Transformace znalostí mezi modely pro zachycování znalostí v XML

Marek Růžička[1]

[1] Evropské centrum pro medicínskou informatiku, statistiku a epidemiologii – Kardio a Katedra informačního a znalostního inženýrství, VŠE Praha
ruza.m@volny.cz

Abstrakt. Tento článek navrhuje systém pravidel XKBT (*XML Knowledge Block Transformation*) určený pro transformace znalostí mezi modely pro zachycování znalostí. XKBT napomáhá identifikovat vztahy mezi třídami znalostí různých modelů a zaznamenává je ve formě pravidel na jejichž základě se následně transformují instance těchto tříd ze zdrojového do cílového modelu. Pro zápis pravidel i znalostních modelů se používá jazyk XML. Modelů pro zachycování znalostí v XML v současnosti stále přibývá a díky tomu začíná být úloha transformace znalostní báze obsažené v takovém modelu aktuální. Kompletní sada pravidel pro konkrétní model zásadním způsobem usnadňuje transformační proces. Výsledkem tohoto procesu je nová znalostní báze zapsaná v odlišném formátu, která je po obsahové stránce do značné míry shodná s bází původní. Pro různé formáty mohou existovat specifické nástroje (např. pro reprezentaci znalostí apod.) a díky transformacím lze přizpůsobit určitou znalostní bázi tak, abychom mohli využít podpory kteréhokoliv z těchto nástrojů.

1 Úvod

Problematika zpracování a reprezentace znalostí je rozvíjena již několik desetiletí a v současnosti si v ní našel své místo i stále populárnější jazyk XML (*eXtensible Markup Language* [2]). Kromě klasického značkování dokumentů a uchovávání rozsáhlých databází se jedná o další alternativní využití tohoto univerzálního nástroje. Obdobně jako jsou v souvislosti s databázemi založenými na XML vyvíjené adekvátní dotazovací jazyky, tak v oblasti znalostního inženýrství vznikají nástroje pro různé úlohy zpracování znalostí. Především se jedná o samotné zachycování znalostí. Například v oblasti medicíny pro tyto účely existuje již celá řada modelů. Některé z nich byly od počátku koncipovány na bázi XML (GEM [8]) a většina ostatních k XML postupně přechází (GLIF [5], Asgaard [7]). Další využití nachází XML jako výměnný formát mezi znalostními systémy. Obzvlášť oblíbený je při přenosu znalostí mezi ontologiemi (XOL [4], DAML-OIL).

V tomto článku se XML staví do role prostředku pro transformaci znalostí. Snažili jsme se vytvořit systém obecných transformačních pravidel (XKBT), který by usnadnil přechod od jednoho znalostního modelu ke druhému. Pro většinu znalostních modelů je typické že definují třídy, které v nich lze zachycovat. Základním

smyslem XKBT je “namapovat“ mezi sebou odpovídající třídy dvou modelů a explicitně vyjádřit transformační pravidla "zařazení" instancí zdrojového modelu do modelu cílového. Pochopitelně se v rámci transformace mohou některé zdrojové instance slučovat či naopak rozdělovat. Mapování tříd sebou přináší problém *kompatibility* modelů. Každý model účelově zachycuje určité typy znalostí, které jsou dány smyslem celého modelu resp. podobou úloh, které chceme nad daným modelem řešit. Z toho vyplývá, že pokud jsou dva modely určeny k řešení zcela odlišných úloh, tak transformace znalostní báze mezi nimi nemá velký smysl.

Jedna z hlavních výhod transformací je skutečnost, že při tvorbě nové znalostní báze nemusíme začínat "na prázdném papíře", ale pouze upravujeme osvědčené jádro znalostí do nového formátu a přibalujeme k němu dodatečné znalosti.

Z pohledu XML se jedná o transformaci mezi dvěma strukturami elementů definovaných v různých DTD. Stejné problematice se věnuje i známý jazyk XSLT [3] z dílen W3C, ale přesto existuje několik závažných důvodů proč XKBT vyvíjet. Hlavní výhoda XKBT spočívá tom, že se jedná o *řízenou* transformaci. Pravidla nabízejí v některých situacích pro konkrétní instanci různá řešení a osoba řídící transformaci vybírá optimální volbu. Pokud například zdrojový model obsahuje podstatné části ve volném textu a naopak cílový model je postaven na detailní sémantické struktuře, tak je mnohdy potřeba vstupní text interpretovat a podle jeho smyslu vybrat příslušné transformační pravidlo. XSLT s tímto přístupem nepočítá, je pouze jednorázově aplikováno na vstupní dokument bez možnosti průběžného usměrňování průběhu transformace. XKBT se výhradně věnuje zpracování znalostí (znalostních bloků, tříd) a výčet smysluplných operací aplikovatelných na takovéto znalosti je velmi omezený. Proto se zdá být výhodnější vytvořit srozumitelný předpis pro každou z těchto operací, než je někdy velmi neobratně popisovat v XSLT. U XKBT se předpokládá, že ho budou využívat i lidé s povrchní či minimální znalostí XML technologií, a tak musí být kladen důraz na jejich snadnou interpretaci a pochopení. Na druhou stranu by bylo škoda nevyužít potenciálu XSLT, a proto je možné v rámci XKBT tyto pravidla integrovat.

