Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Prúdové modely transformácií štrukturovaných dát a ich rozšírenia

Autoreferát dizertačnej práce

Pavel Labath

---

na získanie vedecko-akademickej hodnosti philosophiae doctor
v odbore doktorandského štúdia: 9.2.1 Informatika

Bratislava, 2014

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia na Katedre informatiky Fakulty matematiky, fyziky a informatiky Univerzity Komenského v Bratislave.

Predkladateľ: Mgr. Pavel Labath
Katedra informatiky
Fakulta matematiky, fyziky a informatiky
Univerzita Komenského
Mlynská Dolina
842 48 Bratislava

Školiteľ: Prof. RNDr. Branislav Rovan, PhD.
Katedra informatiky FMFI UK
Bratislava

Oponenti: .....

.....

.....

Obhajoba dizertačnej práce sa koná dňa ..... o ..... na ..... pred komisiou pre obhajobu dizertačnej práce v odbore doktoranského štúdia

9.2.1 Informatika

vymenovanou predsedom odborovej komisie dňa .....

Predseda odborovej komisie:
Prof. RNDr. Branislav Rovan, PhD.
Fakulta matematiky, fyziky a informatiky
Univerzity Komenského
Mlynská Dolina
842 48 Bratislava

Obsah

1 Úvod

V práci skúmame možnosti prúdového spracovania štrukturovaných dokumentov. Špeciálne sa zameriavame na XML a JSON, dve najpoužívanejšie technológie na prácu so štruktúrovanými dátami na Internete.

XML je meta-jazyk definovaný W3 konzorciom s cieľom špecifikovať štrukturované dokumenty. Pôvodná W3C špecifikácia bola zverejnená v roku 1998. Odvtedy sa XML stal populárnym formátom na výmenu dát medzi aplikáciami. Navyše, XML je úžitočný nástroj v každej doméne, v ktorej je potrebné vytvárať štandardy pre formát dokumentov. Populárnym formátom na popis štrukturovaných dát je aj JSON, ktorý vďaka jednoduchosti zápisu začína nahrádzať XML.

Jednou z najdôležitejších operácií na štrukturovaných dátach sú ich transformácie. Je častou situáciou, že si dve aplikácie potrebujú vymieňať dáta, ale každá používa inú vnútornú štruktúru na vlastné spracovanie. Pri prenose z jednej aplikácii do druhej, dáta musia byt transformované. Klasický procesor transformácií štrukturovaných dát (napríklad procesory XML transformačných jazykov XSLT a XQUERY) najprv načíta celý vstupný dokument do pamäte a potom transformuje dáta na základe špecifikácii transformácie. Takémuto spôsobu spracovania dokumentov hovoríme *stromový*. Stromový spôsob spracovania je prirodzený, keďže umožňuje ľahké vyjadrenie a vykonanie zložitých transformácií. Postupom času sa štrukturované dokumenty (najprv XML) stávali populárnejšími, čím vznikla potreba spracovávať dáta, ktoré boli väčšie ako operačná pamäť bežných počítačov. Navyše sa štrukturované dáta začali používať aj v tzv. *real-*

time aplikáciách, kde bolo potrebné poznať časť výstupu ešte predtým, ako bol načítaný celý vstup transformácie.

Alternatívou k stromovému spracovaniu je prúdové spracovanie. Tu je vstupný dokument čítaný sekvenčne, výstupný dokument je taktiež generovaný sekvenčne a tieto dva procesy prebiehajú súbežne. Keďže len časť vstupného dokumentu je prístupná v každom okamihu, pri preklade musia byť použité pokročilé techniky na spracovanie transformácií. Napriek viac ako desaťročiu výskumu XML, problém prúdového spracovania štrukturovaných dát zostáva otvoreným. Prúdovému spracovaniu dotazov v jazyku XPATH bolo venované veľké množstvo prác [6, 37, 38, 36, 69, 67, 68, 17, 41, 66, 20]. V poslednej dobe sa začali objavovať prúdove procesory jazykov XPATH a XQUERY [71, 48, 15, 32, 7, 42, 21]. Avšak, trieda transformácií spracovateľných v prúdovom režime zostáva veľmi obmedzená. Toto je sčasti dôsledkom prirodzených obmedzení prúdového spracovania, kde nie je napríklad možné vymeniť poradie dvoch veľkých podstromov bez uloženia jedného z nich do pamäte. Takýto intuitívny pohľad súhlasí s výsledkami v [24], kde skúmali triedu transformácií, ktoré sa dajú vykonať bez buffrovania. Na tento výskum nadväzuje [25], kde autori uvažujú aj schémy vstupných dokumentov.

