinu Ω_m pozostávajúcu zo všetkých ω_m potom vyzerá nasledovne:

Algoritmus 2 *Vstup: cykly $\gamma_1, \ldots, \gamma_k$ z q a $\delta_1, \ldots, \delta_k$ z r dĺžky m.*
Výstup: množina Ω_m bijekcií ω_m.

1. *Nastav Ω_m na prázdnu množinu.*
2. *Vezmi v cykle všetky permutácie $\chi \in S_m$.*
   (a) *Pre každú permutáciu χ nájdi všetky $\omega_{\gamma_i,\delta_{\chi(i)}}$ pre $i = \{1, \ldots, k\}$.*
   (b) *Ulož všetky nové $\bigvee_{i=1}^{k} \omega_{\gamma_i,\delta_{\chi(i)}}$ do Ω_m, kde každá $\omega_{\gamma_i,\delta_{\chi(i)}}$ prechádza cez m možností .*
3. *Vráť množinu Ω_m.*

Pre analýzu zložitosti výsledného algoritmu je nutné poznať mohutnosť množiny Ω_m. Tú je možné odvodiť nasledovným spôsobom. Pre fixné $\chi \in S_k$ sa nové bijekcie ω_m získajú spojením $\bigvee_{i=1}^{k} \omega_{\gamma_i,\delta_{\chi(i)}}$, kde každé $\omega_{\gamma_i,\delta_{\chi(i)}}$ je možné voliť m spôsobmi (rovnica 4) a teda získame práve m^k nových ω_m. Táto úvaha platí pre všetky $\chi \in S_k$ a teda výsledná množina Ω_m obsahuje $(k!)m^k$ prvkov.

Ako sme si povedali skôr, každá permutácia $p \in q_r$ sa dá zapísať ako spojenie $p = \bigvee_{m=1}^{n} \omega_m$. Môžeme teda napísať algoritmus na nájdenie q_r.

Algoritmus 3 *Vstup: Permutácie $q, r \in S_n$*
Výstup: Množina q_r.

1. *Rozlož permutácie z q, r na disjunktné cykly.*
2. *Vytvor množiny Γ_m, Δ_m zo všetkých cyklov dĺžky m pre každé $m \in \{1, \ldots, n\}$.*
3. *Nájdi množiny Ω_m pre každé $m \in \{1, \ldots, n\}$ algoritmom (2).*
4. *Pridaj všetky spojenia tvaru $\bigvee_{m=1}^{n} \omega_m$ do q_r, kde ω_m prechádza cez všetky prvky Ω_m.*
5. *Vráť q_r.*

Z algoritmu je zrejmé, že počet prvkov q_r sa rovná súčinu $\prod_{m=1}^{n} |\Omega_m|$ mohutností jednotlivých Ω_m, keďže v kroku (4) spájame ω_m spôsobom "každý s každým".

Pre jednoduchšie vyjadrenie mohutnosti q_r vezmime nasledovnú definíciu.

Definícia 2 *Permutácia $q \in S_n$ je typu $(a_1, \ldots, a_n)$, ak má práve a_i cyklov dĺžky i pre každé $i \in \{1, \ldots, n\}$.*

Keď sú permutácie q, r typu $(a_1, \ldots, a_n)$, tak počet prvkov množiny q_r sa vypočíta ako $|q_r| = \prod_{i=1}^{n} a_i! i^{a_i}$. To je v súhlase s [4] a teda podporuje korektnosť algoritmu (3).

Pre väčšiu názornosť ukážme teraz ako algoritmus funguje na konkrétnych permutáciach.

Príklad 1 *Majme permutácie $q = \binom{1234567}{2143675}$ $r = \binom{1234567}{5472136}$, ktoré majú nasledovný cyklový zápis :$q = (12)(34)(567)$ $r = (15)(24)(376)$.*
Množiny cyklov sú teda tvaru $\Gamma_2 = \{(12), (34)\}$, $\Delta_2 = \{(15), (24)\}$ a $\Gamma_3 = (567), \Delta_3 = (376)$. Ukážme si teraz aké nájde algoritmus (2) bijekcie ω_2. Nech $\gamma_1 = (12)$, $\gamma_1 = (34)$ a $\delta_1 = (15), \delta_2 = (24)$. Keďže Γ_2, Δ_2 sú dvojprvkové množiny, volí sa permutácia χ z S_2. Vezmime $\chi = \binom{12}{21}$. Bijekcie $\omega_{\gamma_1,\delta_2}$ sú tvaru $\binom{24}{12}$ alebo $\binom{24}{21}$. Bijekcie $\omega_{\gamma_2,\delta_1}$ sú tvaru $\binom{15}{34}$ alebo $\binom{15}{43}$. Ich spojením vzniknú 4 bijekcie $\binom{2415}{1234}, \binom{2415}{1243}, \binom{2415}{2134}\binom{2415}{2143}$.

Ich prehľad možno vidieť v nasledujúcej tabuľke.

$\omega_{\gamma_1,\delta_2}$	$\omega_{\gamma_2,\delta_1}$	ω_2 pre $\chi = \binom{12}{21}$
$\binom{24}{12}$	$\binom{15}{43}$	$\binom{2415}{1234}$ $\binom{2415}{1243}$
$\binom{24}{21}$	$\binom{15}{34}$	$\binom{2415}{2134}$ $\binom{2415}{2143}$

Pre $\chi = \binom{12}{12}$ dostávame ďalšie 4 bijekcie $\binom{2415}{3412}$,$\binom{2415}{4312}$, $\binom{2415}{3421}$ $\binom{2415}{4321}$. Množina Ω_2 teda obsahuje celkovo 8 bijekcií ω_2

Množina Ω_3 má tvar $\Omega_3 = \{\binom{376}{567}, \binom{376}{675}, \binom{376}{756}\}$. Spojením bijekcií $\omega_3 \in \Omega_3$ a $\omega_2 \in \Omega_2$ spôsobom "každý s každým"dostaneme teda 24 permutácií z q_r.

Vezmime jednu z nich a ukážme, že je obsiahnutá v q_r.

Spojením $\omega_3 = \binom{376}{567}$ a $\omega_2 = \binom{2415}{1234}$ dostávame permutáciu $p = \omega_3 \vee \omega_2 = \binom{3762415}{5671234}$, čo má v štandardnom zápise tvar $p = \binom{1234567}{3152476}$. Ľahko sa overí, že táto permutácia je skutočne z q_r.

3.2 Algoritmus na hľadanie Q_R

Vráťme sa k pôvodnej úlohe a síce, nájsť pre konjugované množiny $Q, R \subseteq S_n$ permutácie $p \in S_n$, také, že $pQp^{-1} = R$. V zhode s predchádzajúcim budeme označovať množinu všetkých takých p ako Q_R. Vezmime ľubovoľnú permutáciu $p \in Q_R$. Vieme, že pre každú permutáciu $q \in Q$ platí $pqp^{-1} \in R$. Ďalej vieme, že konjugované permutácie majú rovnakú cyklovú štruktúru, a teda poznáme, na aké permutácie $r_i \in R$ sa môže q konjugáciou zobraziť. Sú to práve tie, ktoré majú rovnakú cyklovú štruktúru (typ [4]) ako q.

Pomocou algoritmu (3) teda vieme nájsť množinu C_q kandidátov na permutáciu p. Na nájdenie Q_R potom stačí otestovať pre všetky $c \in C$, či je splnená rovnica $cQc^{-1} = R$. Tie, ktoré testom prejdu, tvoria množinu Q_R. Z predchádzajúceho je zrejmé, že množina C_q sa dá nájsť ako zjednotenie $C_q = \cup_i q_{r_i}$, kde permutácie r_i prechádzajú cez všetky permutácie množiny R rovnakého typu ako q.

Dá sa ukázať, že množiny q_{r_i} sú pre rôzne r_i disjunktné a teda mohutnosť c_q množiny $c_q = |C_q|$ sa pre permutáciu q typu $(a_1, \ldots, a_n)$ dá vypočítať ako $c_q = n_q(a_i!)i^{a_i}$ (n_q označuje počet permutácií v R rovnakého typu ako q). Samozrejme mohutnosť C_q závisí na permutácii q, a preto treba voliť q tak, aby bol výraz $n_q(a_i!)i^{a_i}$ čo najmenší. Minimálnu hodnotu c_q budeme označovať ako $c_{Q,R}$, keďže táto zavisí na oboch množinách R, Q.

Poznámka 5 *Pripomeňme, že ak sú množiny R, Q konjugované, potom obsahujú rovnaký počet permutácií daného typu. To predstavuje nutnú podmienku konjugovanosti. V prípade, že Q, R majú rôzny počet permutácií pre nejaký konkrétny typ, tak konjugované nie sú a $C_{Q,R} = 0$.*

Algoritmus 4 *Vstup: Množiny permutácií $Q, R \subseteq S_n$*
Výstup: Množina Q_R.

1. *Rozlož permutácie z Q, R na disjunktné cykly a zisti ich typ.*
2. *Ak majú množiny Q, R rôzny počet permutácií nejakého typu vráť $Q_R = \emptyset$ a ukonči algoritmus.*
3. *Prechodom cez všetky $q \in Q$ nájdi také $\mathbf{q}$ typu $(a_1, \ldots, a_n)$, že výraz $c_{\mathbf{q}} = n_q(a_i!)i^{a_i}$ dosahuje minimálnu hodnotu.*
4. *Pomocou algoritmu (3) nájdi množinu $C_{\mathbf{q}} = \cup_i \mathbf{q}_{r_i}$, kde r_i prechádza cez všetky permutácie z R rovnakého typu ako $\mathbf{q}$.*
5. *V cykle vezmi všetky permutácie $c \in C_{\mathbf{q}}$ a ak pre nejaké c platí $cQc^{-1} = R$ pridaj ho do Q_R.*
6. *Vráť Q_R.*

---

4 Algoritmus zhrnutie

Zhrňme si teraz čo všetko máme k dispozícii. Máme algoritmus (algoritmus 1), ktorý dokáže nájsť dvojice φ_i, ψ_i z ktorých potom možno jednoducho (poznámka 1) nájsť všetky izotopizmy štvorcov L_1, L_2.

Dvojice φ_i, ψ_i sa hľadajú tak, že sa testuje rovnica $pP_{L_1}p_1^{-1}p^{-1} = P_{L_2}p_2^{-1}$ pre všetky permutácie $p \in S_n$ a pre každé $p_2 \in P_{L_2}$. Ďalej máme algoritmus (algoritmus 4), ktorý dokáže pre ľubovoľné Q, R(a teda aj množiny $P_{L_1}p_1^{-1}, P_{L_2}p_2^{-1}$) nájsť minimálnu množinu kandidátov na permutáciu p. Jediné čo zostáva je spojiť algoritmy (1) a (4) a vhodne voliť permutáciu p_1 v algoritme (1). Tá sa volí tak, aby celkový počet B_{p_1} kandidátov na permutáciu p bol pre množinu $P_{L_1}p_1^{-1}$ a všetky $P_{L_2}p_2^{-1}$ čo najmenší.

Vezmime si nejakú permutáciu $q_j \in P_{L_1}$. Tá určuje nejakú množinu $Q_j = P_{L_1}q_j^{-1}$. Označme celkový počet kandidátov, ktorí sa budú musieť prehľadať výberom nášho p symbolom B_{Q_j}. Hodnota B_{Q_j} sa dá vypočítať podľa vzorca $B_{Q_j} = \sum_{i=1}^{n} C_{Q_j,R_i}$, kde množiny R_i sú práve všetky tie, ktoré sú tvaru $P_{L_2}r_i^{-1}$ pre nejaké $r_i \in P_{L_2}$. Celý algoritmus na hľadanie izotopií potom vyzerá nasledovne:

Algoritmus 5 *Vstup:Latinské štvorce L_1, L_2 rádu n*
Výstup:Množina Is dvojíc (φ_i, ψ_i)

1. *Nájdi v cykle množiny $Q_j = P_{L_1}q_j^{-1}$ a množiny $R_i = P_{L_2}r_i^{-1}$ kde q_i prechádza cez všetky permutácie z P_{L_1} a r_i prechádza cez všetky permutácie P_{L_2}.*
2. *Ak je počet množín Q_i nejakého typu rôzny ako počet R_i toho istého typu vráť $Is = \emptyset$ a ukonči algoritmus.*
3. *V cykle pre každé Q_j vypočítaj B_{Q_j} ako $\sum_{i=1}^{n} C_{Q_j,R_i}$.*
4. *Vezmi množinu $\mathbf{Q}$ z množín Q_j, ktorá má najmenšiu hodnotu B_{Q_j}. Ulož si permutáciu $\mathbf{q}$, pomocou ktorej vznikla $\mathbf{Q}$ z P_{L_1} ($\mathbf{Q} = P_{L_1}\mathbf{q}^{-1}$)*
5. *Pomocou algoritmu 4 nájdi množiny $\mathbf{Q}_{R_i}$ pre všetky R_i.*
6. *Pre každú permutáciu $p \in \mathbf{Q}_{R_i}$ pridaj do Is novú dvojicu $\varphi_i = p, \psi_i = \mathbf{q}p^{-1}r_i$.*

Z toho čo sme povedali vyššie je zrejmé, že počet testovaných kandidátov sa vypočíta ako $B_{Q_j} = \sum_{i=1}^{n} C_{Q_j,R_i}$. Akú maximálnu hodnotu môže B_{Q_j} nadobudnúť sa dá odvodiť nasledovným spôsobom. Množiny Q_j, R_i sú pre Latinské štvorce rádu n mohutnosti n. Hodnota C_{Q_j,R_i} sa počíta ako minimum z hodnôt $C_q = n_q(a_i!)i^{a_i}$ kde $q \in Q_j$ je typu $(a_1, \ldots, a_n)$. Dá sa ukázať, že $n_q(a_i!)i^{a_i}$ je pre párne n maximálne $n(n/2)!2^{n/2}$ a pre nepárne $3n((n-1)/2)!2^{(n-1)/2}$. Celková mohutnosť B_{Q_j} teda dosahuje maximálnu hodnotu $n^2(n/2)!2^{n/2}$ prípadne $3n^2((n-1)/2)!2^{(n-1)/2}$ čo určuje aj časovú zložitosť algoritmu.

Táto mohutnosť zároveň zhora ohraničuje počet izotopií medzi Latinskými štvorcami. Výsledná zložitosť algoritmu je potom ohraničená počtom $n^2(n/2)!2^{n/2}$ skladaní permutácií z S_n.

5 Záver

V článku sme popísali algoritmus (algoritmus 5), ktorý hľadá izotopizmy medzi dvoma Latinskými štvorcami. Tento využíva ďalší algoritmus (algoritmus 4), ktorý nájde pre ľubovoľné množiny Q, R množinu Q_R, všetkých takých permutácií $p \in Q_R$, že platí $pQp^{-1} = R$. Odvodili sme tiež počty kandidátov na permutáciu p, ktoré sa v algoritme testujú a ktoré tiež určujú jeho výpočtovú zložitosť. Táto je maximálne $n^2(n/2)!2^{n/2}$ pre n párne a $3n^2((n-1)/2)!2^{(n-1)/2}$ pre nepárne n, čo už pre vyššie hodnoty n dosahuje neporovnateľne menšie hodnoty, ako algoritmus pokusu a omylu popísaný v [6]. Ďalším prínosom článku je posunutie hornej hranice počtu izotopií, ktorá mala doposiaľ hodnotu $n(n!)$ [5] na už spomínanú hodnotu $n^2(n/2)!2^{n/2}$ prípadne $3n^2((n-1)/2)!2^{(n-1)/2}$.

Acknowledgements

This work was supported by Grant VEGA 1/3115/06.

Referencie

1. B. D. McKay and I. M. Wanless, On the number of Latin squares, Annals of Combinatorics, 9 (2005) 335-344
2. Carmichael, R.D. Introduction to the Theory of Groups of Finite Order. New York: Dover, 1956.
3. Mac Lane, S. - Birkhoff, G.: Algebra. Bratislava, Alfa 1973.
4. B na, M. Combinatorics of permutations. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2004.
5. Dénes, A. D. Keedwell, Latin squares and their applications. Academic Press, New York, 1974.
6. Sýs, M., : Isotopy classes of Latin squares. Journal of Electrical Engineering, Vol.58 (2007), No. 7/s, pp. pp 97-100
7. Sýs, M., : On number of isotopisms. Článok bol zaslaný do Mathematica Slovaka.

Acoma: Inteligencia vo vašom mailboxe

Martin Šeleng, Michal Laclavík, Emil Gatial, Zoltán Balogh, Marián Babík, Marek Ciglan and Ladislav Hluchý

Oddelenie paralelného a distribuovaného spracovania informácií, Ústav informatiky, Slovenská akadémia vied
Dúbravská cesta 9, 845 07 Bratislava, Slovensko

Abstrakt. *V príspevku navrhujeme anotáciu e-mailových správ ako nový spôsob využitia predpripravených znalostí pre organizácie, ktoré využívajú e-mailovú komunikáciu ako súčasť svojich procesov. Ďalej v príspevku opisujeme nástroj, ktorý umožňuje poskytovanie znalostí v organizácii pri riešení pracovných postupov (Acoma[1] - Automated Content-based Message Annotator). Ukážeme si tiež možnosť ako prepojiť náš nástroj priamo na pracovný kontext organizácie v prípade, že analyzovaný text je prepojený na špecifickú aplikačnú doménu a existuje ontologický model domény. Na záver predstavíme spôsob ako jednoducho rozširovať možnosti poskytovania znalostí v e-mailoch pomocou technológie OSGI, a tým umožniť ľubovoľné pridávanie modulov do navrhnutého systému.*

1 Úvod

Podľa najnovších prehľadov a štatistík ľudia pracujúci s informáciami posielajú a prijímajú denne 133 e-mailov a strávia 21% pracovného času pri spracovaní e-mailovej komunikácie. Pritom väčšina používateľov hovorí o zavalení informáciami, tzv. „information overload“[2]. Informácie vytvorené v ľubovoľnej organizácii môžu byť prínosom, ale aj záťažou - záleží na tom, ako sú využívané a manažované. E-mail sa v tomto ohľade nijako nelíši od iných informačných zdrojov. Môže byť vysoko efektívnym komunikačných a pracovným nástrojom a zdrojom potrebných informácií, ale iba vtedy, ak sú informácie dobre spravované a manažované.

Jedným zo základných problémov e-mailovej komunikácie je to, že sa používa na účely, na ktoré pôvodne nebola vytvorená. Napríklad na archivovanie informácií alebo manažment pracovných úloh [1, 2]. Ak chcú organizácie dosiahnuť stanovené ciele, základom je efektívna komunikácia, ktorá často prebieha cez e-maily. E-mailová komunikácia sa využíva hlavne pri spolupráci a prepojení (interoperabilite) firiem a organizácií všetkých veľkostí. Je teda vhodným médiom na zistenie kontextu používateľa a poskytovanie relevantných informácií a znalostí v tomto kontexte, ktoré používateľ potrebuje na úspešné vykonanie pracovných aktivít. Podobne ako Gmail[3] zobrazuje kontextovú reklamu a umožňuje niektoré jednoduché akcie, ako je napríklad pridanie udalosti do kalendára, systém Acoma poskytuje informácie a znalosti priamo v kontexte e-mailu. Tieto informácie môžu pomôcť pri práci, ktorú e-mail reprezentuje. Problém ako pripojiť k e-mailu znalosti, resp. kontextové informácie bol riešený vo viacerých projektoch, ako napr. kMail[4] [9], ktorý sa pokúšal zintegrovať e-mailovú komunikáciu s organizačnou pamäťou, ale nútil používateľa používať špeciálneho e-mailového klienta. Iný podobný nástroj je Zimbra[5]. Zimbra je webový e-mailový klient s funkcionalitou poskytujúcou detekciu objektov, ako sú napríklad telefónne čísla alebo adresy komunikujúcich firiem, a umožňuje aj niektoré akcie nad týmito objektmi. Podobne ako kMail aj Zimbra núti používateľa používať špeciálneho e-mailového klienta a e-mailový server a zmeniť tak existujúcu internetovú infraštruktúru v rámci organizácie.