2 Transformace znalostních bloků

2.1 Transformace mezi znalostními modely

Základní situací pro XKBT představuje problém transformace znalostních bloků (instancí tříd zvoleného vstupního modelu) mezi vstupním a výstupním formátem. V případě, že jsou si oba formáty velmi blízké tzn. typy základních tříd si odpovídají, postačuje pro každou třídu vytvořit jedno univerzální transformační pravidlo. Tato představa je ale v praxi velmi nereálná a skutečné mapování znalostí bývá mnohem komplikovanější. Jedno z možných řešení, jak toto mapování realizovat, je rozdělit transformační pravidlo na větší počet elementárních pravidel. Elementární pravidla se pak aplikují sekvenčně. Formálně je tato situace znázorněna rovnicí (1), kde *KB*

znamená znalostní blok (*knowledge block*) a funkce *t* (*transformace*) odpovídají transformačním pravidlům XKBT.

$$KB_{final} = t_n(...t_2(t_1(KB_{source}))) \quad (1)$$

Variabilita mapování je dosažena tím, že pro každou transformaci *t* je možné předdefinovat více aplikovatelných pravidel. Sekvence pravidel pak závisí na volbě konkrétních elementárních pravidel pro každé *t* podle charakteru vstupního znalostního bloku. Situace popsaná rovnicí (1) ovšem stále předpokládá vztah 1:1 mezi třídami obou formátů. Ve skutečnosti XKBT podporuje složitější model, ve kterém každá transformace *t* může mít buď větší počet vstupních nebo výstupních znalostních bloků (nikoli oboje najednou). Tímto způsobem může dojít k již zmíněnému slučování či jinému vzájemné ovlivňování se mezi instancemi.

Tento postup transformace přináší řadu výhod. Je-li dodržen předpoklad, že pravidla XKBT jsou snadno interpretovatelná, tak pro osobu obeznámenou se vstupním formátem je velmi snadné zpracovat libovolné instance a získat jejich odpovídající přepis ve výstupním formátu. Podotkneme-li, že za tímto účelem je vyvíjena softwarová podpora Stepper[1], která umožňuje aplikovat sekvenci XKBT pravidel s tím, že zároveň zcela zakrývá jejich zápis v XML, tak se transformace mění v rutinní záležitost. Uvedená softwarová podpora pochopitelně nabízí i editor XKBT včetně slovní interpretace konkrétního pravidla.

2.2 Formalizace textu

Druhou variantou použití XKBT je zpracování volného textu, kdy nevycházíme z existujícího modelu znalostí, nýbrž tyto znalosti v textu hledáme a určitým způsobem se je snažíme formálně popsat. V širším kontextu se jedná o převod textového dokumentu[2] do jeho znalostní reprezentace. Tento proces se od předešlého liší především tím, že je nejprve nutné ve zdrojovém dokumentu vybrat všechny pasáže textu, které se zdají být potenciálními nositeli znalostní a označit je jako primární znalostní bloky. S těmito bloky se pak pracuje jako se vstupními třídami a zbytek transformačního procesu pak probíhá obdobně jako v předešlém případě.

Vstupní dokument musí být ve formátu XHTML [1], jenž je v mnoha ohledech identický s jazykem HTML, ale zároveň vyhovuje i požadavkům na dokumenty formátu XML. Do XHTML je například možné z XML odkazovat, a to je jeden z velmi důležitých aspektů XKBT. Testování formalizace textu tímto přístupem bylo provedeno na lékařských doporučených postupech[3] a jeho výsledkem bylo potvrzení, že lze pomocí XKBT dospět z textového dokumentu až do formální reprezentace [9]. Původní myšlenka zpracování dokumentu počítala mimo jiné s tím, že proces formali-

[1] euromise.vse.cz/stepper/

[2] Samozřejmě není možné daný postup aplikovat na libovolný dokument, ale pouze jen na dokumenty s určitými vlastnostmi např. odborný charakter, znalostní bohatost, tématická ucelenost a další.