Cieľe práce

Cieľom práce bolo preskúmať triedy prúdovo spracovateľných transformácií štrukturovaných dát, nájsť ich charakterizácie a algoritmy na ich spracovanie. Dalej, sme analyzovali rozšírenia klasického modelu prúdových transformácií a hľadali nové modely spracovania, ktoré zachovávajú výhody prúdového spracovania a zachytávajú širšie triedy transformácií.

2 Základné pojmy a označenia

Jednoduchý všeobecný XML prekladač (SGXT— *Simple General* XML *Transducer*) bol definovaný v [25] za účelom skúmania transformácií spracovateľných v prúdovom režime.

Definícia 1. *Les nad abecedou* Σ *indexovaný množinou* I *označujeme* $\mathcal{H}_\Sigma(I)$ *a rekurzívne definujeme nasledovne:*

- *každý symbol z* I *patrí do* $\mathcal{H}_\Sigma(I)$*,*
- *pre každé* $n \geq 0$ *a pre stromy* $t_1, \ldots, t_n \in \mathcal{H}_\Sigma(I)$*, les* $(t_1, \ldots, t_n)$ *patrí do* $\mathcal{H}_\Sigma(I)$*,*

- *pre každé* $\sigma \in \Sigma$ *a les* $h \in \mathcal{H}_\Sigma(I)$*, strom* $\sigma(h)$ *patrí do* $\mathcal{H}_\Sigma(I)$*.*

Definícia 2. *Nech* $\mathcal{P}$ *je trieda* XPath *výrazov. Jednoduchým všeobecným* Xml *prekladačom nad* $\mathcal{P}$ *(*Sgxt$_\mathcal{P}$*) je pätica* $T = (Q, \Sigma, \Delta, q_0, R)$*, kde*

- Q *je konečná množina stavov,*
- Σ *a* Δ *sú (konečné) vstupné a výstupné abecedy,*
- $q_0 \in Q$ *je začiatočný stav,*
- R *je konečná množina pravidiel taká, že pre každé* $q \in Q$ *a* $\sigma \in \Sigma$ *existuje práve jedno pravidlo tvaru*

$$(q, \sigma) \to \xi$$

v R*, kde* $\xi \in \mathcal{H}_\Sigma(I)$*.*

Spôsob, akým Sgxt naviguje vstupným stromom je určený parametrom $\mathcal{P}$. V [25] sa používa trieda *top-*$\mathcal{P}(\Sigma)$, definovaná nasledovne:

Definícia 3. *Trieda jednoduchých vertikálnych výrazov top-*$\mathcal{P}(\Sigma)$ *je trieda výrazov definovaná gramatikou*

$$E ::= A_1 :: \sigma_1 / \cdots / A_l :: \sigma_l$$
$$A ::= \mathit{child} \mid \mathit{descendant} \ ,$$

kde $\sigma_1, \ldots, \sigma_l \in \Sigma$*.*

Aby transformácia definovaná pomocou Sgxt T mohla byť spracovaná prúdovo pomocou algoritmu v [24], T musí spĺňať určité vlastnosti. Tieto vlastnosti sú definované pomocou nasledovných dvoch funkcií:

- hodnotou funkcie *eval-doc*(p, d, α) je zoznam vrcholov dokumentu d označených výrazom XPath p začínajúcim z vrchola α.
- hodnotou funkcie *eval-expseq-doc*(ξ, d, α) je zreťazenie zoznamov, ktoré su hodnotame funkcií *eval-doc*(p_i, d, α), kde ξ je pravá strana Sgxt pravidla a $p_1, \ldots, p_l$ sú výrazy vyskytujúce sa v ξ (v tomto poradí).