Systém Acoma sa začal vyvíjať ako jeden z komponentov v rámci APVT projektu Raport[6] a ďalej sa bude rozvíjať v rámci medzinárodného projektu Commius[7], ktorý rieši medzipodnikovú interoperabilitu [12] využívajúcu e-mailovú komunikáciu. Jednou z úloh projektu Commius je aj semi-automatická anotácia založená na vzoroch, ktorá stavia na metóde Ontea[8] vyvinutej v rámci projektu NAZOU[9].

Práve sémantická anotácia je jedným zo spôsobov riešenia problémov spojených s e-mailovou komunikáciou. Aby bolo možné zistiť formalizovaný kontext e-mailu napríklad vzhľadom na obchodný model organizácie, je potrebné mapovať text e-mailu na objekty v modeli organizácie.

E-maily podobne ako informácie na webe obsahujú neštruktúrovaný text, často v ešte väčšom množstve. Existujúce anotačné prístupy a riešenia, ktoré sú väčšinou zamerané na dokumenty na webe a využívajú HTML štruktúru, nie je možné použiť na anotáciu e-mailov. Sémantická anotácia však predstavuje možnosť, ako ďalej riešiť odvodzovanie, vyhľadávanie na základe sémantiky alebo zisťovanie kontextu e-mailovej komunikácie.

Príspevok je rozdelený do 5 kapitol: Úvod, Ciele, Prístup a riešenie, Architektúra a riešenie, Záver a budúca práca. V kapitole Ciele opíšeme, aké kroky musíme vykonať, aby sme naplnili požiadavky uvedené v tejto kapitole. V kapitole Prístup a riešenie uvedieme niektoré riešenia vedúce k naplneniu niektorých cieľov. V kapitole Architektúra a technológie uvedieme architektúru nástroja Acoma a v poslednej kapitole Záver a budúca práca uvedieme možnosti rozširovanie funkcionality nástroja Acoma.

2 Ciele

Na splnenie požiadaviek, ktoré vyplynuli z prvej kapitoly, je potrebné poskytovať:

[1] http://acoma.sourceforge.net/

[2] University of Southern California 2007 Center for the Digital Future Report: http://www.digitalcenter.org/pdf/2007-Digital-Future-Report-Press-Release-112906.pdf

[3] http://gmail.com

[4] http://kontact.kde.org/kmail/

[5] http://www.zimbra.com/

[6] http://raport.ui.sav.sk

[7] http://www.commius.eu

[8] http://ontea.sourceforge.net

[9] http://nazou.fiit.stuba.sk

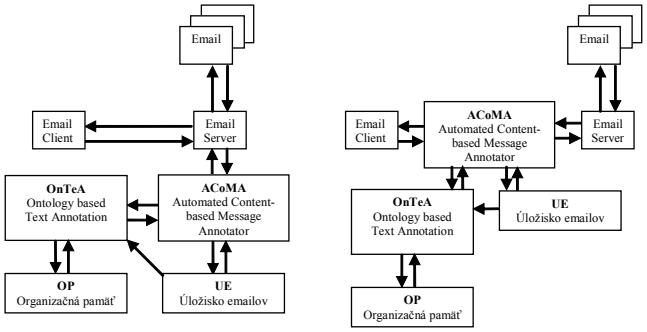
- spoločnú organizačnú pamäť,
- spoločné úložisko (súborové aj ontologické),
- spracovanie e-mailov v danom kontexte pre získavanie znalostí napríklad vo forme textových poznámok s hypertextovými prepojeniami,
- nástroj na semi-automatickú anotáciu (Ontea),
- mechanizmus pre aktualizovanie znalostí v organizačnej pamäti,
- nástroj na dekompozíciu a manažovanie e-mailov (Acoma),
- možnosť pridávať do systému Acoma ďalšie moduly na prispôsobenie sa iným doménam.

3 Prístup a riešenie

V tejto časti sa budeme zaoberať hlavne predposledným a v závere aj posledným bodom z kapitoly Ciele, a to nástrojom Acoma a možnostiam jeho rozšírenia.

E-maily sú silne napojené na prácu v organizácii, ich obsah je však väčšinou neštrukturalizovaný. Vyvinutý nástroj je priamo prepojený na pracovný kontext organizácie, takže nie je ťažké analyzovať a pochopiť kontext týkajúci sa znalostí v organizačnej pamäti.

Používanie e-mailov umožňuje získať „aktívny“ zdieľaný znalostný kanál, pretože používateľ nemusí na získanie určitej znalosti využívať rozsiahle vyhľadávanie. Zdieľané znalosti sú priamo doručené v e-mailovej správe na základe aktuálneho problému alebo aktivity riešenej používateľom. Používateľ dostane e-mail s pripojenými informáciami na konci správy (textové prílohy, resp. HTML prílohy). V e-mailovej správe sa tiež zobrazia informácie o ďalšom probléme alebo aktivite v danom pracovnom procese. Používanie textových príloh sa ukazuje ako vhodné riešenie, pretože sa nemení text pôvodného e-mailu, len sa dopĺňa o relevantné informácie (text, hypertextové prepojenia, a pod.). Momentálne existujú dve implementácie nástroja Acoma: SMTP a POP3 implementácia, ktoré pracujú v nasledovných cykloch (obr. 1).



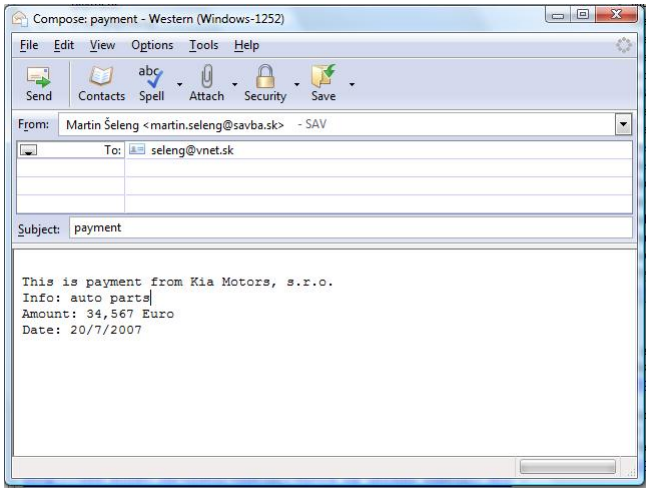
Obr. 1. Cyklus práce nástroja Acoma: SMTP a POP3 implementácia.

V obidvoch implementáciách funguje nástroj Acoma podobne ako proxy s tým rozdielom, že v prípade SMTP implementácie je to nepriame prepojenie (Acoma prijme od SMTP servera naraz celý e-mail) a v prípade POP3 implementácie slúži ako priame proxy (preposiela medzi e-mailovým klientom a e-mailovým serverom celú komunikáciu, až pokiaľ nezachytí samotný text e-mailu).

V prípade SMTP implementácie je nástroj Acoma nainštalovaný na poštovom serveri podobne ako antivírové alebo antispamové programy. Po prijatí e-mailu nástroj Acoma daný e-mail rozloží na prílohy, text e-mailu a hlavičku a následne ich zanalyzuje pomocou sémantickej anotácie [6, 7]. Nástroj Ontea na základe získaného kontextu vyberie z organizačnej pamäte všetky relevantné informácie, ktoré následne pošle späť nástroju Acoma. Acoma tieto informácie naformátuje a pripojí k prijatému e-mailu a e-mail ponechá na serveri. Používateľ následne pri preberaní pošty dostane už takto upravený e-mail.

V prípade POP3 implementácie je Acoma spustená na klientskom počítači. Výhodou tejto implementácie je to, že v prípade viacerých používateľov jedného e-mailového konta (napríklad v prípade malých podnikov, keď sa pre objednávky a faktúry používané len jedno e-mailové konto) systém Acoma nemodifikuje e-mail na poštovom serveri ale len u používateľa, takže ostatní používatelia, ktorí nemajú nainštalovaný nástroj Acoma, vidia pôvodný e-mail bez zmien.

Na nasledujúcom obrázku (obr. 2) je zjednodušený príklad notifikácie o platbe.



Obr. 2. Príklad e-mailu odoslaného používateľom (informácia o zbehnutej platbe)

Nástroj Acoma odosielaný e-mail (obr.2) zanalyzuje, rozloží na hlavičku a telo správy, ktoré následne uloží do úložiska e-mailov. Ontea následne spracuje telo aj hlavičku e-mailu, pričom sa použijú regulárne výrazy, ktoré sa (pre náš prípad) nachádzajú na obr. 3.

Ontea na základe týchto regulárnych výrazov nájde nasledujúce objekty (obr. 4).

Ďalej sú definované jednoduché poznámky s odkazmi na externé dátové zdroje, ktoré musí definovať administrátor. V budúcnosti to však bude umožnené aj bežnému používateľovi cez používateľské rozhranie.

Na základe nájdených objektov sa definované poznámky pridajú do e-mailu ako inline textová príloha (obr. 6), ak sú splnené podmienky vo vlastnosti „MATCH“.

Na nasledujúcom obrázku je zobrazený prijatý e-mail, ktorý obsahuje prepojenia na zdroje (v tomto prípade prepojenia na webové aplikácie). Tieto zdroje súvisia s obsahom e-mailu a podporujú používateľa v ľahšom splnení úlohy, ktorú e-mailová správa reprezentuje.

```
PAYMENT_INFO_PATTERN=[Ii]nfo:
*([^\n,;]+)
PAYMENT_INFO_CLASS=Acoma:PaymentInfo
PAYMENT_AMOUNT_PATTERN=[aA]mount: *([0-
9]+[ ,0-9]+)
PAYMENT_AMOUNT_CLASS=Acoma:PaymentAmoun
t
PAYMENT_DATE_PATTERN=([0-9]{1,2}[./
]+[0-9]{1,2}[./ ]+[0-9]{4})
PAYMENT_DATE_CLASS=Acoma:PaymentDate
ORG1_PATTERN=\s+(\p{Lu}[^\s,]+[
]*[^\s,.]*[ ]*[^\s,.]*)[, ]*(
]*s\.r\.o\.|a\.s\.)[\\s]+
ORG1_CLASS=Acoma:Organization
```

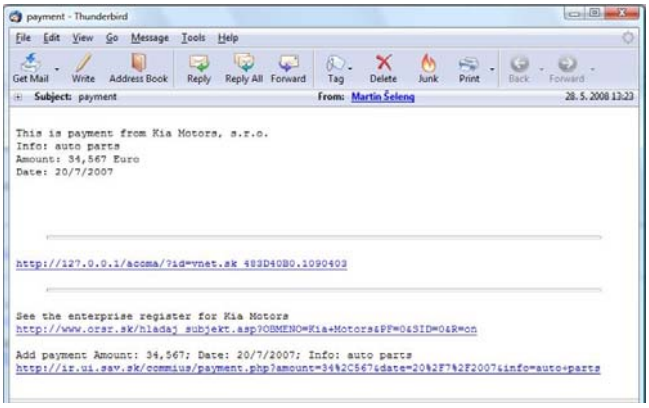
Obr. 3 Ukážka regulárnych výrazov.

```
Acoma:Organization Kia Motors
Acoma:PaymentInfo auto parts
Acoma:PaymentDate 20/7/2007
Acoma:PaymentAmount 34,567
```

Obr. 4 Objekty nájdené nástrojom Ontea v ukážkovom e-maili z obr. 3.

```
COMP_REG_NOTE=See the enterprise
register for {Acoma:Organization}
COMP_REG_URL=
http://www.orsr.sk/hladaj_subjekt.asp?O
BMENO={Acoma:Organization}&PF=0&SID=0
COMP_REG_MATCH=Acoma:Organization
PAYMENT_NOTE=Add payment Amount:
{Acoma:PaymentAmount}; Date:
{Acoma:PaymentDate}; Info:
{Acoma:PaymentInfo}
PAYMENT_URL=http://ir.ui.sav.sk/Acoma/p
ayment.php?amount={Acoma:PaymentAmount}
&date={Acoma:PaymentDate}&info={Acoma:P
aymentInfo}
PAYMENT_MATCH=Acoma:PaymentAmount,
Acoma:PaymentDate, Acoma:PaymentInfo
```

Obr. 5 Objekty a im prislúchajúce textové poznámky.



Obr. 6 Príklad e-mailu upraveného nástrojom Acoma.

4 Architektúra a technológia

V nasledujúcej časti si rozoberieme architektúru a technológiu systému Acoma..

Systém ACOMA sa skladá zo štyroch hlavných častí:

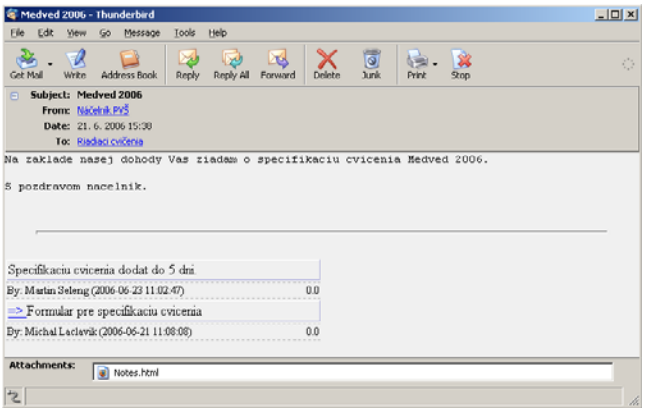
- Acoma Core
- Acoma Server
- Acoma MultiThread
- Acoma Email

Acoma Core slúži len na načítanie konfiguračného súboru so špecifickými nastaveniami pre určitú doménu.

Acoma Server je zodpovedný za počúvanie a prijímanie prichádzajúcich žiadostí o pripojenie so strany poštového klienta (v SMTP aj POP3 implementácii).

Acoma MultiThread je zodpovedný za preposielanie komunikácie medzi poštovým klientom a serverom.

Acoma E-Mail slúži na dekompozíciu prijatého e-mailu, uloženie jednotlivých príloh, spustenie sémantickej anotácie a vytvorenie nového e-mailu s pridanými textovými informáciami. Nástroj Acoma používa na prácu s e-mailovými správami JavaMail API[10]. Na vytvorenie HTML prílohy e-mailu sa použije XSLT[11] transformácia textových poznámok získaných z organizačnej pamäte na HTML dokument, ktorý sa následne pomocou JavaMail API pripojí k už existujúcemu e-mailu (príklad e-mailu s HTML prílohou je zobrazený na nasledujúcom obrázku).



Obr.7 Ukážka prílohy formátovanej ako html

5 Záver a budúca práca

Článok opisuje možnosť využitia znalostí v organizácii tak, aby implementácia ich využitia nezasahovala do zabehnutého pracovného procesu. Pri väčšine projektov manažmentu znalostí sa v organizáciách inštalujú nové systémy, s ktorými sa používateľ musí naučiť pracovať. V prípade nástroja Acoma nie je potrebné inštalovať nové systémy - používateľ dostane relevantné informácie a znalosti priamo pri vybavovaní úloh prostredníctvom e-mailovej komunikácie. Dané informácie môže, ale nemusí využiť, pričom ho neobťažujú v zabehnutých

[10] http://java.sun.com/products/javamail

[11] http://www.w3.org/TR/xslt

pracovných postupoch. Zdá sa, že je vhodné použiť takýto systém všade tam, kde sa elektronická komunikácia používa ako primárny nástroj na manažovanie pracovného procesu.

V ďalšej práci sa budeme snažiť rozšíriť funkcionalitu systému pomocou technológie OSGI, ktorá umožňuje spájanie jednotlivých modulov (bundles) do veľkých aplikácií. V tomto prípade nástroj Acoma bude slúžiť ako kontajner pre jednu z OSGI implementácií, ako napríklad Felix[12], Equinox[13], Oscar[14] alebo Knopflerfish[15]. Moduly vyvíjané ľubovoľnou spoločnosťou môžu byť jednoducho zaradené do systému Acoma, ktorý ich v prípade potreby dokáže stiahnuť z webu, doinštalovať a spustiť nad určitými typmi e-mailov.