[3] Lékařské doporučené postupy jsou zvláštní formou medicínských dokumentů popisujících určitou chorobu či syndrom z různých úhlů (diagnóza, postup léčby, užívání léků apod.)

zace bude rozdělen do více separátních kroků, aby byla zachována maximální přehlednost [6]. Jinými slovy během prvního kroku se vyznačí všechny primární bloky a v následujících krocích se na každý blok použije právě jedno transformační pravidlo. Tímto způsobem získáme po dokončení každého kroku novou verzi původního dokumentu v určitém stádiu transformace resp. formalizace. Pokud je výběr jednotlivých pravidel XKBT pečlivě zaznamenán, tak je možné celý transformační proces přehledným způsobem zpětně zobrazit.

3 Specifikace XKBT transformačních pravidel

XKBT definuje čtyři základní typy pravidel - *vztah 1:1*, *dekompozice*, *agregace* a *situaci mrtvého konce*. Základem každého pravidla je identifikace zdrojové a cílové sestavy znalostních bloků resp.elementů XML, jenž je reprezentují. Záměrně zde hovoříme o sestavě bloků a nikoli o sekvenci, protože v XKBT lze vstupní sestavu popsat volnějším způsobem než jen pouhým výčtem elementů (výskyt některých elementů může být nepovinný, nemusí být dodrženo přesné pořadí elementů apod.).

Pokud se vstupní sestava neshoduje se sestavou, kterou chceme transformovat, tak musíme zvolit jiné, vhodnější pravidlo. Samozřejmě může nastat situace, kdy nevyhovuje žádné pravidlo. To lze řešit několika způsoby. Buď je zvolená sestava nemapovatelná směrem k výstupnímu formátu a je potřeba hledat jiné řešení (např. některé znalostní bloky "opustit" a dále je již netransformovat - případ *pravidla mrtvého konce*), nebo je nedostatečná soustava pravidel, a tak by měla být náležitě doplněna. Další možností je přidání externích znalostí ke vstupní sestavě, kdy obohacením základních znalostí ze vstupního modelu získáme plnohodnotný znalostní blok ve vyšším stádiu transformace. Z důvodu zachování lepší čitelnosti je povoleno zadat sestavu znalostních bloků pouze na jedné straně pravidla (vstupní/výstupní). Na druhé pak musí být pouze jeden samostatný znalostní blok.

Pravidlo tedy obsahuje jeden element pro vstupní sestavu (*source* pro samostatný blok nebo *compound-source* pro kombinovanou sestavu) a jeden pro výstupní sestavu (obdobně *dest* a *compoud-dest*) Na následujícím příkladě je znázorněno agregační pravidlo, kde vstupní sestavu tvoří sekvence až pěti elementů *findings* a jeden element *conclusion* a výsledkem agregace je nový element *diagnosis*.

```
<aggregation name="example - aggregation">
  <compound-source type="sequence">
    <compound-source type="iteration" maxOccurs="5">
      <source element="findings" />
    </compound-source>
    <source element="conclusion" />
  </compound-source>
  <dest element="diagnosis" />
</aggregation>
```

Takto jednoduché pojetí pravidel by zdaleka nestačilo postihnout variabilitu reálných situací. Je velmi nepravděpodobné, že pro typově stejnou vstupní sestavu zna-

lostních bloků chceme použít vždy stejné pravidlo. Pokud máme pro tutéž sestavu více pravidel, potřebovali bychom jemnější způsob pro určení, kdy které z nich vybrat. To zajišťují *podmínky* zkoumající vstupní sestavu po stránce její strukturu elementů a hodnot atributů. Druhý příklad demonstruje základní typy podmínek testující hodnoty atributů a existenci atributů.

```
<one-to-one name="example – conditions>
  <source element="diagnosis" conditions="all">
    <cond attribute="treatable_disease" value="yes" />
    <cond attribute="contraindications" exist="no" />
  </source>
  <dest element="begin_treatment" />
</one-to-one >
```

Další typy podmínek se věnují struktuře elementů tzn. jaké má určitý element potomky, předky a následníky. Díky podmínkám dochází k automatické selekci pravidel aplikovatelných na zvolenou vstupní sestavu. Pokud je sada pravidel dobře navržena, tak se díky podmínkám vždy vybere úzký okruh vhodných pravidel, čímž je usnadněna závěrečná volba. Zároveň je eliminován případ, kdy by na vstupní sestavu bylo použito logicky nesprávné pravidlo.