Sgxt T zachováva poradie (je *order-preserving*) na množine dokumentov $\mathcal{D}$, ak sú vrcholy v návratovej hodnote funkcie *eval-expseq-doc*(ξ, d, α) usporiadané pre každý dokument $d \in \mathcal{D}$, zakaždým keď je pravidlo $(q, \sigma) \to \xi$ použité vo vrchole α dokumentu d. Sgxt T má disjunktné podstromy (je *branch-disjoint*), ak zoznam *eval-expseq-doc*(ξ, d, α) neobsahuje vrchol α a potomka α.

Ak uvažujeme $\mathcal{D}$ ako množinu všetkých dokumentov nad danou abecedou, tak trieda prúdovo spracovateľných SGXT bude veľmi malá. Preto v [25] autori uvažujú triedu dokumentov danú schémou a ukazujú algoritmus, ktorý rozhoduje či daný SGXT zachováva poradie a ma disjunktné podstromy na množine dokumentov danej vstupnou schémou. Autori uvažujú obmedzenú podtriedu schém, v ktorej nie je možné používať rekurzívne definície elementov schémy a tiež ani zjednotenie v definícii obsahu elementov.

3 Test prúdovosti pre väčší fragment

Prvým výsledkom práce je navrhnutie nového algoritmu na rozhodovanie prúdovosti pre daný SGXT a danú schému vstupných dokumentov.[1] V porovnaní s algoritmom v [25], náš algoritmus podporuje širšiu triedu SGXT a zároveň aj širšiu triedu schém. Zväčšenie triedy SGXT je realizované širšou množinou podporovaných XPATH výrazov, *fwd*-$\mathcal{P}(\Sigma)$.

Definícia 4. *Trieda jednoduchých dopredných* XPATH *výrazov fwd*-$\mathcal{P}(\Sigma)$ *nad abecedou* Σ *je trieda výrazov daná gramatikou:*

$$
\begin{aligned}
X &::= A::a_0/B::a_1/\cdots/B_k::a_k \\
A &::= \mathit{child} \mid \mathit{descendant} \\
B &::= A \mid \mathit{following\text{-}sibling}\texttt{[1]} \mid \mathit{following\text{-}sibling} \ ,
\end{aligned}
$$

kde $a_0, \ldots, a_k \in \Sigma \uplus \{*\}$.

Širšia trieda schém sa prejavuje odstranením obmedzenia na rekurzívne definície elementov schém a povoleniu zjednotenia v definiciach obsahu elementov. V porovnani s triedou *top*-$\mathcal{P}(\Sigma)$, trieda výrazov *fwd*-$\mathcal{P}(\Sigma)$ povoľuje dve nové osi a používanie znaku '$*$', označujúceho ľubovoľný nazov elementu. Ak použijeme nasledovné symboly na označenie rôznych parametrov vstupu

n	počet vrcholov v schéme
m	počet pravidiel v SGXT
l	maximálny počet XPATH výrazov v jednom SGXT pravidle
k	maximálna dĺžka XPATH výrazu
c	maximálna dĺžka popisu obsahu jedného vrchola v schéme

tak potom časová zložitosť nášho algoritu bude $O(n^2mck^2l^2)$. Tieto výsledky boli publikované v [53] a boli prezentované na workshope INQUEST na univerzite v Oxforde [52].

[1] Teda, algoritmus rozhoduje či SGXT zachováva poradie a má disjunktné podstromy.

		E ::=
premenné		x
znaky	|	$'c'$, for $c \in \Sigma$
rovnosť	|	$E_1 = E_2$
konštrukcia zoznamov	|	$E_1 \mid E_2$
rozklad zoznamov	|	**match** E { $x_1 \mid x_2 \rightarrow E_1$, **nil** $\rightarrow E_2$ }
podmienený výraz	|	**if** E **then** E_1 **else** E_2
konštrukcia n-tíc	|	$(E_1, \ldots, E_n)$, $n \geq 2$
rozklad n-tíc	|	**let** $(x_1, \ldots, x_n) = E_1$ **in** E_2
definovanie funkcií	|	**fun** $x{:}T_1 \rightarrow T_2$ { E }
volanie funkcií	|	$E_1(E_2)$
ošetrovanie výnimiek	|	**try** E_1 **catch**(x) E_2
vyvolanie výnimiek	|	**raise**$_T(E)$