Poďakovanie

Táto práca je vyvíjaná a podporovaná projektmi Commius FP7-213876, NAZOU SPVV 1025/2004, SEMCO-WS APVV-0391-06, VEGA 2/7098/27

Referencie

1. S. Whittaker, C. Sidner: Email Overload: Exploring Personal Information Management of Email. In Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems, 276-283.
2. D. Fisher, A.J. Brush, E. Gleave & M.A. Smith: Revisiting Whittaker & Sidner's "email overload" ten years later. In CSCW2006, New York ACM Press.
3. Laclavik, M.: Commius: ISU Via Email, Workshop on Enterprise Interoperability Cluster: Advancing European Research in Enterprise Interoperability II at eChallenges 2007 conference.
4. M. Laclavik, M. Seleng, L. Hluchy: Acoma: Network Enterprise Interoperability and Collaboration using E-mail Communication In Proceedings of eChallenges 2007; Expanding the Knowledge Economy: Issues, Applications, Case Studies Paul Cunningham and Miriam Cunningham (Eds) IOS Press, 2007 Amsterdam ISBN 978-1-58603-801-4, p 1078-1085.
5. M. Šeleng, M. Laclavík, Z. Balogh, L. Hluchý: Automated content-based message annotator - Acoma. In Vojtáš, Peter (editor). ITAT 2006: Information technologies - applications and theory. - Department of Computer Science, Faculty of Science, P.J.Šafárik University, 2006. ISBN 80-969184-4-3, s. 195-198.
6. M. Laclavik, M. Seleng, E. Gatial, Z. Balogh, L. Hluchy: Ontology based Text Annotation – OnTeA; In: Proc. of 16-th European-Japanese Conference on Information Modelling and Knowledge Bases, EJC'2006, Y.Kiyoki et.al. eds., 2006, Dept.of Computer Science, VSB - Technical University of Ostrava, pp. 280-284, ISBN 80-248-1023-9. Trojanovice, Czech Republic.
7. M. Laclavik, M. Ciglan, M. Seleng, S. Krajci: Ontea: Semi-automatic pattern based text annotation empowered with information retrieval methods., In HLUCHY, Ladislav. Tools for acquisition, organisation and presenting of information and knowledge : proceedings in informatics and information technologies. Kosice : Vydavatelstvo STU, Bratislava, 2007. ISBN 978-80-227-2716-7, part 2, P. 119-129.
8. J. Habermas: The Theory of Communicative Action. Beacon, Boston, 1981.
9. D.G. Schwartz, D. Te'eni: Bar-Ilan University, Tying Knowledge to Action with kMail, MAY/JUNE 2000, IEEE Knowledge Management, p 33-39.
10. D. Te'eni, D.G. Schwartz: Contextualization in Computer-Mediated Communication, Information Systems - The Next Generation, L. Brooks and C. Kimble, eds., McGraw-Hill, New York, 1999, pp. 327-338.
11. C.A. O'Reilly, L.R. Pondy: Organisational Communication, Organisational Behavior, S. Kerr, ed., Grid, Columbus, Ohio, 1979, pp. 119-150.
12. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/ebusiness/ei-roadmap-final_en.pdf

[12] http://felix.apache.org

[13] http://www.eclipse.org/equinox

[14] http://oscar.objectweb.org

[15] http://www.knopflerfish.org

Interaktívne vnútroúlohové toky práce

Branislav Šimo, Ondrej Habala, Emil Gatial, and Ladislav Hluchý

Ústav informatiky, Slovenská Akadémia Vied, Dúbravská cesta 9, 845 07 Bratislava, SR
branislav.simo@savba.sk

Abstrakt *Tento článok popisuje nový prístup k interaktívnym tokom práce v gridoch. Modifikáciou existujúceho systému pre manažment aplikácií skladajúcich sa z webových a gridových služieb bol vytvorený nástroj pre interaktívny manažment tokov práce, ktorý umožňuje používateľovi interaktívne manažovať komplexné úlohy skladajúce sa z vykonania viacerých programov. Nástroj využíva mechanizmus pre interaktívnu prácu s aplikáciou bežiacou v gride, ktorý bol vyvinutý v projekte Interactive European Grid. Tento mechanizmus umožňuje predávať v reálnom čase príkazy z používateľského rozhrania bežiaceho na klientskom počítači mananažérovi toku práce bežiacom v rámci gridovej úlohy spustenej v gride. Nástroj poskytuje vizualizáciu vnútorného toku práce úlohy a dáva používateľovi možnosť riadiť vykonávanie úlohy riadením jej toku práce a jeho modifikáciou, ako aj prehliadať čiastočné výsledky vyprodukované jednotlivými podúlohami. Týmto je umožnené nielen modifikovať tok práce s ohľadom na dosiaľ vyprodukované výsledky, rozšíriť ho, alebo naopak skrátiť, ale tiež interaktívne ladiť úlohu bežicu v gride.*

1 Úvod

Súčasné gridové infraštruktúry, akou je napríklad EGEE [7], a systémy pre ich správu, ako napríklad gLite [5], sa sústreďujú na dávkové spracovanie výpočtovo náročných úloh, zvyčajne sekvenčných. Tento model je veľmi vhodný pre širokú škálu aplikácií, kde čas spracovania jednej úlohy nie je až tak podstatný ako čas spracovania celej množiny úloh. Príkladom môžu byť parametrické štúdie. Existujú ale tiež aplikácie vyžadujúce minimalizáciu času potrebného na vykonanie jedinej inštancie. Jedným zo spôsobov dosiahnutia tohto cieľa je paralelizácia výpočtu rozdelením programu na skupinu spolupracujúcich procesov za použitia MPI [6] protokolu pre výmenu správ, ktorými si procesy vymieňajú dáta.

Ďalšou vlastnosťou chýbajúcou v súčasných gridových infraštruktúrach je možnosť interakcie s aplikáciou bežiacou v gride, čo vyplýva zo zamerania gridovej architektúry na aspekt vysokej priepustnosti. Keďže vysoká priepustnosť je už viac-menej vyriešená na produkčnej úrovni, je potrebné venovať sa aj ďalším typom aplikácií.

Vývoj v projekte Interaktívny Európsky Grid (int.eu.grid) [2] sa zameriava na implementáciu týchto dvoch spomínaných vlastností – podpory vnútroklastrových a medziklastrových MPI aplikácií a interaktívnych aplikácií. Nástroje poskytujúce túto funkcionalitu sú popísané v nasledujúcej kapitole.

Kapitola 3 popisuje systém pre interaktívny manažment tokov práce. Tento systém bol vyvinutý ako modifikácia systému pôvodne vyvinutého v projekte K-Wf Grid [8] ako nástroja pre manažovanie aplikácií skladajúcich sa z webových a gridových služieb. Umožňuje používateľom interaktívne a komfortne manažovať zložité úlohy skladajúce sa z viacerých spustení rôznych programov, popísané ako toky práce. Systém používa nástroj pre vytvorenie interaktívneho kanála, vyvinutý v projekt int.eu.grid, ktorý používa na prenášanie príkazov z používateľského rozhrania do manažéra tokov práce bežiaceho v rámci gridovej úlohy na vzdialenom počítači, a tiež na prenos informácií od manažéra tokov práce a jednotlivých programov do používateľského rozhrania. Tento systém je vhodný pre akúkoľvek aplikáciu zloženú z viacerých programov, v ktorej používateľ chce meniť štruktúru toku práce alebo jeho vykonávanie počas jeho behu. Dôvody, pre ktoré by to mohol chcieť, sú rôzne, príkladom môže byť zmena zámeru používateľa po oboznámení sa s čiastočnými výsledkami.

2 Nástroje pre interaktivitu a MPI

Aby mohla aplikácia využívať podporu interaktivity a MPI, musí sa zintegrovať a prispôsobiť niekoľkým komponentom vyvinutým v projekte int.eu.grid. Použitie MPI a interaktivity nie je vzájomne závislé a je teda možné použiť každé jedno nezávisle. Nástroje int.eu.grid-u podporujú aj aplikácie využívajúce buď jednu alebo obe tieto funkcionality.

Pre využívanie MPI je potrebné modifikovať aplikáciu na úrovni zdrojového kódu, aby používala MPI volania, a musí byť linkovaná s MPI knižnicou. Táto modifikácia môže byť v závislosti na aplikácii značne komplikovaná a jej popis je nad rámec tohto článku. Na internete je možné nájsť množstvo príručiek popisujúcich samotné MPI, ako aj spôsoby jeho použitia.

Viac sa budeme venovať popisu konfigurácie interaktívneho kanála a jeho použitia pre interaktívne aplikácie. Interaktívny kanál predstavuje sieťové spojenie, ktoré umožňuje obojsmerný prenos dát medzi aplikáciou bežiacou v gride a klientským používateľským rozhraním. Od aplikácie sa očakáva, že bude výstupy pre klienta zapisovať na svoj štandardný výstup, odkiaľ sú interaktívnym kanálom prenášané ku klientovi, a dáta, ktoré do kanála zapisuje klient, sú napojené na štandardný vstup aplikácie.

V prípade MPI aplikácie sú výstupy všetkých MPI procesov zapisované do kanála v poradí, v akom prichádzajú. Údaje z kanála sú napojené na vstup len jen jedného MPI procesu – mastera – ktorý ich musí v prípade potreby distribuovať ostatným MPI procesom.

Interaktivitu na strane klienta je možné riešiť viacerými spôsobmi, doporučovaným a projektom priamo podporovaným je využitie Migrating Desktop-u. Migrating Desktop (MD) [3], je tučný klient napísaný v jazyku Java slúžiaci ako generický nástroj pre prácu s gridom. Jeho úlohou je nahradiť použitie príkazového riadku používateľsky prívetivejším a zrozumiteľnejším prostredím. Poskytuje všetku základnú funkcionalitu pre prácu s gridom: systém jednorázového prihlásenia sa (angl. single sign-on) za použitia certifikátu používateľa, dátový manažment (prenos súborov medzi pracovnou stanicou a gridom, registrácia súborov do virtuálneho adresára a iných gridových katalógov), manažment úloh (ich spúšťanie, monitorovanie a správu) a vizualizáciu výsledkov. MD je založený na OSGi platforme [9] pre tvorbu programov skladaním rôznych balíkov a zásuvných modulov. Zásuvné moduly hrajú dôležitú úlohu pri podpore aplikácií v MD, lebo umožňujú prispôsobovanie rôznych častí MD pre potreby tej–ktorej aplikácie.

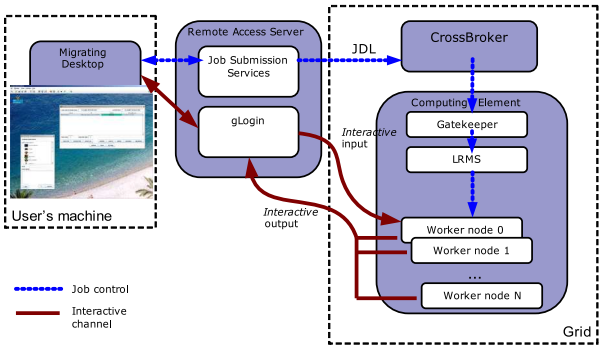
Zásuvné moduly pre vstupy umožňujú aplikačne špecifické zadávanie vstupných parametrov, ktoré môže mať formu jednoduchých textových polí až po komplikované grafické rozhranie. Vizualizačné zásuvné moduly umožňujú implementovať vizualizáciu výstupov aplikácie a v prípade interaktívnej aplikácie slúžia ako používateľské rozhranie pre aplikáciu. Zásuvný modul je potrebné dodať vo forme špeciálneho Java archívu, tzv. balíka (angl. bundle). Balíky sú automaticky načítané pri štarte MD po tom, čo boli zaregistrované do centrálneho registra. Môžu sa nachádzať na rôznych serveroch, podmienkou je dostupnosť cez HTTP protokol. Pre každú interaktívnu aplikáciu je zvyčajne nevyhnutné napísať vizualizačný zásuvný modul na mieru.

Preň je potrebné implementovať tzv. “vizualizačný” zásuvný modul, ktorý poskytuje aplikačne špecifické používateľské rozhranie pre danú aplikáciu. MD poskytuje aplikačné programové rozhranie (API) pre vývojára obsahujúce funkcie pre prácu s interaktívnym kanálom.

Konceptuálna schéma celého systému je zobrazená na obrázku 1.

Pri spúšťaní aplikácie z MD je možné špecifikovať, že sa jedná o interaktívnu úlohu, čo má za následok automatické a pre aplikáciu aj používateľa transparentné vytvorenie interaktívneho kanála, ktorý je následne napojený na vizualizačný zásuvný modul registrovaný pre danú aplikáciu.

Keďže sa MD musí vedieť vysporiadať s firewallmi a nemôže očakávať, že bude priamo dostupný z vonkajšieho sveta, jeho autori sa rozhodli vytvoriť medzičlánok, ktorý bude predávať komunikáciu medzi gridom a MD.



Obr. 1. Interaktívny kanál spájajúci Migrating Desktop s aplikáciou bežiacou v gride.

Nazýva sa Remote Access Server (RAS) a slúži aj na ďalšie úlohy súvisiace prechádzaním firewallmi, napríklad na prenos súborov. Interaktívny kanál medzi RAS serverom a gridovými uzlami sa vytvára pomocou nástroja glogin [4], ktorý používa špeciálnu inicializačnú procedúru a autentifikáciu založenú na certifikátov pre vytvorenie SSH tunela. Kanál medzi MD a RAS je implementovaný ako periodické dotazovanie sa cez HTTP protokol.

3 Interaktívny manažment tokov práce

Automatické vykonávanie toku práce v gridovom prostredí zvyčajne znamená vykonávanie jeho jednotlivých úloh nejakým procesorom, resp. manažérom, toku práce. Z pohľadu používateľa je celý tok práce spracovaný ako jedna veľká úloha, pričom používateľ môže prinajlepšom monitorovať vykonávanie jeho jednotlivých podúloh. V tejto kapitole popisujeme systém pre dynamické vykonávanie toku práce a jeho manažment, ktorý dovoľuje interaktívne monitorovanie a modifikáciu toku práce bežiacom v gride.

Rozdiel medzi klasickými gridovými systémami pre manažment tokov práce a tým popisovaným v tomto článku je, že náš tok práce je predložený gridu (t.j. sprostredkovateľovi zdrojov [15] spravujúcemu vkladanie úloh pre daný grid) ako jedna úloha, ktorá bude spustená na jednom z gridových zdrojov, čo je zvyčajne klaster. Všetky podúlohy toku práce sú následne vykonávané interne v rámci tejto gridovej úlohy. Táto gridová úloha toku práce je napojená na používateľské rozhranie cez interaktívny kanál, ktorý umožňuje používateľovi monitorovať vykonávanie jednotlivých častí toku práce a prípadne modifikovať jeho štruktúru alebo vlastnosti. Výhodou takéhoto vykonávania toku práce je rýchly štart jednotlivých podúloh, keďže tieto nemusia znova prechádzať sprostredkovateľom zdrojov.

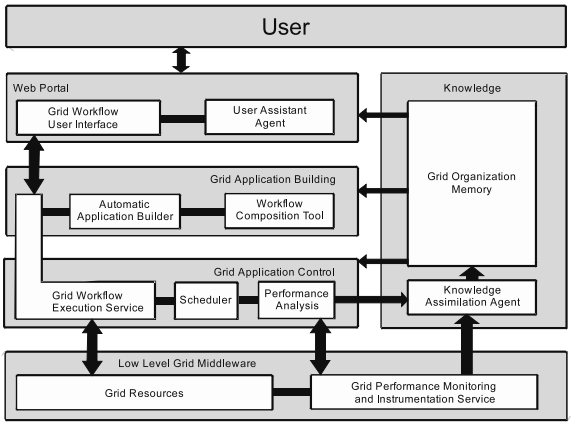
Tento nástroj je vhodný pre aplikácie, kde chce používateľ modifikovať ich vykonávanie za behu s ohľadom na čiastočné výsledky. Ak je to potrebné, môže pridať ďalšie spracovanie čiastočných výsledkov z predchádzajúcich krokov. V prípade, že údaje z medzivýsledkov z nejakého dôvodu prestanú byť relevantné, môže zrušiť časť

toku práce, pričom sa zdroje presunú iným častiam úlohy. Aplikácia, ktorá v súčasnosti používa shell skript volajúci niekoľko komponentov (binárnych modulov alebo iných skriptov), môže byť jednoducho prekonvertovaná na vizuálne a interaktívne manažovateľný tok práce. Tento môže byť uložený, exportovaný do XML súboru a neskôr znova použitý.

V nasledujúcej kapitole 3.1 popisujeme pôvodnú implementáciu manažéra tokov práce, ktorý bol implementovaný v projekte KWf-Grid [8,1] a následne v kapitole 3.2 popisujeme jeho reimplementáciu pre gridové prostredie projektu int.eu.grid.

3.1 Interaktívne toky práce za použitia nástrojov projektu K-Wf Grid

Hlavným komponentom modulu Grid Application Control a jadrom architektúry projektu K-Wf Grid (viď. Obrázok 2) je Grid Workflow Execution Service (GWES) [10,11]. Tento komponent je webovou službou, ktorej hlavnou funkciou je analýza, spracovanie, manažovanie a vykonávanie tokov práce popísaných v jazyku založenom na Petriho sieťach, ktorý sa volá Grid Workflow Description Language (GWorkflowDL) [12].



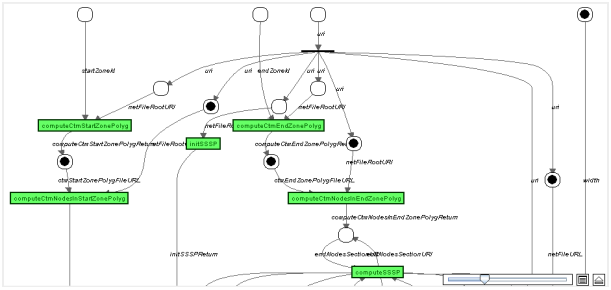
Obr. 2. Architektúra komponentov použitých v projekte K-Wf Grid.

GWorkflowDL je dialektom XML navrhnutým špeciálne pre riadenie tokov práce služieb, programov, gridových úloh alebo dátových prenosov používajúc sémantiku Petriho sietí. Kým najrozšírenejšou abstrakciou tokov práce sú v súčasnosti orientované acyklické grafy (DAG), Petriho siete poskytujú teóriu, ktorá je prinajmenšom porovnateľná s teóriou stojacou za DAG-mi a ktorá umožňuje popísať širšiu škálu konštrukcií, vrátane cyklov a podmieneného vetvenia. Naviac, v Petriho sieťach sú údaje integrálnou časťou celej konštrukcie (reprezentované tzv. žetónmi), a tak GWorkflowDL dokument v akejkoľvek fáze popisuje celý stav systému, čo je veľmi vhodné pre opakovanie experimentov a robenie parametrických štúdií. Je možné nechať tok práce bežať do určitej fázy, uložiť si obraz jeho aktuálnej štruktúry do súboru a potom spustiť viacero rôznych behov so zmenenými parametrami jednoduchou modifikáciou uloženého GWorkflowDL súboru.

GWES bol implementovaný ako webová služba umožňujúca nasledovné operácie s tokom práce: inicializáciu, naštartovanie, pozastavenie a znova rozbehnutie, reštartovanie dokončeného, dotazovanie sa na jeho stav, uloženie do databázy a jeho opätovné načítanie, zisťovanie a nastavovanie špecifických vlastností a iné.

Detailnejší popis všetkých schopností GWES, ako aj kompletný stavový diagram pre toky práce popísané GWorkflowDL, je dostupný v [13].

Jedným z dôležitých nástrojov podporujúcich GWES je Grid Workflow User Interface (GWUI). GWUI je grafické používateľské rozhranie pre GWES, schopné vizualizovať tok práce ním vykonávaný. Použitím GWUI môže používateľ vykonávať väčšinu operácií, ktoré GWES umožňuje, vrátane modifikácie dátových žetónov v Petriho sieti. Príklad vizualizácie je zobrazený na obrázku 3.



Obr. 3. Snímka obrazovky zobrazujúca tok práce vizualizovaný v GWUI.

3.2 GWES v projekte int.eu.grid

V projekte int.eu.grid je infraštruktúra podporujúca GWES ako aj samotný GWES modifikovaný, aby sa dal použiť v gridovej infraštruktúre založenej na systémoch pre správu gridu LCG [14] a gLite [5].

Keďže aplikácie v int.eu.grid-e nie sú založené na službách, ale na paradigme gridových úloh, GWES bol modifikovaný, aby sa stal jadrom vykonateľného modulu, ktorý je vykonaný ako gridová úloha v prjektovej infraštruktúre. Takáto úloha je počas behu interaktívne manažovaná používateľom za použitia GWUI vstavaného do používateľského rozhrania Migrating Desktop (MD).

GWES bol prekonvertovaný na samostatnú Java aplikáciu vykonateľnú z príkazového riadku. Keď sa gridová úloha spustí, prvou spustenou aplikáciu je GWES s parametrom ukazujúcim na GWorkflowDL popis toku práce, ktorý sa má vykonať. Namiesto rozhrania webovej služby GWES komunikuje cez svoj štandardný vstup a výstup, ktorý ja napojený na interaktívny kanál. Na druhom konci

kanála je GWUI reimplementované ako vizualizačný zásuvný modul MD.

Základné schopnosti GWES zostávajú takmer nezmenené. Bol pridaný nový typ úlohy, takže je teraz schopný spúšťať lokálne programy, ktoré sú odkazované aktivitami v GWorkflowDL Petriho sieti. GWUI bol rozšírený o možnosť pridávať, odstraňovať a modifikovať aktivity a miesta. Zostala taktiež zachovaná možnosť editácie údajov.

Komponenty WCT a AAB boli odstránené, keďže tok práce je už pri štarte zkonštruovaný. Plánovač bol nahradený jednoduchším modulom, ktorý je schopný alokovať vykonávaným aktivitám jednotlivé uzly klastra z tých, ktoré boli pridelené celej gridovej úlohe. Toto je teraz internou časťou GWES-u.