Doposud se předpokládalo, že po aplikaci konkrétního transformačního pravidla vzniknou ve výstupní sadě odpovídající nástupci, kteří jsou tvořeni prázdnými elementy. To je značně nepraktické, protože bychom potřebovali, aby se při transformaci automaticky vyplnily některé atributy případně i substruktura cílových elementů. XKBT se samozřejmě věnuje i této problematice. Poslední příklad letmo nastiňuje způsob přenášení hodnot atributů při transformaci. Atribut *allAttrCopy* s hodnotou *yes* říká, že všechny atributy vstupního elementu *diagnosis* se zkopírují do výstupního bloku. První element *attrException* navíc kopíruje hodnotu atributu *name* do atributu *ID* a druhý vylučuje z okruhu kopírovaných atributů atribut *ref*.

```
<source element="diagnosis">
  <exception allAttrCopy="yes" >
    <attrException attribute="name" copyTo="ID" />
    <attrException attribute="ref" copy="no" />
  </exception>
</source>
```

Obdobně snadným způsobem se pracuje i s vnitřní substrukturou vstupních elementů. Detailní popis syntaxe pravidel, podmínek a způsobu kopírování se nachází přímo v souboru DTD[4] definujícím XKBT.

4 Závěr

XKBT nabízí snadný způsob zpracování znalostí a jejich transformací mezi znalostními modely. K jeho možnému využití přispívá fakt, že se v současnosti dynamicky

[4] euromise.vse.cz/stepper/xkbt/xkbt.dtd

rozšiřuje nabídky znalostních modelů založených na XML. Zásadně pak XKBT ovlivní vyvíjená softwarová podpora, která jednak odboural potřebu podrobného seznámení s technickými detaily XKBT i samotného XML, a dále zastane automaticky podstatnou část aplikace pravidel.

Budoucí vývoj XKBT bude zaměřen zejména na hledání vhodných znalostních modelů a přípravě odpovídajících sad transformačních pravidel, které budou zahrnuty v softwarové podpoře jako standardní knihovny.

Článek vznikl s podporou projektu MŠMT ČR LN00B107.

Reference

1. Altheim M., McCarron S.:XHTML 1.1 – Module-based XHTML – W3C Working Draft. W3C 2000, http://www.w3.org/TR/xhtml11
2. Bray T., Paoli J., Sperberg-McQueen C.M., Maler E..: Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 6 October 2000. http://www.w3.org/TR/REC-xml
3. Clark, J.: XSL Transformations (XSLT) Version 1.0. W3C 1999., http://www.w3.org/TR/xslt
4. Karp PD, Chaudhri VK, Thomere J.: XOL: An XML-Based Ontology Exchange Language, 1999., http://www.ai.sri.com/pkarp/xol/
5. Peleg M., Boxwala AA, Ogunyemi O, Zeng Q, Tu S, Lacson R, Bernstam E, Ash N, Mork P, Ohno-Machado L, Shortliffe EH, Greenes Ra.: GLIF3: The evolution of a guideline representation format. Proc AMIA Symposium, 2000; 645-649
6. Růžička M, Svátek V, Kroupa T.: Víceúrovňová formalizace obsahu textových dokumentů, Znalosti 2001;114-123
7. Shahar, Y.; Miksch, S.; Johnson, P.: The Asgaard Project: A Task-Specific Framework for the Application and Critiquing of Time-Oriented Clinical Guidelines, Artificial Intelligence inMedicine, 14, pp. 29-51, 1998.
8. Shiffman RN, Karras BT, Agrawal A, Chen R, Marenco L, Nath S.: GEM: A Proposal for a More Comprehensive Guideline Document Model Using XML. Journal of the American Medical Informatics Association 2000 Sep-Oct;7(5):488-98
9. Svátek V. & Růžička M., Step–By–Step Mark–Up of Medical Guideline Documents. In: (Surjan G. et al., eds.) Health Data in the Information Society. Proceedings of MIE2002, Budapest 2002. IOS Press, 591-595.

Abstract. This paper lays out the design of XKBT (*XML Knowledge Block Transformation*), a rule markup language for knowledge transformations. XKBT was developed to identify relations between classes from different knowledge models and to describe them as specific type of rules. These rules are later used for transformation of instances stored in one knowledge model to another. This approach enables fast transformation of the whole knowledge base to any suitable model. Main advantage of XKBT is possibility of sharing the same knowledge base in several formats. This includes also sharing of all additional tools dedicated to only one format.