Úloha pre vykonanie toku práce je spustená z MD ako špeciálna MPI interaktívna úloha. Počet uzlov žiadaných pre úlohu musí byť rovný alebo väčší než je počet uzlov vyžadovaných ktoroukoľvek jednou podúlohou toku práce, inak by tok práce zlyhal. Alokácia uzlov v rámci úlohy toku práce je vykonávaná podľa parametrov nastavených používateľom v GWorkflowDL dokumente. Ak je niekoľko aktivít pripravených na vykonanie, tie z nich, pre ktoré nie je dostatočný počet voľných uzlov, musia počkať na dokončenie iných aktivít a uvoľnenie príslušných uzlov. Ak GWES narazí pri vykonávaní toku práce na aktivitu vyžadujúcu väčší počet uzlov, než je k dispozícii, signalizuje zlyhanie používateľovi a ukončí vykonávanie toku práce.

4 Záver

Interaktívny manažment tokov práce popísaný v tomto článku dáva používateľovi k dispozícii nový druh riadenia, čo sa týka dynamickej reštrukturalizácie toku práce. Aplikačné komponenty bežiace vo vnútri tohto systému nemajú takmer žiadne štartovacie oneskorenie v porovnaní s klasickými gridovými úlohami, čo je výhodou pre toky práce skladajúce sa z krátko trvajúcich podúloh.

Poďakovanie

Táto práca bola podporená projektami int.eu.grid EU 6FP RI-031857, SEMCO-WS APVV-0391-06, VEGA No. 2/6103/6 a INTAP (RPEU-0029-06).

Referencie

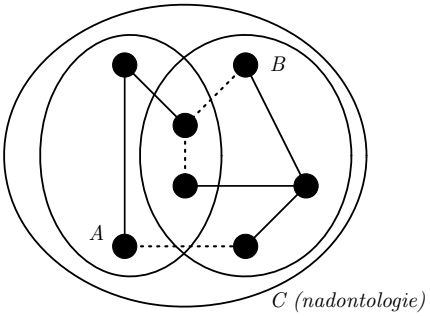
1. Babík, M., Habala, O., Hluchý, L., Laclavík, M.: Semantic Services Grid in Flood-Forecasting Simulations. In: Computing and Informatics. Vol. 26, no. 4 (2007), 447-464
2. Interactive European Grid project, (Accessed January 2008) http://www.interactive-grid.eu
3. Kupczyk, M., Lichwala, R., Meyer, N., Palak, B., Plociennik, M., Wolniewicz, P.: “Applications on demand” as the exploitation of the Migrating Desktop. Future Generation Computer Systems, Volume 21, Issue 1 , 1 January 2005, 37-44
4. Rosmanith, H., Volkert, J.: glogin - Interactive Connectivity for the Grid. In: Juhasz Z., Kacsuk P., Kranzlmüller D.: Distributed and Parallel Systems - Cluster and Grid Computing, Proc. of DAPSYS 2004, 5th Austrian-Hungarian Workshop on Distributed and Parallel Systems, Kluwer Academic Publishers, Budapest, Hungary, September 2004, 3-11
5. gLite - Next generation middleware for grid computing, (Accessed January 2008) http://glite.web.cern.ch/glite
6. Message Passing Interface Forum, (Accessed January 2008) http://www.mpi-forum.org
7. EGEE (Enabling grids for e-science) project, (Accessed January 2008) http://www.eu-egee.org
8. Bubak, M., Fahringer, T., Hluchy, L., Hoheisel, A., Kitowski, J., Unger, S., Viano, G., Votis, K., and K-WfGrid Consortium: K-Wf Grid - Knowledge based Workflow system for Grid Applications. In: Proceedings of the Cracow Grid Workshop 2004, p.39, Academic Computer Centre CYFRONET AGH, ISBN 83-915141-4-5, Poland 2005
9. Equinox - an OSGi framework implementation. (Accessed January 2008) http://www.eclipse.org/equinox
10. Hoheisel, A., Ernst, T., Der, U.: A Framework for Loosely Coupled Applications on Grid Environments. In: Cunha, J.C., Rana, O.F. (eds.) Grid Computing: Software Environments and Tools. 2006. ISBN: 1-85233-998-5.
11. Hoheisel, A.: User Tools and Languages for Graph-based Grid Workflows. In: Special Issue of Concurrency and Computation: Practice and Experience, (c) Wiley, 2005
12. Pohl, H.W.: Grid Workflow Description Language Developer Manual. K-Wf Grid manual, 2006, (Accessed January 2008) http://www.gridworkflow.org/kwfgrid/gworkflowdl/docs/KWF-WP2-FIR-v0.2-GWorkflowDLDeveloperManual.pdf
13. Hoheisel, A., Linden, T.: Grid Workflow Execution Service - User Manual. K-Wf Grid, 2006. (Accessed January 2008) http://www.gridworkflow.org/kwfgrid/gwes/docs/KWF-WP2-D2-FIRST-GWESUserManual.pdf
14. LCG - LHC Computing Grid project, (Accessed January 2008) http://lcg.web.cern.ch/LCG
15. Fernández, E., Heymann, E., Senar, M.A.: Resource Management for Interactive Jobs in a Grid Environment. In: Proc. of IEEE Int. Conf. On Cluster Computing (Cluster 2006), Barcelona (Spain), CD-ROM edition, IEEE CS Press, September, 2006.
16. Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W.: The WS-Resource Framework. March 5, 2004, (Accessed January 2008) http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf
17. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2–13, 2005, (visited January 2008) http://www.globus.org/toolkit
18. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The Physiology of the Grid, (Accessed January 2008) http://www.globus.org/alliance/publications/papers/ogsa.pdf

Kombinace metod pro srovnávání ontologií

Pavel Tyl and Martin Řimnáč

Ústav informatiky AV ČR, v. v. i.
Pod Vodárenskou věží 271/2, 182 07 Praha 8, Česká republika
{rimnacm,tyl}@cs.cas.cz

Abstrakt *Zatímco dílčí ontologie pokrývají jeden pohled na úzce vymezenou oblast, mnohé aplikace však vyžadují obecnější přístup k popisovaným datům. Z tohoto důvodu se přistupuje ke srovnávání ontologií (Ontology Matching), které, pokud je to možné, transformuje několik různých ontologických popisů do jediného. Příspěvek popisuje případovou studii takového procesu za využití různých metod, srovnává jejich úspěšnost a diskutuje možnost využití dílčích výsledků k definici výsledné ontologie. Pro experiment byly nezávisle vytvořeny dvě triviální ontologie, které byly různými nástroji a metodami integrovány do jedné.*



Obr. 1. Původní ontologie zůstávají v nezměněné podobě.

1 Úvod

V různých odvětvích lidských činností bylo vytvořeno mnoho ontologií. Tyto ontologie často obsahují překrývající se koncepty. Například podniky chtějí použít standardní ontologii určité komunity či autority v oboru a zároveň ontologii specifickou pro jejich společnost. Jinými slovy tvůrci ontologií mohou využít již existujících ontologií jako základu pro vytvoření nových ontologií a to integrací existujících ontologií nebo spojováním dílčích ontologií.

Srovnávání ontologií (Ontology Matching) je proces, který ze vstupních ontologií vytvoří *mapování* (Ontology Mapping) mezi páry či skupinami elementů původních ontologií, které jsou nějak významově svázány. Vztahy vzniklé srovnáváním ontologií můžeme využít k provedení následujících operací s ontologiemi:

- *Spojování ontologií* (Ontology Merging), kdy vytváříme ontologii novou, obsahující koncepty z obou zdrojových ontologií.
- *Integrace ontologií* (Ontology Integration) ponechávající zdrojové ontologie v nezměněné podobě a vytvoření nadontologie propojující koncepty dílčí (viz obr. 1).

Jelikož jsou původní ontologie při *spojování ontologií* nahrazeny ontologií novou (bez uvedení přímé korespondence mezi původními a novou)[1], některé dokumenty toto nahrazení nemusí reflektovat a mohou stále ukazovat do původních ontologií. Tím se důvody pro vytváření společné ontologie stírají. Naopak v případě *integrace ontologií* je nadontologie vztahy logicky propojena s původními ontologiemi a v případě, že se nějaký dokument odkazuje na koncept z původní ontologie, je tento koncept propagován do nadontologie. Z tohoto důvodu se v praxi přikláníme spíše k *integraci ontologií*.

Srovnávání ontologií se provádí nejčastěji poloautomaticky nebo ručně, většinou s podporou nějakého grafického uživatelského prostředí. Ruční specifikace částí ontologií pro srovnávání je proces zdlouhavý a navíc náchylný k chybám. Proto je potřeba vyvíjet rychlejší a méně pracné metody, které by zpracovávaly ontologie alespoň poloautomaticky.

Existuje několik nástrojů, které umožňují srovnávání ontologií uživatelem. Tyto nástroje používají rozličné techniky pro návrh integračních pravidel, některé pokročilé nástroje řeší i otázku, jak efektivně výsledky jednotlivých technik kombinovat. Techniky se odvíjejí od úrovně abstrakce, na které pracují.

Nevýhodou některých metod je nutnost nastavení četných parametrů, od nichž se odvíjí návrh integračních pravidel. U mnohých z nich hraje nastavení parametrů tak zásadní roli, že jej nelze provést bez větší znalosti konceptů popsaných v dílčích vstupních ontologiích.

Přestože většina nástrojů určitým způsobem inovuje proces srovnávání ontologií, existuje několik podobných vlastností, které jsou pro tyto nástroje (až na výjimky) společné [4]:

- Propracovanější jsou metody srovnávání na základě struktury oproti metodám založeným na srovnávání instancí.
- Většina nástrojů se zaměřuje na konkrétní aplikační domény (lékařství, hudba…) nebo uvažuje pouze konkrétní typy ontologií (DTD, OWL...). Pouze některé nástroje dokáží zpracovávat ontologie generické

[1] Korespondence mezi ontologiemi může být zachycena jinými nepřímými prostředky.

spolu s doménovými či ontologie v různých formátech. Takovým nástrojem je například COMA++ [2] či S-Match [5].
- Jako vstup většiny nástrojů ke srovnávání ontologií slouží dvojice ontologií. Pokročilejší nástroje jako DCM [6] a Wise-Integrator [7] (automatická integrace dat z webových formulářů) dokáží pracovat s obecnějšími strukturami.
- U nástrojů pro srovnávání převládá stromový přístup k ontologiím. Pouze několik nástrojů pracuje s obecnějšími grafovými strukturami. Jsou jimi například COMA/COMA++ [2] nebo OLA [12] (používá Alignment API [11]).
- Většina nástrojů se zaměřuje na objevování tzv. one-to-one srovnání, přitom je možné narazit na složitější vztahy jako one-to-many či many-to-many. S tou se dokáží vypořádat například DCM [6] (využívá statistických metod a pro naše účely se nehodí) a CTXMatch2 [1].
- Nástroje mohou identifikovat vztah (např. Prompt [8]), pokročilé nástroje ho navíc ohodnotí i mírou spolehlivosti v rozsahu $< 0, 1 >$ (např. COMA++ [2] či CTXMatch2 [1]).
- Nástroje většinou detekují relace ekvivalence, některé z nich počítají další logické vztahy jako je subsumpce atd. Jedním z takových nástrojů je například CTXMatch2 [1].

2 Experiment

K experimentu jsme použili nástroje COMA++ [2], CTXMatch [1] a Alignment API [11]. Pro ukázku automatického návrhu srovnávání dvou použitých ontologií jsme ještě použili nástroj Prompt [8], což je plugin do systému Protégé[2].

Experimenty jsme prováděli s testovacími OWL[3] ontologiemi (MyPerson.owl a MyCustomer.owl), které jsou zobrazeny na obr. 2. Pro testování metod jsme použili ontologie obsahující pouze třídy.

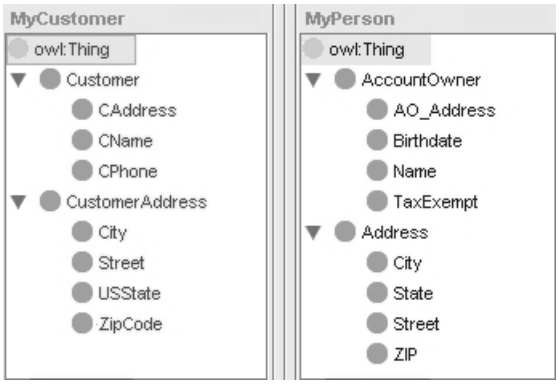
Testovací ontologie byly srovnávány přímo jednotlivými nástroji či aplikačními rozhraními.

Elementy testovacích ontologií byly očíslovány takto:

1: AccountOwner	1: CustomerAddress
2: AO_Address	2: Street
3: Birthdate	3: ZipCode
4: TaxExempt	4: City
5: Name	5: USState
6: Address	6: Customer
7: State	7: CPhone

[2] Protégé – Ontology Editor and Knowledge Acquisition System [online]: <http://protege.stanford.edu>.

[3] Web Ontology Language (OWL) / W3C Semantic Web Activity [online]: <http://www.w3.org/2004/OWL>.



Obr. 2. Testovací ontologie.

8: Street	8: CName
9: City	9: CAddress
10: ZIP	

Následující tabulka reprezentuje vztahy, jež bychom subjektivně očekávali ve výsledku jako „ideální“ za předpokladu, že *vlastníky účtů* považujeme za *zákazníky* nebo skupinu jim odpovídající ($\sim$). Objeví-li se v tabulkách znak $\sqsubset$, znamená v tomto případě relaci subsumpce, v opačném potom zobecnění ($\sqsupset$). Hodnoty v tabulkách poté vyjadřují míru spolehlivosti faktu, že výše zmíněné vztahy odpovídají. Chybějí-li v tabulkách některé sloupce či řádky, neobsahovaly žádná data.

	1	2	3	4	5	6	7	8	9	10
1						$\sim$				
2								$\sim$		
3										$\sim$
4									$\sim$	
5							$\sim$			
6	$\sim$									
7										
8					$\sim$					
9		$\sim$								

2.1 CTXMatch

CTXMatch2.2 [1] využívá sémantický přístup srovnávání. Překládá problém srovnávání na problém logické validity a počítá logické vztahy jako ekvivalence či subsumpce mezi koncepty a vlastnostmi. Je to sekvenční systém, který na úrovni elementů používá databáze WordNet k nalezení počátečních srovnání pro třídy. Na úrovni struktury pomáhají logické odvozovače (např. Pellet) k výpočtu výsledného srovnání pomocí deduktivní techniky a ověřování splnitelnosti výrokových formulí.

Prahová hodnota – Filtrování výsledků srovnávání můžeme provést pomocí nastavení prahové hodnoty (threshold) v rozsahu $< 0, 1 >$. Vztahy ohodnocené nižší hodnotou (v našem případě 0.5) nejsou dále uvažovány (a ani nejsou uvedeny v tabulkách) pro neprůkaznost.

Úloha průchodu ontologií – Průchod ontologie do hloubky (hierarchical task) je u tabulek označen zkráceně slovem „hierarchie“, průchod plošný (flat task) je označen spojením „na plocho“.

Mapování – Mapování **one-to-one** je v popiscích tabulek označeno **1:1**. Mapování **many-to-many** je označeno **M:M**.

Nastavení shodná pro všechny pokusy jsou tato:

- vstupní formát: **OWL**
- výstupní formát: **XML**
- srovnávací metoda: **DL** (využití deskripční logiky pro odvození možných vztahů)

2.2 Alignment API

Alignment API je Java aplikační rozhraní využívající metod založených na zpracování slovních řetězců (String-Based metody). Využívají ho srovnávací nástroje jako OLA [12] či FOAM [3].

Algoritmus Levenshtein – Levenshteinova vzdálenost [9] se definuje jako minimální počet nahrazení, vložení a smazání znaků nutných k převodu jednoho řetězce na druhý.

Algoritmus Smoa – Smoa [9] je míra závislá na délce „běžných“ subřetězců a „neběžných“ subřetězců, kdy se druhá zmíněná část odstraňuje z první části.

Databáze WordNet – Wordnet [13] je vedoucí lingvistickou databází pro anglický jazyk ve světovém měřítku. Seskupuje anglická slova do množiny synonym zvaných *synsety* a poskytuje jejich krátké obecné definice.

2.3 COMA++

COMA/COMA++ [2] je nástroj, který využívá paralelního skládání srovnávacích technik a v grafickém prostředí nabízí i další rozšiřitelné knihovny srovnávacích algoritmů. Doporučená nastavení programu není složité pro určité ontologie upravit tak, aby dávali kvalitnější výsledky. Nejedná se v tomto případě pouze o určení prahových hodnot, ale například i o nastavení souslednosti použitých technik.

2.4 Prompt

Prompt [8] je rozšiřujícím pluginem editoru Protégé. Mimo jiných operací s dvojicemi ontologií (spojovaní, extrakce, porovnání verzí) nabízí i prostředí pro transformaci jedné ontologie na druhou. K tomu využívá srovnávání.

	5	6	7	8	9	10
1		$\sqsupset$ **1.0**	$\sqsupset$ 1.0	$\sqsupset$ 0.56	$\sqsupset$ 0.56	$\sqsupset$ 0.56
2		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ **1.0**	$\sqsupset$ 0.56	$\sqsupset$ 0.56
3		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ 0.56	$\sqsupset$ **0.56**
4		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ 0.56	$\sqsupset$ **1.0**	$\sqsupset$ 0.56
5		$\sqsupset$ 1.0	$\sqsupset$ **1.0**	$\sqsupset$ 0.56	$\sqsupset$ 0.56	$\sqsupset$ 0.56
6			$\sqsupset$ 0.67			
7			$\sqsupset$ 0.67			
8	$\sqsupset$ **0.67**		$\sqsupset$ 0.67			
9		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ 0.56	$\sqsupset$ 0.56	$\sqsupset$ 0.56

Tab. 1. Měření podobnosti CTXMatch – hierarchie – M:M.

	5	6	7	8	9	10
1		$\sqsupset$ **1.0**	$\sqsupset$ 1.0	$\sim$ 0.48	$\sim$ 0.48	$\sim$ 0.48
2		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sqsupset$ **1.0**	$\sim$ 0.4	$\sim$ 0.4
3		$\sim$ 0.62	$\sim$ 0.62	$\sim$ 0.4	$\sim$ 0.4	$\sim$ **0.4**
4		$\sqsupset$ 1.0	$\sqsupset$ 1.0	$\sim$ 0.4	$\sqsupset$ **1.0**	$\sim$ 0.4
5		$\sqsupset$ 1.0	$\sqsupset$ **1.0**	$\sim$ 0.4	$\sim$ 0.4	$\sim$ 0.4
6			$\sim$ 0.53			
7			$\sim$ 0.45			
8	$\sim$ **0.45**		$\sim$ 0.45			
9		$\sqsupset$ 1.0	$\sim$ 0.62	$\sim$ 0.4	$\sim$ 0.4	$\sim$ 0.4

Tab. 2. Měření podobnosti CTXMatch – hierarchie + Semantic Relation – M:M.

	1	4	5	6	7	8	9
1				$\sqsupset$ **1.0**			
2	$\sqsupset$ 0.39						
3					$\sqsupset$ 1.0		
4						$\sqsupset$ 0.56	
5							$\sqsupset$ 0.56
6		$\sqsubset$ 0.39					
7							
8			$\sqsupset$ **0.67**				

Tab. 3. Měření podobnosti CTXMatch – hierarchie – 1:1.

	1	4	5	6	7	8	9
1				$\sqsupset$ **1.0**			
2	$\sim$ 0.31						
3					$\sim$ 0.62		
4						$\sqsupset$ 0.4	
5							$\sqsupset$ 0.4
6		$\sqsubset$ 0.31					
7							
8			$\sqsupset$ **0.45**				

Tab. 4. Měření podobnosti CTXMatch – hierarchie + Semantic Relation – 1:1.

	5	6	7	8	9
1					
2				$\sqsubset\sqsupset\sim$ **1.0**	
3					
4					$\sqsubset\sqsupset\sim$ **1.0**
5					
6					
7			$\sqsupset$ 1.0		
8	$\sqsupset$ **1.0**				

Tab. 5. Měření podobnosti CTXMatch – na plocho – 1:1.

	5	6	7	8	9
1					
2				$Ekviv.\ \mathbf{1.0}$	
3					
4					$Ekviv.\ \mathbf{1.0}$
5					
6					
7			$\sqsupset$ 1.0		
8	$\sqsupset$ **1.0**				

Tab. 6. Měření podobnosti CTXMatch – na plocho + Semantic Relation – 1:1.

	5	6	7	8	9
1		$\sqsupset$ **1.0** $\sim$ **0.7**	$\sqsupset$ 1.0 $\sim$ 0.7		
2			$\sqsupset$ 1.0 $\sim$ 0.7	$\sqsubset\sqsupset\sim$ **1.0**	
3			$\sqsupset$ 1.0 $\sim$ 0.7		
4			$\sqsupset$ 1.0 $\sim$ 0.7		$\sqsubset\sqsupset\sim$ **1.0**
5			$\sqsupset$ **1.0** $\sim$ **0.7**		
6			$\sqsupset$ 1.0 $\sim$ 0.7		
7			$\sqsupset$ 1.0 $\sim$ 0.7		
8	$\sqsupset$ **1.0** $\sim$ **0.7**		$\sqsupset$ 1.0 $\sim$ 0.7		
9		$\sqsupset$ 1.0 $\sim$ 0.7	$\sqsupset$ 1.0 $\sim$ 0.7		

Tab. 7. Měření podobnosti CTXMatch – na plocho – M:M.

	1	2	4	5	6	7	8	9	10
1		$\sim$ **0.53**							
2							$\sim$ **1.0**		
3									$\sim$ **0.43**
4								$\sim$ **1.0**	
5						$\sim$ **0.71**			
6			$\sim$ 0.5						
7	$\sim$ 0.33								
8				$\sim$ **0.8**					
9					$\sim$ 0.87				

Tab. 8. Měření podobnosti Alignment API – Levenshtein.

	5	6	7	8	9	10
1		$\sim$ **0.82**				
2				$\sim$ **1.0**		
3						$\sim$ **0.6**
4					$\sim$ **1.0**	
5			$\sim$ **0.92**			
8	$\sim$ **0.89**					
9		$\sim$ 0.93				

Tab. 9. Měření podobnosti Alignment API – Smoa.

	5	6	7	8	9	10
1		$\sim$ **0.64**				
2				$\sim$ **1.0**		
3						$\sim$ **0.86**
4					$\sim$ **1.0**	
5			$\sim$ **0.83**			
6	$\sim$ 0.33					
7						$\sim$ 0.6
8	$\sim$ **0.94**					
9		$\sim$ 0.97				

Tab. 10. Měření podobnosti Alignment API – WordNet.

	5	6	7	8	9	10
1		$\sim$ **0.69**				
2		$\sim$ 0.77		$\sim$ **0.78**		
5			$\sim$ **0.83**			$\sim$ 0.61

Tab. 11. Měření podobnosti COMA++ – Vlastní nastavení + COMA defaults.

	1	2	3	4	5	6	7	8	9	10
2								$\sim$		
4									$\sim$	
5							$\sim$			
8					$\sim$					
9		$\sim$				$\sim$				

Tab. 12. Měření podobnosti Prompt – Automatické srovnávání.

3 Vyhodnocení experimentu

Mapování vrácené nástrojem CTXMatch při hierarchickém průchodu identifikovalo 6 vztahů z 8, ale jak je patrné z tabulky 1, vedle těchto vztahů jsou se stejnými koeficienty spolehlivosti detekovány další vazby mezi ontologiemi, které neodpovídají skutečnosti. Jinými slovy lze metodu použít spíše pro vážení již detekovaných vztahů nežli pro samotnou detekci.

Jestliže míru podobnosti zkombinujeme s lingvistickou analýzou (Semantic Relation), viz tabulka 2, u většiny chybně vybraných kandidátů dojde k poklesu ohodnocení. Tento pokles je zaznamenán i u dvou nekonfliktních pravidel, zde bez újmy na správnosti výsledku.

V případě, že zvolíme u stejné metody mapování jeden element na jeden element, na základě ohodnocení z tabulek 1 a 2 je proveden výběr kandidátů v tabulce 3, resp. s lingvistickou analýzou v tabulce 4. Tímto výběrem byly zredukovány původně nesprávně označené vztahy, avšak pouze 2 vybrané vztahy odpovídají skutečnosti. Při průchodu na plocho (viz tab. 5) byly správně detekovány 3 vztahy z 8, přičemž došlo k označení pouze 1 chybného vztahu při mapování jeden na jeden. Lexikální analýza (viz tab. 6) nepřinesla do výsledného mapování žádné změny.

Pokud nebudeme uvažovat nutnost výběru mapování jednoho elementu na jeden, jak ukazuje tabulka 7, nejednoznačnost přiřazení elementů je pouze u elementu 7 – *State*.

Srovnáme-li hierarchický průběh a průběh na plocho, výběr kandidátů na plocho je více restriktivní (kandidáti mají menší ohodnocení). Pokud se soustředíme na typ analýzy, metody založené na Levenshteinově vzdálenosti dokáží správně označit 6 z 8 vztahů a 3 chybné (viz tab. 8), metody založené na algoritmu Smoa 6 správných a 1 chybný (viz tab. 9) a metody využívající WordNet 6 správných a 3 chybné (viz tab. 10). Porovnáním výsledků z tab. 1 a výsledků těchto analýz je patrné, že vztahy vybrané těmito analýzami mohou pomoci při řešení nejednoznačnosti výběru např. pomocí nástroje CTXMatch. V neposlední řadě nástroj COMA++ detekoval 5 vztahů z 8 (viz tab. 11)

a lze pozitivně hodnotit, že nedošlo k označení žádného chybného vztahu. Podobně rozšíření Prompt detekovalo pouze 3 ekvivalence (viz tab. 12), opět bez uvedení chybného pravidla. Oba poslední nástroje používají kombinaci metod a z hlediska jistoty vrací korektní mapovací pravidla za cenu, že některá pravidla nejsou detekována vůbec.

4 Závěr

Jak ukazuje vyhodnocení experimentu, nebyl nalezen žádný obecný nástroj, který by plně a bezchybně pokryl celou problematiku integrace ontologií. To nahrává nutnosti použít vícero nástrojů a zkombinovat jejich výsledky.

Z provedené studie vyplývá (za použití daných nástrojů), že je vhodné udělat nejprve hrubý nástin mapování pomocí nástroje CTXMatch, který odfiltruje kandidáty s malou podporou. Nejednoznačnost přiřazení pak může být řešena pomocí různých na textu založených analýz (Wordnet, Levenshteinova vzdálenost), které principielně vždy nemusí vracet správné výsledky. Subsumpce se objevuje velmi zřídka, dokáže ji většinou odhalit až logický odvozovač, který využije informace (např. z WordNetu) o vztahu některých konceptů.

Ukazuje se, že nástroje (COMA++, Prompt) nabízející větší portfolio metod, které kombinují, vracejí přesnější – byť konzervativní výsledky s tím, že některá přípustná mapování vůbec nejsou navržena.

Výsledky srovnávání je proto vhodné ještě validovat proti datům, které původní ontologie používají. I tímto směrem se bude ubírat následující práce.

Poděkování

Práce byla částečně podpořena výzkumným centrem 1M0554 Ministerstva školství, mládeže a tělovýchovy České republiky: Pokročilé sanační technologie a procesy, projektem 1ET100300419 programu Informační společnost (Tématického programu II – Národního programu výzkumu v ČR: Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu) a výzkumným záměrem AV0Z10300504 ”Computer Science for the Information Society: Models, Algorithms, Applications”.

Reference

1. Bouquet, P. – Serafini, L. – Zanobini, S.: Semantic coordination: A new approach and application. In *Proc. 2nd International Semantic Web Conference (ISWC)*, volume 2870 of Lecture Notes in Computer Science, p. 130–145, Sanibel Island (FL US), 2003.
2. Do, H. – Rahm, E.: COMA – A system for Flexible Combination of Schema Matching Approaches. In *Proc. 28th International Conference on Very Large Data Bases (VLDB)*, p. 610–621, Hong Kong (CN), 2002.
3. Ehrig, M. – Sure, Y.: FOAM – Framework for Ontology Alignment and Mapping – Results of the Ontology Alignment Evaluation Initiative. In *Proceedings K-CAP Workshop on Integrating Ontologies*, volume 156, p. 72–76, Banff (CA), 2005.
4. Euzenat J. – Shvaiko P.: *Ontology Matching*. Springer, Berlin/Heidelberg, 2007. ISBN 978-3-540-49611-3.
5. Giunchiglia, F. – Shvaiko, P. – Yatskevich, M.: S-Match: An Algorithm and an Implementation of Semantic Matching. In *Proc. Dagstuhl Seminar*, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl (DE), 2005.
6. He, B. – Chang, K.C.: Automatic complex schema matching across Web query interfaces: A correlation mining approach. *Volume 31 of ACM Transactions on Database Systems (TODS)*, p. 346–395, ACM, New York, 2006.
7. He, H. – Meng, W. – Yu, C. – Wu, Z.: WISE-Integrator: A System for Extracting and Integrating Complex Web Search Interfaces of the Deep Web. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, p. 1314–1317, Trondheim (NO), 2005.
8. Noy, F. N. – Musen, M.: PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th National Conference of Artificial Intelligence (AAAI)*, p. 450–455, Austin (TX US), 2000.
9. Stoilos, G. – Stamou, G. – Kollias, S.: A string metric for ontology alignment. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of Lecture Notes in Computer Science, p. 624–637, Galway (IE), 2005.
10. Straccia, U. – Troncy, R.: oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies. *Volume 3806 of Lecture Notes in Computer Science*, p. 133–147, Springer, Berlin / Heidelberg, 2005.
11. Alignment API and Alignment Server [online]. <`http://alignapi.gforge.inria.fr`>.
12. OLA – Owl Lite Alignment [online]. <`http://www.iro.umontreal.ca/~owlola/alignment.html`>.
13. WordNet – Lexical database [online]. <`http://wordnet.princeton.edu`>.

ITAT'08

Information Technology – Applications and Theory

POSTERS

On approximating the longest monotone paths in edge-ordered graphs

Ján Katrenič

Institute of Computer Science, P.J. Šafárik University, Faculty of Science
Jesenná 5, 041 54 Košice, Slovak Republic
jan.katrenic@upjs.sk

Abstract. *An* edge-ordering *of a graph $G = (V, E)$ is a one-to-one function f from E to a subset of the set of positive integers. A simple path P in G is called an f-*ascent *if f increases along the edge sequence of P. We consider the k-ascent problem (given a graph G and an edge-ordering f and an integer k, either construct an ascent of length k or report that no such ascent exists). We show that for any positive k, the $n^{\frac{1}{k}}$-ascent problem is NP-hard. Moreover, no constant factor approximation algorithm is possible for the longest ascent problem unless P=NP. We also apply a randomized method to solve the ($\log n$)-ascent problem in polynomial time.*

1 Introduction

In this paper we consider finite undirected graphs without loops and multiple edges. A one-to-one mapping f from E to the set of positive integers is called an *edge-ordering* of the graph $G = (V, E)$. A simple path of G, for which f increases along the edge sequence, is called an *f-ascent* of G. An *f-ascent* of a length k will be named a *(k, f)-ascent*. The *height* $h(f)$ of f is the maximum length of an *f-ascent*. Denote the set of all edge-orderings of G by $\mathcal{F}$. A graph theoretical invariant *altitude* of a graph G is defined as $\alpha(G) = \min_{f \in \mathcal{F}} h(f)$. This invariant was studied rather extensively from graph theoretical point of view in connection with graph invariant known as *altitude* of graph [2–4, 10]. Burger et al. [2] presented an algorithm to determine the altitude of the complete graph of order 7 and 8. The value of $\alpha(K_n)$ is still unknown for $n \leq 9$.

In this paper we deal with time complexity problem of finding longest ascent in a given graph and a given edge-ordering. This problem is closely related to the problem of finding value of $h(f)$.

Problem 1. (k-ascent problem)
Let G be a graph. Let f be an edge-ordering of G and k be a positive integer. Is there any (k, f)-ascent in G?

Behind the theoretical importance, the monotone path may appear in many practical situations like the following one. Let the vertices represent businessmen and the edges their mutual communication concerning a given commodity. An owner of the commodity wants to sell it if he obtains more money than he paid for the commodity to the previous owner. Clearly, the trajectory of the commodity forms a monotone path.

2 NP-completeness

The following theorem has been proved in [7].

Theorem 1. *There exists a polynomial time algorithm which from a given graph G produces a new graph G' and an edge-ordering f such that, there exists a simple path of length k in the graph G if and only if there exists a $(2k + 1, f)$-ascent in G'.*

This reduction implies some negative results. Since the decision version of the k-path problem (given a graph G and an integer k, either construct a simple path of k vertices in G or report that no such path exists) is NP-complete, the k-ascent problem is NP-complete as well. This also implies that k-ascent problem is at least as hard as k-path problem. Especially, one can easily prove that for an arbitrary fixed positive $\epsilon < 1$ the problem of finding a (monotone) path of length n^ϵ is NP-hard.

The problem remains NP-complete even if we find a monotone path containing all vertices of the graph [7], what is an corresponding equivalent to the Hamiltonian path problem.

For complete graphs, we also prove NP-completeness of the following problem.

Problem 2. Given an edge-ordering f of a complete graph G and one edge e. Is there any f-ascent containing the edge e and all vertices of G?

3 Approximation

There are several negative results for k-path problem, whose claims for k-ascent problem as well. Karger, Motwani, and Ramkumar [6] proved that, for any $\epsilon < 1$, the problem of finding a path of length $n - n^\epsilon$ in an n-vertex Hamiltonian graph is NP-hard. It claims that no constant factor approximation algorithm is possible for the longest path problem unless P=NP.

In order to prove a similar results for the k-ascent problem, we shall to use a bit stronger formulation of the Theorem 1.

Theorem 2. *There exists a polynomial time algorithm which from a given graph G produces a new graph G' and an edge-ordering f such that, there exists a simple path of*

length k in the graph G if and only if there exists a $(2k+1, f)$-ascent in G'. Moreover, there is a polynomial time algorithm, which is able to find a $(2k+1, f)$-ascent from a k-path and vice versa.

One can show that this result together with the one from [6] directly implies no constant factor approximation algorithm for k ascent problem unless P=NP.

4 Finding k-ascent

First of all, we shall to introduce a polynomial time algorithm for the k-path problem, if $k = \log n/\log\log n$. A naive algorithm that goes over all ordered k-tuples requires more than $\binom{n}{k}$ steps, i.e. $\Omega(n^{\log n/\log\log n})$ running time, a super polynomial time.

An improvement is based on the fact, that the problem can be solved effectively, if the input graph is DAG (directed acyclic graph). A simple method to solve k-ascent problem in a DAG uses topological search and dynamic programming.

The paper [8] suggests using a random permutation of vertices, in order to create DAG. More formally, let π is a permutation of vertices. We create a DAG G' from the graph G, and remove all *backward* edges, i.e. edges v_i, v_j, $i < j$, where $\pi(i) < \pi(j)$. In fact, the permutation π is the topological sorting of G'.

In summary, our algorithm randomly chooses a permutation π, creates a new DAG G', and finds the longest ascent in G'. The last question is, what is the probability of success? Well, consider a longest ascent P of k vertices in G. What is the probability of survival in G'? Clearly, P will appear in G' iff the order of vertices in P is the same as in π. Since π has been chosen randomly, the probability of that is $1/k!$.

Finally, after $k!$ repetitions of the previously described algorithm, we obtain the probability of success at least e^{-1}. Such an algorithm is a Monte Carlo algorithm for the k-ascent problem with time complexity $O^*(k!)$, what is polynomial if $k = \log n/\log\log n$.

In general, the algorithms for k-path problem may not be applicable for the k-ascent problem. On the other hand, using the Theorem 2, an effective algorithm for k-ascent problem could be used to solve k-path problem. We take a look on the k-path problem. Papadimitriou and Yannakakis [9] studied $(\log n)$-path problem and conjectured that it can be solved in polynomial time. This conjecture was confirmed by Alon, Yuster, and Zwick [1], who presented a randomized algorithm of running time $O(2^{O(k)}n^{O(1)})$. Consequently, the complexity upper bounds for these problems were subsequently improved. Jianer, Songjian, Sing-Hoi, and Fenghui [5] presented a randomized divide-and-conquer technique which solves k-path problem in time $O(4^k k^{3.42} m)$ and improved previous best algorithms. This algorithm is based on a recursive function $\mathrm{find\_path}$ (P', G', k), which returns a set of paths, each is a concatenation of a k'-path from P and a k-path in G'. In particular, $\mathrm{find\_path}(\emptyset, G, k)$ returns a set of k-paths in the graph G. Moreover, [5] also presents a derandomization of this algorithm to obtain the final deterministic algotirhm for the k-path problem in the runnting time $O(4^{k+o(k)}m)$.

In the presentation we also show how to adjust this technique for the k-ascent problem in order to solve $(\log n)$-ascent problem in polynomial time.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. Journal of the ACM, **42** (1995) 884–856.
2. Burger, A.P., Cockayne, E.J., Mynhardt, C.M.: Altitude of small complete and complete bipartite graphs. Australas. J. Combin. **31** (2005) 167–177.
3. Calderbank, A.R., Chung, F.R.K. and Sturtevant, D.G.: Increasing sequences with nonzero block sums and increasing paths in edge-ordered graphs. Discrete Math. **50** (1984) 15–28
4. Graham, R.L. and Kleitman, D.J.: Increasing paths in edge ordered graphs. Period. Math. Hungar. **3** (1973) 141–148.
5. Jianer, C., Songjian, L., Sing-Hoi, S., Fenghui, Z.: Improved Algorithms for Path, Matching, and Packing Problems. SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, (2007) 298–307.
6. Karger, D., Motwani, R. and Ramkumar, G.D.S.: On Approximating the Longest Path in a Graph. Algorithmica **18** (1997) 82–98.
7. Katrenič, J., Semanišin, G.: Complexity of Ascent Finding Problem. SOFSEM 2008: Theory and Practice of Computer Science, Volume II-SRF (2008) 70–77.
8. Kortsarts, Y., Rufinus, J.: How (and why) to introduce Monte Carlo randomized algorithms into a basic algorithms course? Journal of Computing Sciences in Colleges, **21** (2005) 195–203.
9. Papadimitriou, C., Yannakakis, M.: On limited nondeterministism and the complexity of the V-C dimension. Journal of Computer and System Sciences, **53** (1996) 161–170.
10. Roditty, Y., Shoham, B. and Yuster, R.: Monotone paths in edge-ordered sparse graphs. Discrete Math. **226** (2001) 411–417.

Language as a complex network

Peter Náther

Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava, Slovakia
nather@ii.fmph.uniba.sk

1 Introduction

Theory of complex networks is in the center of interest of many scientists in the last time. They can be found all around us (social networks, networks of citations, biological or computer networks). It is often useful to use a network as model of the studied system [1, 3, 2]. Complex networks are usually large networks, with apparently random structure. In reality, they have a large-scale architecture organized by some strict principles. Complex networks used to be characterized mostly by following measures:

- $\overline{k}$ - average node degree
- ℓ - average shortest distance between randomly chosen pairs of nodes
- C - average clustering coefficient. It measures the probability that two neighbours of some node are mutual neighbours as well.

Network having a high clustering and low node separation, is called *small world network* [4].

However, the final structure of real networks depends on their development in time. Barabási and Albert have studied growing networks with preferential node attachment. They have shown that the final structure is hierarchical, having a huge amount of nodes with few neighbours, and few nodes having many neighbours [5]. The dynamics of a network reflects in the degree distribution. In the network studied by Barabási and Albert, it has a power law character.

Network having degree distribution following $P(k) \propto k^{-\gamma}$, is called *scale free* [5].

Human language is essential part of the communication between people. Important part of the language is its lexicon. Interesting results were published by Cancho and Solé in [1]. They have studied a positional word web of the English National Corpus. Used relation was the co-occurrence of the words in the sentences. Their network shows to have the character of a small world and a scale-free network.

In the degree distribution of this network, Cancho and Solé have identified two different regimes, with different γ exponents. Dorogovtsev and Mendés [2] are suggesting some theoretical models, also producing networks with two regimes in the degree distribution.

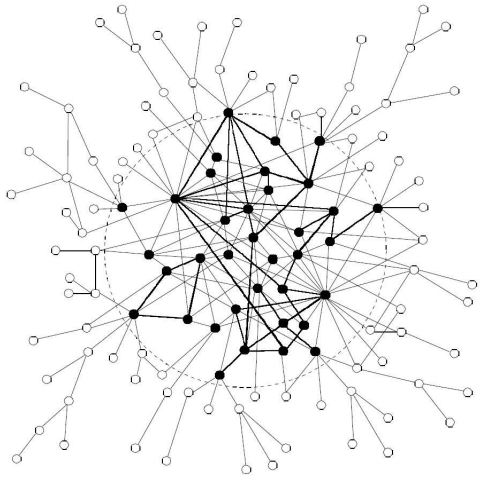


Fig. 1: A possible pattern of wiring in the graph of a human language. Black nodes are common words and white nodes are rare words. Two words are linked if they co-occur significantly.

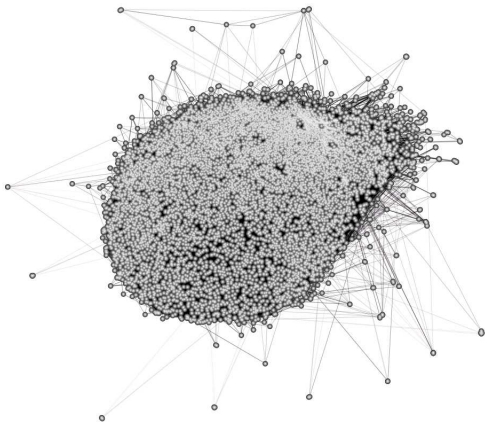


Fig. 2: Small world network of the positional word web. Used text was the King James Version of The Bible. Network is displayed in the Average link layout [12].

Here we will show that the small world characteristic is universal and holds also for other languages. We are also suggesting a model of growing network that should be a better approximation of language networks, than the model of Dorogovtsev and Mendés [2]. We have found the analytical solution of this model and tested it on some experimental networks.

2 Small world

In 1929, the Hungarian writer Frigyes Karinthy [8] speculated that anyone in the world could be connected to anyone else through a chain consisting of no more than five intermediaries. This hypothesis was verified by Stanley Milgram [9] in his popular Milgram Experiment, from which we have a famous six degrees separation phenomenon that says that two people on the planet are connected via chain of only a few intermediate acquaintances.

The small world networks are result of the coexistence of small average distance between nodes and the large clustering coefficient. Usually the average distance ℓ is proportional to the logarithm of the network order. Having a small degree distance is also property of the random graph. However the clustering coefficient in random graph tends to zero, while in a small world network it is close to 1.

Preserving small distance between nodes in highly structured networks is an interesting ability and thus small world networks are widely studied by many scientists. Knowledge from this area can be used in designing of communication networks, social networks, simulations of spreading of the infection diseases, game theory and many others [4].

3 Word web

Positional word web reflects the co-occurrence of words in a sentence. The words (graph sites) are connected by an edge, if they are neighbours in a sentence. In all following cases, the nearest and next nearest neighbours were taken into account. All networks proved to be small worlds.

Cancho and Solé [1] studied a positional network on the basis of the English national corpus. RWN means Restricted word network, UWN means Unrestricted word network. Restricted version means that only word pairs, with the co-occurrence larger than expected by chance were considered.

Graph type	number of words	C	ℓ	$\overline{k}$
RWN	460902	0.437	2.67	70.20
UWN	478773	0.687	2.63	74.13

In [3] was shown that the small world properties are universal for different languages. The positional graph of Slovak lexicon was studied. Because of the morphological complexity of the Slovak language, which have the same root, were creating only one graph site. G_1 denotes the net in which only nearest neighbour interactions in a sentence were taken into account, in G_2 also the next nearest neighbour interactions are included

Graph type	number of words	C	ℓ	$\overline{k}$
G_1	59542	0.369	2.87	29.96
G_2	59542	0.607	2.602	53.01

The positional word web of the several translations of The Bible was also studied [3]. The goal was to study text that consist mainly from the words in the kernel lexicon. EME means Early Modern English, ME means Modern English and SE means Simplified English. ASV is in fact revisited KJV, maintaining its language with some corections. BEV is a special case, written in simplified English language.

version	year	number of words	C	ℓ	$\overline{k}$	language
KJV	1611	11624	0.771	2.18	47	EME
ASV	1901	10105	0.768	2.17	46	EME
NRSV	1989	14985	0.717	2.22	49	ME
BBE	1941	4961	0.765	2.11	59	SE

4 Scale-free network

Scale-free networks were introduced by Barabási and Albert [5]. They have proposed a very simple model, leading to the network with the power-law like degree distribution. The algorithm was following. In each time step:

– **Step 1.:** Add a new node to the graph. Connect it with m edges to the old nodes. Choose the old nodes by preference (here preference is proportional to the degree distribution).

While studying the word web of the English corpus, Cancho and Solé have measured also the degree distribution [1]. The degree distributions of their network have a power law character, but there are two different regimes, with two different exponents.

To explain this, Dorogevtsev and Mendés have proposed following addition to the BA model:

– **Step 2.:** Simultaneously, ct new edges ($c \ll 1$) are created and connect the old nodes with preference [2].

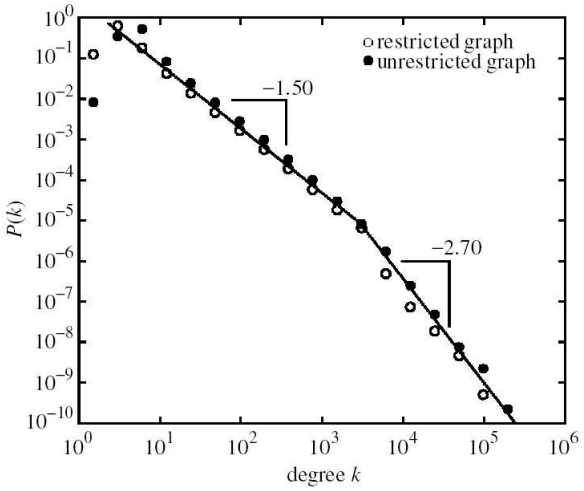


Fig. 3: Degree distribution for the word networks, with two regimes.

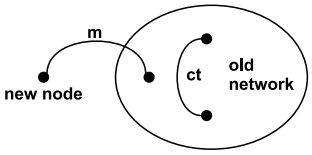


Fig. 4: Scheme of the network growth in the DM model.

Thus for the DM model we have $\overline{k}(s,t) = \left(\frac{t}{s}\right)^{\frac{1}{2}} \left(\frac{2+ct}{2+cs}\right)^{\frac{3}{2}}$, where t is the current time. According [6], if $\overline{k}(s,t) \propto s^{-\beta}$, than $\gamma = 1 + \frac{1}{\beta}$. Than for the DM model we have: for $s \ll t$(well connected words) $\gamma = 3$ and for $s \sim t$ (less connected words) $\gamma = 1.5$ [2]. This model therefore explains the two scaling regimes in the degree distribution function.

Our model is adding another step to the DM model that leads to the lowering of the γ exponent (value measured by Cancho was 2.7).

– **Step 3.:** Simultaneously, m_r old nodes are randomly selected and one edge end of each of them is rewired preferentially.

We have tested our model on two word web networks. On the King James Version of The Bible and the text of four randomly selected books from the Project Gutenberg's site [11]. We have also run the algorithm and generated networks that should approximate these two word webs.

On the contrary with the theory, we can see that all degree distributions show only one scaling regime, with the exponent $\gamma \simeq 1.5$.

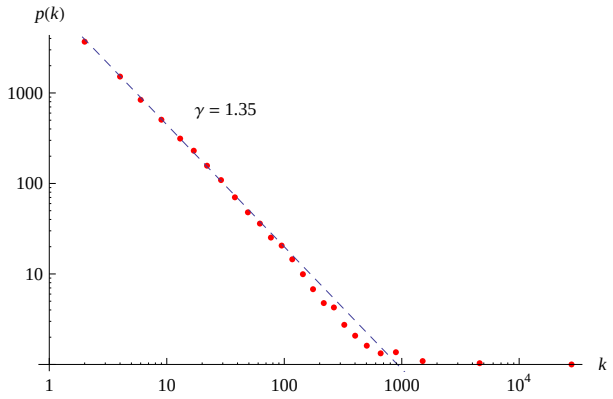


Fig. 5: Degree distribution of The Bible.

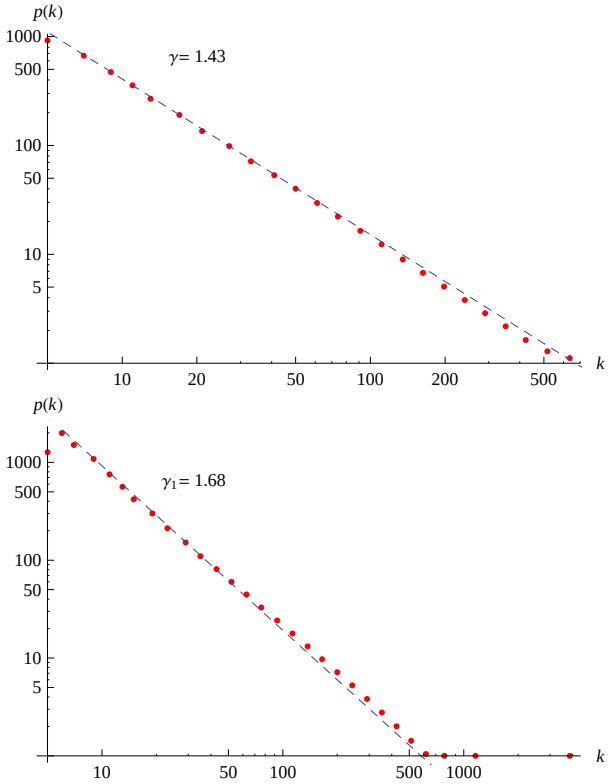


Fig. 6: Degree distributions generated according algorithm and using (2).$N = 11623$, $m = 5$, $c = 0.027$, $m_r = 2$.

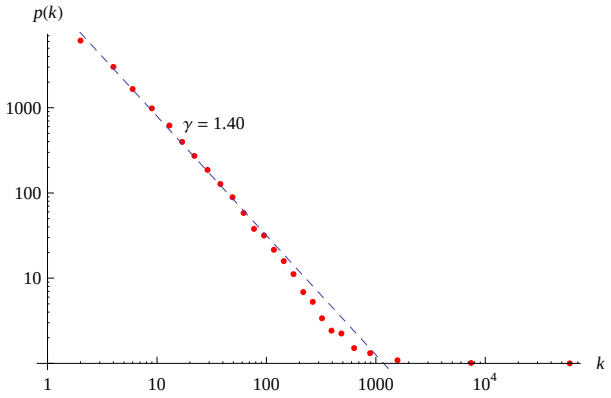


Fig. 7: Degree distribution of text from The Project Gutenberg books.

5 Analytical solution

For the best description of the proposed model, we need to find analytical equation for the degree distribution for this model. If processes creating this model run a long time, continuum approach [6] is good for their description. In this approximation $\overline{k}(s,t)$ is a continuous variable. Parameter s represents the time in which the node was introduced in to the network.

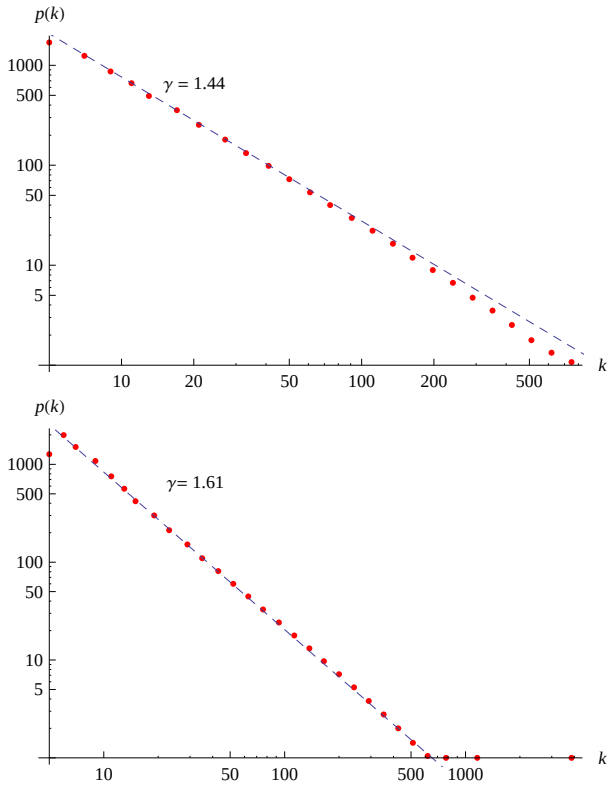


Fig. 8: Degree distribution generated according algorithm and using (2). $N = 20273$, $m = 5$, $c = 0.0041$, $m_r = 1$.

The dynamical equation to be solved is as follows:

$$\frac{\partial \overline{k}(s,t)}{\partial t} = (m + 2ct + m_r)\frac{\overline{k}(s,t)}{\int_0^t k(s,t)ds} - \frac{m_r}{t} \qquad (1)$$

This equation has a following solution:

$$\overline{k}(s,t) = \left(\frac{t}{s}\right)^{\frac{m+m_r}{2m}} \left(\frac{2m+ct}{2m+cs}\right)^{2-\frac{m+m_r}{2m}} g(s,t), \qquad (2)$$

where $g(s,t)$ is a function for which we can prove that $\lim_{t\to\infty} g(s,t) = const$.

Values of exponents using (2)

	$s \ll t$	$s \sim t$
β	$\frac{m+m_r}{2m}$	$2 - \frac{m+m_r}{2m} + \frac{m+m_r}{2m} = 2$
γ	$2 + \frac{m-m_r}{m+m_r}$	1.5

In this new model (2), scaling exponent γ is lower than the value in the model of Dorogovtsev and Mendés in the region of great k. This is exactly what has been measured by Cancho and Solé [1]. We hope that this model could explain their results [7].

6 Conclusions

We have verified results of Cancho and Solé [1]. Furthermore, on the text in Slovak language we have shown that the small-world characteristic of the positional word web is universal also for other languages.

We have also introduced a model of the growing network with the rewiring of old links. We hoped that this model could explain the difference between results of Cancho and Solé [1] and the parameters of the model proposed by Dorogovtsev and Mendés [2]. However our theoretical results does not correspond to the values measured on the texts of The Bible or books from The Project Gutenberg's site. As we have found later, our dataset was too small and the k_{cross} point [2], where exponent γ_1 changes to γ_2 was too large. Thus the two regimes were not present in measured distributions. Our next step should be to verify our results on some dataset, where the k_{cross} point is remarkable.

References

1. Cancho, R. F., Solé, R.: The Small World of Human Language, Proc. Royal Soc. London B *268* (2001) 2261 -2265
2. Dorogovtsev, S. N., Mendes, J. F. F., Language as an Evolving Word Web, Proc. Royal Soc. London B *268* (2001) 2603 - 2606
3. Markošová, M., Náther. P., Language as a Graph (in Slovak), in Mind, Intelligence and Life, STU Bratislava (2007) 298 - 307
4. Watts, D. J., Small Worlds, Princeton University Press, Princeton (2004)
5. Barabási, A. L., Albert, R., Emergence of Scaling in Random Network, Science *286* (1999) 509 - 512
6. Dorogovtsev, S. N., Mendes, J. F. F., Scaling Properties of Scale - Free Evolving Networks: Continuous Approach, Phys. Rev. *E63* (2001) 56125 - 56146, cond - mat/001009 v1 (2000)
7. Markošová, M., Network model of human language, Physica A *387* (2008) 661 - 666
8. Karinthy F., Chains. Everything is different, Budapest (1929)
9. Milgram S., The small world problem, Psychology today *2* (1967) 60 - 67
10. http://unbound.biola.edu
11. http://www.gutenberg.org
12. NWB Team, Network Workbench Tool, Indiana University and Northeastern University (2006) http://nwb.slis.indiana.edu

Požadavky na webové aplikace pro volby přes Internet

Radek Šilhavý, Petr Šilhavý, Zdenka Prokopová
Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně
Nad Stráněmi 4511, 75501 Zlín, Česká Republika
rsilhavy@fai.utb.cz

Výzkumem elektronizace volebních systémů se dnes zabývá celá řada zemí. Každé z řešení elektronických voleb má za cíl být pevnou součástí přímé demokracie a systému elektronické vlády, e-govrnmentu. Systémy elektronického hlasování lze dnes rozdělit v intervalu od čtecích zařízení na klasické papírové hlasovací lístky až po systémy podporující přímé volby prostřednictvím webových technologií. Tou poslední možností se však dnes zabývá pouze velmi malé procento zemí.

Problematika elektronických hlasovacích systémů je podporována velkou popularitou moderních technologií a zejména popularitou Internetu. Dalším důvodem je stoupající množství volebních klání a referend během roku. Cílem výzkumu v netechnické oblasti je tedy hledání systému vzdáleného hlasování, které pomohou tyto problémy řešit. V Evropě již od poloviny devadesátých let klesá účast u volebních klání. Malý zvrat je možné pozorovat u zemí, kde zavedli hlasování poštou, avšak tento není nijak významný. Ostatní možnosti vzdáleného hlasování jsou elektronická pošta či fax. Tyto možnosti jsou občas ve světě využívány.

Cílem výzkumu v oblasti aplikace webových aplikací jako nástroje pro volby přes Internet je tedy výzkum a vývoj takové aplikace, která přinese zlepšení možnosti vzdáleného hlasování. Cílem tohoto příspěvku je provést diskusi základních podmínek, které musí takový webový volební systém splňovat. Lze je shrnout do pěti základních oblastí: Volí pouze oprávnění voliči, každý hlas je započítán pouze jednou, každý volič volí osobně, hlasování je anononymní a bezpečné a volební hlasovací schránka je bezpečná.

Ve volbách je účast povolena pouze oprávněným voličům. Tato podmínka předpokládá existenci elektronického volebního seznamu, který lze využít přímo pro kontrolu oprávněnosti nebo alespoň pro generování přístupových údajů pro voliče. Voliči také musí volit pouze jednou. Což znamená, že do výsledků voleb se musí započítat hlas jednoho voliče pouze jednou. Nemusí se nutně jednat o jeho hlas první, protože možnost opakovaného hlasování je zase důležitá pro dosažení zajištění osobní volby v soukromí. Zatímco u klasických voleb je volič chráněn tím, že hlas upravuje samostatně v odděleném prostoru, u vzdálených elektronických voleb přes internet není zajištění této podmínky bez možnosti opakované volby možné.

Další skupinou požadavků je bezpečnost hlasu. Vhodný volební systém je realizován ve dvou oddělených částech. První bude zodpovědná za autorizaci voliče a druhá část pro uložení hlasů do elektronické volební schránky. Tímto bude dosažena anonymita hlasování. S anonymitou souvisí také ověřitelnost hlasování. Tohoto požadavku lze dosáhnout tak, že každý hlas bude mít jednoznačný identifikátor, který nebude spojen s fyzickým voličem, ale bude pouze jemu znám. Po uzavření voleb lze ve veřejném seznamu hlasů jednoduše ověřit jak byl hlas započítán.

Elektronická hlasovací schránka bude mít formu databáze a všechny uložené hlasy budou chráněné formou PKI. Veřejný klíč je dostupný všem voličům a slouží k šifrování hlasů. Privátní klíč je pak rozdělen mezi členy volební komise a slouží pro dešifrování všech hlasů před zpracováním hlasů.

Tyto základní požadavky slouží pro definování základního přístup k návrhu webové aplikace, která se využívá pro hlasování přes internet. Výhodou webové aplikace jsou dány její podstatou. Je možné centralizovat na jednom místě a obsah personalizovat pro voliče, dle jeho příslušnosti.

Cílem dalšího výzkumu je zejména zpracování standardu, metodického rámce a dořešení bezpečnostních otázek. Zejména otázek spojených s útokem na volební aplikaci formou zablokování služby.

Reference

1. Alexander Prosser, Robert Krimmer (Eds.): Electronic Voting in Europe - Technology, Law, Politics and Society, Workshop of the ESF TED Programme together with GI and OCG, July, 7th-9th, 2004, in Schloß Hofen / Bregenz, Lake of Constance, Austria, Proceedings. GI 2004, ISBN 3-88579-376-8
2. Leenes, R., Svensson, K.: Adapting E-voting in Europe: Context matters.Proceedings of EGPA, 2002.
3. Commission on Electronic Voting: Secrecy, Accuracy and Testing of the Chosen Electronic Voting System. Dublin, 2006, available at http://www.cev.ie/htm/report/download_second.htm accessed on 2007-06-06.
4. Chevallier, M.: Internet voting: Status; perspectives and Issues, ITU E-Government Workshop, Geneva, 6 June 2003, available at: http://www.geneve.ch/chancellerie/EGovernment/doc/UIT_6_6_03_web.ppt accessed on 2007-06-06.

Categorical approach to database modeling

David Toth

Czech Technical University in Prague, Faculty of Electrical Engineering
Department of Computer Science and Engineering
tothd1@fel.cvut.cz

Abstract. *This very short article describes main ideas of category theory approach to database models—their description and comparison. Category theory lets us see database models from new perspective, lets us compare properties and possibilities of considered database models. It is almost impossible to envision theoretical comparison without using one unified language, conceptual framework, which category theory embodies, in this case. We deal with pure theoretical aspects of database formalisms such as data models and query languages, even with update facilities. Pervasive empirical reason for this theoretical study comprises pure practical question of which database technology is appropriate in particular software project. Another interesting motivation is question if it is possible to determine one database technology as better than some other; and furthermore to define what that better should mean.*

1 Introduction

This article should be grasped as an extended abstract. We will try to explain main idea of applying mathematical category theory to the database modeling.

There are three major database technologies or approaches: relational, object and XML. Furthermore we can see some kind of similarity between older models:

network↔object and hierarchical↔XML.

Therefore these "older" models will not be considered later anymore.

Today many algorithms of mapping between different database paradigms are well known and described. ORM means Object-Relational Mapping and it is heavily used by object-oriented programmers today. More and more ORM frameworks such as Hibernate or TopLink, to cite a few, are built and used [7]. OXM, XOM respectively (Object-XML, XML-Object respectively) Mapping is implemented e.g. in the technology JAXB (which stands for Java API for XML Binding) of Sun corp. [8].

But still here arises the question of theoretical equivalence or isomorphism of possible and appropriate data models. In other words we can ask about which data model is better than other model in particular situation. Category theory helps us define what this "better" actually means [6].

2 XML data models

Several formalims, i.e. models and languages, for XML databases were proposed; SAL (Semistructured Algebra), XML Algebra (based on trees), HNR (Heterogeneous Nested Relations) and others. Are they "essentially the same" [5]?

We were evolved several categorical models such as category of graphs, trees, labeled trees, labeled graphs and hereditary finite sets (HFS), beside others. Surprisingly tree-based models have not properties we would expect. Concretely they are not Cartesian Closed [1]. This property looks like to be very important because it is strong property which some models have. And stronger means more control. The category of HFS seems to be the only appropriate formalism from the above mentioned.

We develop encoding of XML into the modified HFS and back. We call this category as XHFS (from XML-HFS). Several functorial approaches are currently under development.

3 Conclusions, work in progress and future works

Categorical approach can help to describe data models, properties of these models, languages operating on these models. We can even use it for semantics.

After categorical models for object-oriented databa-ses were introduced [4], category theory was used for the purposes of semantic web modeling [3]. Several categorical models for XML database were introduced [6]. Further description will be found in future works. The way we look forward covers cartesian closedness, functors and natural transformations.

It seems to be inevitable that many models of object databases such as monoidal, semi-monad approach and others have close interrelationships. These associations needed to be discovered and described.

Relational data model has its own well founded and described data model, i.e. relational data model. We plan describe new associations of category of relations and other categories of other models.

It seems to be very promising to use the idea of category of data models as an analogy to category of categories [2]. This principle should helps us to better understand data models and their properties.

References

1. M. Barr and C. Wells. *Category Theory for Computing Science*. International Series in Computer Science. Prentice-Hall, 1990. Second edition, 1995.
2. R. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. Cambridge University Press, 1993.
3. J. Güttner. *Object Databases and the Semantic Web*. PhD thesis, 2004.
4. P. Kolenčík. *Categorical Framework for Object-Oriented Database Model*. PhD thesis, 1998.
5. J. Lambek and P. J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, March 1988.
6. D. Toth. Database engineering from the category theory viewpoint. In J. Pokorný, V. Snášel, and K. Richta, editors, *DATESO*, CEUR Workshop Proceedings. CEUR-WS.org, 2008.
7. D. Toth and P. Loupal. Metrics analysis for relevant database technology selection. In *Objekty*, 2007.
8. D. Toth and M. Valenta. Using Object And Object-Oriented Technologies for XML-native Database Systems. In J. Pokorný, V. Snášel, and K. Richta, editors, *DATESO*, CEUR Workshop Proceedings. CEUR-WS.org, 2006.

Dependence of variables identification with polynomial neural network

Ladislav Zjavka
Department of Informatics, Faculty of Management Science and Informatics, University of Žilina
Univerzitná 8215/1, 010 01 Žilina

Abstract. *Polynomial neural network creates a structural model of system, which is described only with small input-output data samples. The output of neuron is represented as a polynomial function of inputs. Identification dependencies of variables could be used to model any system with not proportionally changed related values of variables. Polynomial neural network creates a multi-parametric function, which output is based only on relations of inputs. Non linear periodic activation function transforms the dependent increased variables of input vector into proportionally changed related values.*

1 Introduction

Artificial neural networks identify in general patterns according to their relationship. But input vector often contains unknown dependencies of variables. Identification of this would to form functional output of the network as a **generalization of input patterns**. Neural network would be able to identify correctly all patterns, which behave this learned dependence.

General connection between input and output variables is expressed by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial :

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \dots \quad (1)$$

m - number of variables
X(x_1, x_2, ... , x_m) - vector of input variables
A(a_1, a_2, ... , a_m), ... - vectors of parameters

This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or system of Gaussian normal equations [1].

2 Polynomial GMDH neuron

In 1961 Frank Rosenblatt had identified the key weakness of neurocomputing as the lack of means for effectively selecting structure and weights of the hidden layer(s) of the perceptron. In 1968, when backpropagation technique was not known yet, a technique called Group Method of Data Handling (GMDH) was developed by a Ukrainian scientist Aleksey Ivakhnenko who was working at that time on a better prediction of fish population in rivers. He attempted to resemble the Kolmogorov-Gabor polynomial (1) by using low order polynomials for every pair of the input values [4] :

$$y' = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \quad (2)$$

Ivakhnenko made the neuron a more complex unit featuring a polynomial transfer function. The interconnections between layers of neurons were simplified, and an automatic algorithm for structure design and weight adjustment was developed. The GMDH neuron (Fig.1) has two inputs and its output is a quadratic combination of 2 inputs (total 6 weights). Thus GMDH network (Fig.2) builds up a polynomial (actually, a *multinomial*) combination of the input components. Typical GMDH network maps a vector input **x** to a scalar output *y'*, which is an estimate of the true function $f(\mathbf{x}) = y$. The output y' can be expressed as a polynomial of degree $2(K\text{-}1)$, where K is the total number of layers in the network [5].

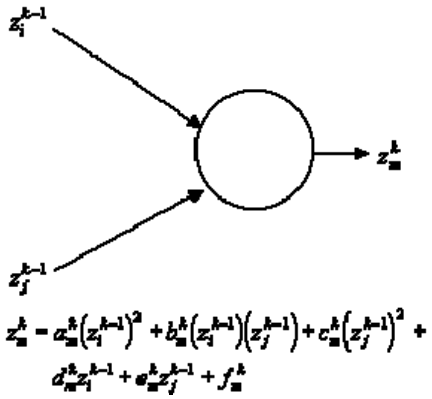


Fig. 1. Polynomial GMDH neuron.

3 Polynomial GMDH neural network

The basic idea of GMDH adjustment is that each neuron wants to produce y at its output (i.e., the overall desired output of the network). In other words, each neuron of the polynomial network fits its output to the desired value y for each input vector **x** from the training set. The manner in which this approximation is accomplished is through the use of linear regression [4].

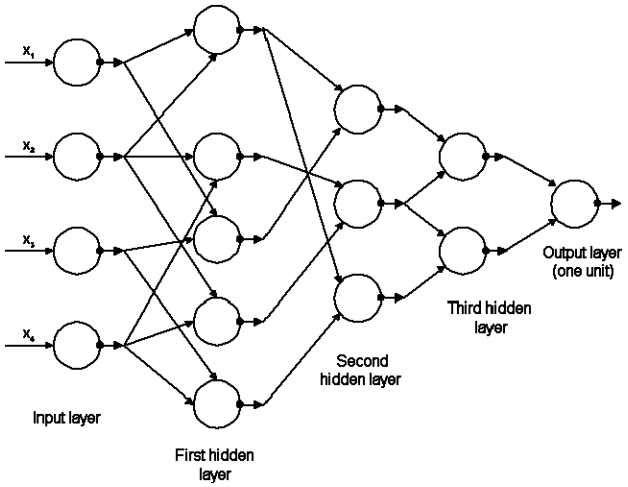


Fig. 2. Polynomial GMDH neural network..

The GMDH network is developed by starting at the input layer and growing the network progressively towards the output layer, one layer at a time. Each next layer *k* starts with maximum possible number of neurons (which is a number of combinations $C(M_{k-1}, 2)$, where M_{k-1} = number of neurons in previous layer *k-1*), *adjusted* by **trimmimg of extraneous neurons,** determining weights, and then frozen. This is different from the backpropagation / counterpropagation technique where all of the layers may participate simultaneously in the training process [5].

The process of building the network continues layer by layer until a stopping criterion is satisfied. Usually, the mean square error of the best performing neuron is lower with each subsequent layer until an absolute minimum is reached. If further layers are added, the error of best performing neuron actually rises. The GMDH are inductive self-organizing networks, which are capable of organizing themselves in response to some features of the data. Polynomial neural networks create a model of system with unknown interrelationships and dependencies of variables [2].

4 Identification dependence of variables

Consider 2 input variables, which behave functional dependence – their sum is constant (for example 20). Polynomial neural network will learn this relation by training data set by means of genetic algorithm. There it is necessary only 1 hidden layer of polynomial neurons with output :

$$y = a_0 + a_1x_1 + a_2x_2 + a_3\,x_1x_2 \tag{3}$$

Right input (of trained dependence) is indicated by neural network for example as output “1”. The mean square error takes on a minimal value :

$$E = \frac{1}{M}\sum_{i=1}^{M}\left(y_i^d - y_i\right)^2 \rightarrow \min \tag{4}$$

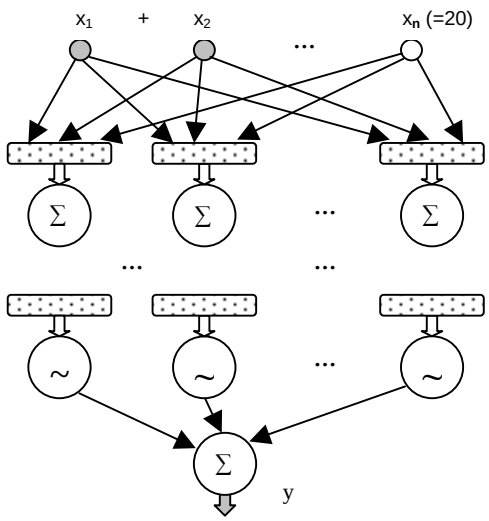


Fig. 3. Polynomial neural network for dependence of variables identification.

Input variables	Training data	Testing data
x_1	1 16 3 8 10	18 5 11 13 6
x_2	19 4 17 12 10	2 15 9 7 14

Tab. 1. Sum of input variables is constant (=20). Their values are changing proportionally.

5 Transformation of dependent variables

But if the values of input variables change not proportionally (they can differ enormously), neural network will not correctly evaluate all input patterns. Consider for example 2 input variables again, which difference is constant (=5).

Input variables	Training data	Testing data
x_1	7 16 30 100 190	10 21 59 155 1000
x_2	2 11 25 95 185	5 16 54 150 995

Tab. 2. Difference of input variables is constant (=5). Their values are growing.

You can see, that inputs {7, 2} and {1000, 995} polynomials cannot evaluate as the same. The multiplications (combinations) will linearise with root-square function (if the input values are greater than 1). The proportion of neuron outputs, which inputs behave the trained dependence, will be after polynomial parameters adjustment corresponding (to all input patterns). Addition of each next layer should make better reciprocal proportion of neuron outputs. Then it is used a periodic activation function (cosinus) - in the last hidden layer of the network (Fig.3.). It transforms the results of polynomials into proportional values, which functional dependence of inputs will be kept. Period ω of the cosinus-function (5) must change its value according to the bigness of inputs. Another way could be used instead of periodic function a rational quotient function on each pair of neurons. This type of neural network ought to approximate a differential equation (7), which can describe a system with dependent variables.

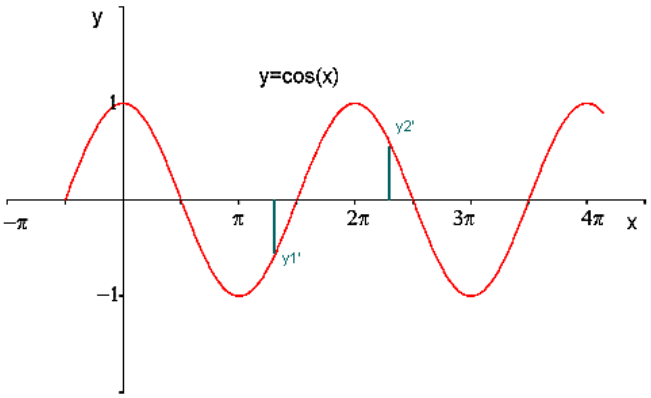


Fig. 4. Transformation of dependent input variables with periodic function.

All input variables will transform by following equations (5, 6). Any new output of polynomials (formed for 2, 3 or 4 inputs) must be so independent to all other outputs as possible.

$$y_1' = \cos(\omega_1 (a_0 + a_2 x_2 + a_3 x_1 x_2)^{\frac{1}{2}} + \phi_1) \quad (5)$$

$$y_2' = \cos(\omega_2 (a_0 + a_1 x_1 + a_3 x_3 + a_4 x_1 x_2 + a_5 x_2 x_3 + a_7 x_1 x_2 x_3)^{\frac{1}{3}} + \phi_2) \quad (6)$$

ω_n = *period of n-th neuron (*2π)* ϕ_n = *phase*

$$Y = a + \sum_{i=1}^{n} \sum_{j=1}^{n-1} b_{ij} \frac{\partial x_i}{\partial x_j} + \sum_{i=1}^{n} \sum_{j=1}^{n-1} \sum_{k=1}^{n-1} c_{ijk} \frac{\partial^2 x_i}{\partial x_j \partial x_k} + \ldots \quad (7)$$

a, B(b_{11}, b_{12},, ..., b_{nn-1}) ,... - parameters

Another example can solve 3 inputs, where 3rd input is sum of the first two inputs. Neural network will train to identify right result of sum (1+10=11) with output "1", wrong (7+2=5) with "0".

Neural network can teach itself to play chess. There are three chess pieces on the chessboard, two of which are black (rook and bishop) and one is white (knight). Input of the neural network is formed by **x** and **y**-positions of each piece on the chessboard (that means six inputs altogether). If any black piece is checking the white knight, neural network will indicate this (with output **"1"**). So there is only one output of the network enough (value **0** or **1**) for the state indication. There is an evident **dependence of input variables**, because if knight is checked by rook, so their either x or y-positions will equal. Likewise if knight is checked by bishop (chess pieces are on diagonal), there either sum or difference their x and y-positions will equal.

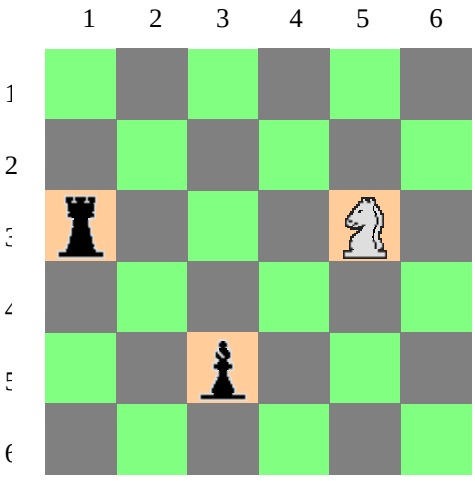


Fig. 5. Situation on chessboard.

So even if the neural network was trained only for identification of some inputs, it will be able to evaluate the other positions correctly, too. These **functional dependencies of inputs,** which form the output, will obtain a universal effect.

6 Conclusion

Identification dependencies of variables could be used to model any system with not proportionally changed related values of variables. Polynomial neural network creates a multi-parametric function, which output is based only on relations of inputs. Non linear periodic activation function transforms the dependent increased variables of input vector into proportionally changed related values. For example input pattern is moved (or changed) on matrix, where some points (coordinates) of pattern make inputs of the neural network. The points keep their dependence (although the pattern was changed), which is used to form the output of the network. This could be applied to weather prediction, where some points of pressure map are moving. There is not necessary to use all inputs, but some characteristic points of the map. The next state is forecasted in accordance with their dependence, which is learned by polynomial neural network. There will be not applied time-series prediction, which takes advantage of pattern identification.

References

1. Ivakhnenko, A.G.: Polynomial theory of complex systems, IEEE Transactions on systems, Vol. SMC-1, No.4, October 1971.
2. Vasechkina, E.F. & Yarin, V.D.: Evolving polynomial neural network by means of genetic algorithm, Complexity international, Vol. 09, July 2001.
3. Stanley J. Farlow: The G.M.D.H. Algorithm of Ivakhnenko, The american statistician, Vol. 35, No.4, November 1981.
4. Anastakis, L., Mort, N.: The development of self-organisation techniques in modeling : A Review of the GMDH. Research report num.813, October 2003 University of Sheffield.
5. Galkin, I.: Polynomial neural networks. Materials for Data mining course, University Mass Lowell.

Stimulácia mozgu pred učením generátorom vysokofrekvenčných impulzov

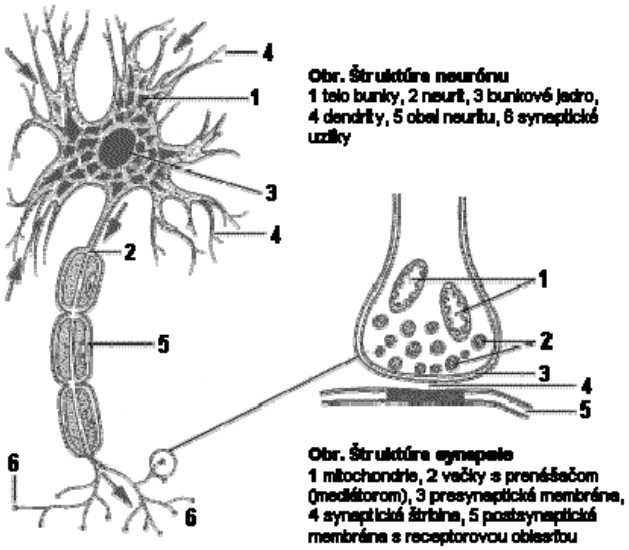
Ladislav Zjavka
Katedra informatiky, Fakulta riadenia a informatiky Žilinskej univerzity
Univerzitná 8215/1, 010 01 Žilina

Abstrakt. *Nervové bunky mozgu – neuróny vytvárajú navzájom prepojenia, ktoré umožňujú všetky poznávacie (kognitívne) procesy. Informácia skladajúca sa z rôznych pojmov nie je uložená na jednom mieste mozgu ale je distribuovaná v jeho viacerých oblastiach. Jej vyvolanie z pamäti pozostáva z integrácie týchto čiastočných záznamov do jednotnej predstavy. Účinnosť prenosu medzi nervovými bunkami sa mení pri učení (sústredenom prijímaní informácií) prostredníctvom synapsií, ktorými sú prepojené ich výbežky (axóny a dendrity). Opakovaná aktivácia jedného neurónu iným, prostredníctvom určitej synapsie, zvyšuje (alebo znižuje) silu jej prepojenia. Zmena účinnosti týchto prepojení v synapsiách sa odohráva zložitými molekulárnymi mechanizmami, pri ktorých dôležitú úlohu zohráva najmä vylučovanie vápnika. Synaptická účinnosť je mierou príspevku danej synapsie k sumárnemu postsynaptickému potenciálu, ktorý určuje čas a frekvenciu impulzov generovaných po dosiahnutí prahu excitácie neurónu. V mnohých oblastiach mozgu je možné prostredníctvom vysokofrekvenčnej stimulácie prívodných axónov nervových buniek vyvolať dlhodobé zvýšenie synaptickej účinnosti (vo forme zväčšenej amplitúdy excitačného postsynaptického potenciálu) – tzv. dlhodobú synaptickú potenciáciu v cieľových neurónoch (LTP). Týmto sa zvyšuje učinok vzájomného pôsobenia neurónov a tiež aj veľkosť zmeny synaptických prepojení, čo je možné využiť pri učení. Nízkofrekvenčnou stimuláciou je možné vyvolať dlhodobé zníženie sily synaptického prenosu – tzv. dlhodobú synaptickú depresiu (LTD). Tento spôsob sa využíva v uspávacích zariadeniach (impulzové generátory) používaných v nemocniciach v Rusku a Japonsku.*

1 Nervové bunky mozgu

Nervové bunky vysielajú a prijímajú (od iných buniek) impulzy, ktoré sú elektrickej povahy. V mozgu neexistuje žiadna centrálna riadiaca jednotka (ako v počítači), vyhodnotenie údajov a vydávanie riadiacich povelov sa uskutočňuje veľkým množstvom paralelne spracovávaných signálov na jednotlivých neurónoch. **Nervová bunka** (neurón) sa skladá z tela a tisícov výbežkov. Tieto môžu byť vstupné (dendrity) - vetvy nachádzajúce sa na tele bunky, ktoré privádzajú **signály** do neurónu od iných buniek. Jeden výbežok je výstupný (neurit, axón) - rozvetvuje sa do početných vetiev, ktorými sa šíri vzruch z neurónu k iným bunkám. Vetvy na konci axónu sú cez **synapsy** (môže ich byť niekoľko tisíc) spojené s dendritmi iných buniek{ XE "synapsa" }. V synapse sa môže signál prichádzajúci od inej bunky buď **zosilniť alebo zoslabiť**, silu jeho pôsobenia určuje premenlivá **váha** synapsy [3].

Keď suma kladných a záporných signálov od ostatných neurónov ovplyvnená váhami príslušných synáps prekročí istú hodnotu, nazývanú *prah excitácie*, neurón vygeneruje **výstupný impulz**. Ten sa šíri axónom do jeho vetví, odkiaľ cez synapsie a dendrity ovplyvňuje ďalšie neuróny. Zvyčajne neurón ako odpoveď na svoju stimuláciu vygeneruje celú sériu impulzov, ktoré majú nejakú priemernú frekvenciu ($10 – 10^2$ Hz). Frekvencia je úmerná celkovej stimulácii neurónu [2].



Obr. 1. Nervová bunka – neurón..

2 Učenie - vytváranie väzieb neurónov

Nervová sústava je pri vzniku organizmu čiastočne chaoticky usporiadaná, avšak základná štruktúra synaptických prepojení je určená **geneticky**. V priebehu života človeka sa vplyvom učenia a adaptácie na prejavy prostredia menia najmä váhy jednotlivých synaptických väzieb. *Vplyvom opakovaného zvýšeného (budiaceho) alebo zmenšeného (tlmiaceho) pôsobenia axónu jednej nervovej bunky na synapsiu inej bunky dochádza k trvalej alebo dočasnej biochemickej zmene jeho synaptickej účinnosti* (zosilneniu alebo zoslabeniu vstupného signálu). Pravidlo synaptických zmien pri učení navrhol D. Hebb [3].

Neuróny vytvárajú navzájom prepojenia, ktoré umožňujú všetky poznávacie (kognitívne) procesy. Zmenou účinnosti týchto synaptických prenosov, ich kombinácií a frekvencie šírenia sa v mozgu uchovávajú informácie a súvislosti medzi nimi. Informácia skladajúca sa z rôznych pojmov nie je uložená na jednom mieste mozgu ale je kódovaná celou oblasťou (sieťou) neurónov. Jej vyvolanie z pamäti pozostáva z integrácie týchto čiastočných záznamov do jednotnej predstavy. Tvorba pamäťovej stopy, ako vzorca distribuovanej aktivity neurónov, je z hľadiska rýchlosti závislá na intenzite procesu učenia. Pamäťová stopa sa tvorí veľmi rýchlo a môže sa vytvoriť po jednom spojení najmä ak ide o biologický alebo emociálne významný podnet. Učenie podľa Hebbovho pravidla vedie k zosilňovaniu vzájomných väzieb medzi synchrónne aktívnymi neurónmi. Tak sa v sieti vytvára vzor – silno pospájané zoskupenie neurónov. Pri vyvolávaní informácie z pamäti sa vplyvom vonkajšieho stimulu aktivuje časť zoskupenia neurónov, ktoré sú navzájom pospájané väzbami a kódujú danú informáciu [1].

3 Synaptická účinnosť prenosu

Účinnosť prenosu medzi nervovými bunkami sa mení pri učení (sústredenom prijímaní informácií) prostredníctvom synapsií, ktorými sú prepojené ich výbežky (axóny a dendrity). Opakovaná aktivácia jedného neurónu iným, prostredníctvom určitej synapsie, zvyšuje (alebo znižuje) silu jej prepojenia. Zmena účinnosti týchto prepojení v synapsiách sa odohráva zložitými molekulárnymi mechanizmami, pri ktorých dôležitú úlohu zohráva najmä vylučovanie vápnika. Synaptická plasticita je proces, v ktorom synaptické spojenia menia svoju účinnosť v dôsledku predchádzajúcej aktivity. **Synaptická účinnosť** (váha) môže byť definovaná ako veľkosť transmembránového napätia na membráne somy (tela bunky) postsynaptického neuróna (alebo na postsynaptickej membráne v samotnej synapse) ako dôsledok jednotkovej stimulácie presynaptického terminálu synapsy. Synaptická účinnosť je mierou príspevku danej synapsie k sumárnemu postsynaptickému potenciálu, ktorý určuje čas a frekvenciu impulzov generovaných po dosiahnutí **prahu excitácie neurónu.** Je priamo úmerná amplitúde a trvaniu postsynaptického potenciálu (PSP) na danej synapse. Synaptická váha (PSP po jednotkovej stimulácii synapsy) závisí od dvoch skupín faktorov :

A. Presynaptické factory : uvoľnené množstvo prenášača

B. Postsynaptické factory :

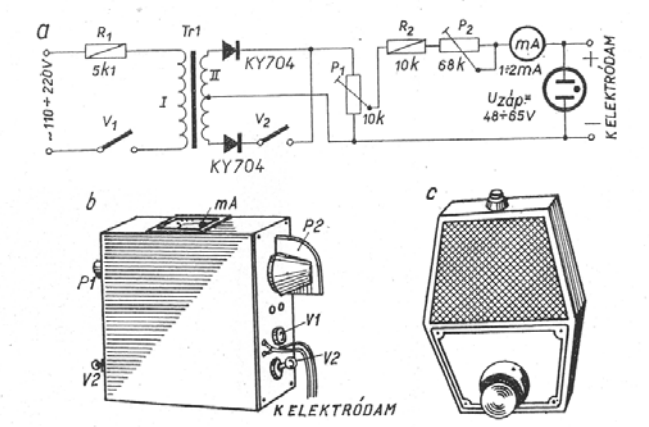
- počet receptorov
- typ a vlastnosti receptorov
- vstupná elektrická impendancia (odpor – závisí od morfológie dendritického tŕnika a jeho elektrických vlastností).

Zmena týchto synaptických vlastností vedie k zmene synaptickej váhy. Táto zmena môže byť krátko- alebo dlhotrvajúca, pozitívna alebo negatívna [1].

Pri popise mechanizmov synaptickej plasticity je užitočné rozdeliť si tento proces na dve fázy: **indukciu** (vyvolanie – odštartovanie procesov vedúcichk LTP/LTD) a **expresiu** (procesy zabezpečujúce samotnú zmenu váhy vo forme LTP/LTD a jej udržanie).

4 Vysokofrekvenčná stimulácia neurónov

V mnohých oblastiach mozgu je možné prostredníctvom vysokofrekvenčnej stimulácie prívodných axónov nervových buniek vyvolať dlhodobé zvýšenie synaptickej účinnosti (vo forme zväčšenej amplitúdy excitačného postsynaptického potenciálu) – tzv. **dlhodobú synaptickú potenciáciu** (long-term potentiation – LTP) v cieľových neurónoch. Týmto sa zvyšuje účinok vzájomného pôsobenia neurónov a tiež aj veľkosť zmeny synaptických prepojení, čo je možné využiť pre zvýšenie efektívnosti učenia. Nízkofrekvenčnou stimuláciou je možné vyvolať dlhodobé zníženie sily synaptického prenosu – tzv. **dlhodobú synaptickú depresiu** (long-term depression – LTD), teda opačný process [1]. Tento spôsob sa využíva v uspávacích zariadeniach (impulzové generátory) používaných v nemocniciach v Rusku a Japonsku na uspávanie pacientov po operáciách. Do prvej skupiny patria prístroje, ktoré obsahujú zvukový generátor šumov (ich frekvenciu je možné meniť) a imitujú šum dažďa alebo horskej bystriny. Do druhej skupiny patrí generátor elektrických impulzov. Jeho elektródy sa priložia na citlivé nervové zakončenia hlavy alebo krku a nastaví sa veľkosť impulzov (podľa subjektívnych pocitov pacienta). Prístroj sa nechá pôsobiť asi 10 min. Frekvencia impulzov je nastaviteľná v pásme 1 – 100 Hz. Dĺžka impulzov je tiež regulovateľná od 0.3 do 2 m/s, tvar impulzov je zvonový. Elektródy majú plochu 2 cm^2, ovinú sa gázou namočenou do roztoku kuchynskej soli a pripevnia bandážou na hlavu. Jedna z elektród sa pripevní na čelo blízko koreňa nosa (oči) a druhá na podbradok (krk). Prístroj sa zapne a nastaví tak, aby bolo citiť na miestach pripojenia elektród jemné šteklenie. Dobré výsledky sa zvyčajne dosiahnu, keď sa pripojí mínus pól zdroja na elektródu upevnenú na čele a plus pól na elektródu umiestnenú na krku. Pri nesprávnom pripojení elektród môžu vzniknúť bolesti hlavy. Počas uspávania musí byť pacient v takej polohe, aby sa vylúčil dotyk s akýmkoľvek kovovým predmetom. Zistilo sa, že pocity pacientov počas elektrospánku i po ňom sú príjemné. Elektronické uspávacie prístroje sa používajú aj pri inej liečebnej terapii. Dochádza k uvoľneniu a regenerácii organizmu, čo má priaznivý vplyv na zdravotný stav pacienta. Pri prvých pokusoch pacienti spravidla nezaspia, spánok alebo útlm sa dostaví až neskôr. Klinické výskumy dokázali úplnú neškodnosť používania elektronických uspávacích prístrojov. Pokusy s nimi sa robili najmä v Rusku a Japonsku. Uspávací prístroj na obr.2. bol s úspechom vyskúšaný v klinickej praxi v Poľsku [4].

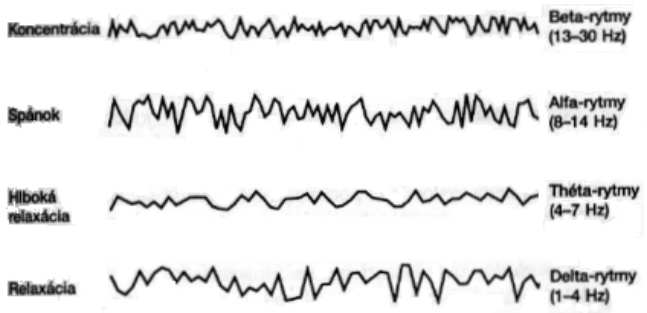


Obr. 2. a – jednoduchý generátor elektrických impulzov; b – konštrukcia; c – uspávací prístroj imitujúci zvuk dažďa.

Podobné zariadenie s vyšším frekvenčným rozsahom (500 Hz – 2 kHz) by sa mohlo využiť pri stimulácii mozgu pred učením na vyvolanie opačného účinku ako LTD, teda na zvýšenie synaptickej účinnosti prenosu (LTP) medzi nervovými bunkami. Pri tomto synaptickom prenose dochádza k metabolickým procesom, ktoré sú spúšťané zvýšenou koncentráciou vápnika postsynaptickej membrány, ktorý ovplyvňuje trvalú plasticitu synaptickej váhy. Nízkofrekvenčná aferentná aktivita spôsobuje nízku koncentráciu vápnika. Vnútrobunková koncentrácia kalcia je teda primárnym spúšťačom LTD alebo LTP [2].

Elektrická aktivita mozgu sa meria prístrojom EEG (elektroencefalograf)), podobne ako frekvencia srdca sa meria prístrojom EKG. Pôsobenie nízko- a vysoko-frekvenčných zvukových alebo elektrických impulzov na

na zvýšenie synaptickej účinnosti prenosov medzi nervovými bunkami v mozgu človeka by bolo možné overiť napr. pri zapamätaní nových slov cudzieho jazyka. Vhodné asi bude striedať pôsobenie vysoko- a nízko-frekvenčných signálov pri stimulácii (napr. 10 min.), teda vytvárať periódy zvýšenej synaptickej plasticity a depresie. Pomocou EEG by sa dal sledovať účinok stimulačných alebo útlmových impulzov na aktivitu mozgu, zistiť najvhodnejšiu frekvenciu, veľkosť impulzov a tiež miesta (citlivé nervové zakončenia) pre priloženie elektród.



Obr. 3. EEG základných stavov mozgu.

5 Časovo asymetrická synaptická plasticita

Z experimentov vyplýva, že dlhodobá modifikácia synaptickej účinnosti závisí od časového rozloženia presynaptických a postsynaptických akčných potenciálov (AP). Časové poradie aktivácie presynaptického vstupu a postsynaptického AP určuje, či vznikne LTP alebo LTD. Opakované spojenie presynaptickej aktivácie s tesne nasledujúcou postsynaptickou sériou AP-ov (spätne sa šíriaca depolarizácia AP) spôsobuje väčší postsynaptický vstup vápnika a LTP. Postsynaptický výboj spätne sa šíriacichA AP-ov, ktorý časovo predbehne presynaptickú aktivitu, naopak vedie k menšej koncentrácii vápnika a LTD. Z počítačových simulácií vyplýva, že táto časovo asymetrická hebbovská synaptická plasticita podporuje učenie a zapamätávanie si sekvencií signálov, pretože má tendenciu zosilňovať vzájomnú väzbu neurónov, ktoré vytvárajú kauzálne reťazce. Ak neurón A opakovane vysiela signál tesne pred aktivitou neurónu B, spojenie medzi nimi sa posilní, takže v budúcnosti bude s vyššou pravdepodobnosťou neurón B vysielať signál tesne po neuróne A. Pokusy so zvieratami s veľmi jednoduchým nervovým systémom (mäkkýš Aplysia Californica) a tiež u cicavcov dokázali správnosť uvedenej teórie (habituácia, senzitizácia – neasociačné učenie, podmieňovanie – asociačné učenie) [1].

Literatúra

1. Jedlička, P., Beňušková, Ľ., Mačáková, J., Ostatníková, D.: Molekulové mechanizmy učenia a pamäti. Kapitola v knihe - Hulín, I.: Patofyziológia. Slovak Academic Press, Bratislava 2002, str. 1183-1199.
2. Beňušková, Ľubica : Neurón a mozog. Kognitívne vedy, Kalligram Bratislava 2002.
3. Kotek, Z., Chalupa, V., Brúha, I., Jelínek, J.: Adaptivní a učící se systémy, SNTL 1980.
4. Wojciechowski, J. : Amatérske elektronické modely. SNTL Bratislava 1988.
5. Buráková, B.: Zrýchlené učenie v hladine alfa. Zborník konferencie „Quo vadis zdravotníci“ Prešov 22.-23.apríla 2005