Obr. 1: Syntax jazyka X-Fun

4 Programovací jazyk na spracovanie stromov

Ďalším prínosom práce je návrh nového funkcionálneho programovacieho jazyka X-Fun. X-Fun bol navrhnutý ako prostriedok na efektívny popis stromových transformácií a zláštny dôraz bol kladený na umožnenie efektívnej prúdovej implementácie programov v tomto jazyku. Jazyk X-Fun je prvým jazykom, ktorý má presne definovanú formálnu sémantiku, podporuje dotazy na vstupných stromoch pomocou dotazovacích jazykov (napr. XPath, JsonPath, atď.) a zároveň sa v ňom dajú vyjadriť všetky hlavné funkcionality jazykov používaných na transformáciu dokumentov v praxi. Doterajšie jazyky buď nemali formálne definovanú sémantiku, alebo mali obmedzenú vyjadrovaciu silu. Syntax jazyka X-Fun uvádzame na Obr. 1.

Súčasťou prácu su nové kompilátory z fragmentov jazykov Xslt, XQuery a XProc do nášho jazyka X-Fun, ktoré sme navrhli a implementovali. Vďaka týmto kompilátorom sa z jazyka X-Fun stáva platforma, ktorá zjednocuje všetky fázy spracovania Xml dát (dotazovanie databáz—XQuery, formátovanie výstupu—Xslt, skladanie častí procesu do celku—XProc). Naprogramovaním jedného procesora transformácií definovaných v X-Fun získavame procesor, ktorý dokáže spracovať transformácie definované v hociktorom z uvedených troch jazykov.

Kompilátor z Xslt a XQuery, ako aj stromový algoritmus na vyhodnocovanie X-Fun programov, sme implementovali v jazyku Java. Rýchlosť prekladu našej implementácie sme experimentálne porovnávali s existujúcimi imple-

mentáciami. Porovnanie s implementáciou jazyku XProc sme robili pomocou manuálne preložených programov, kým pri Xslt a XQuery sme použili náš kompilátor. Výsledkom je, že naša implementácia má porovnateľnú rýchlosť s existujúcimi, a v prípade XProc je dokonca rýchlejšia.

Novinkou v jazyku X-Fun je aj úplné oddelenie jazyku na navigáciu vstupným stromom od jazyku na generovanie výstupu. Navigačný jazyk je parametrom X-Fun a vďaka tomu X-Fun môže byť použitý na preklad nielen Xml dokumentov (kde ako navigačný jazyk volíme XPath), ale aj Json dát (JsonPath) a iných.

5 Prúdové spracovanie X-Fun a zovšeobecnenia

Posledným prínosom práce je navrhnutie prúdového algoritmu na spracovanie programov v jazyku X-Fun. Pri tvorbe algoritmu sme zmiernili požiadavku na buffrovanie, a namiesto úplného zákazu sme sa riadili pravidlom z [36], kde sa použitá pamäť porovnáva s pamäťovými nárokmi optimálneho prúdoveho algoritmu spracujúceho rovnakú trasformáciu. Pamäťové nároky sme ešte viac znížili zavedením nového konceptu hyper-prúdového spracovania dát. Pri hyper-prúdovom spracovaní je výstupný prúd možno dynamicky rozstrihať na podprúdy, ktoré sú prepojené pomocou synchronizačných premenných. Pri takomto spracovaní sa znižujú pamäťové nároky preto, že zaniká potreba buffrovať výstup z dôvodu, že ešte nie je dostupná predchádzajúca časť výstupu. Dokonca aj transformácie, ktoré typicky nie sú spracovateľné prúdovo, napríklad výmena dvoch podstromov, sa môžu ľahko realizovať v hyper-prúdovom režime.

Signifikantnú časť hyper-prúdového algoritmu sme implementovali v jazyku Java a experimentálne sme ju porovnali s existujúcimi prúdovými implementáciami. Vďaka naším kompilátorom z Xslt, XQuery a XProc sme takto získali aj prúdovú implementáciu týchto jazykov. Naša implementácia, použitá ako klasická prúdová implementácia (bez strihania), dosahuje porovnateľné výsledky ako existujúce implementácie, ale pri tom pokrýva omnoho širšiu triedu transformácií (Saxon a Xslt 3.0 špecifikácia podporujú najviac jeden XPath výraz v Xslt šablóne a ten výraz môže obsahovať len *child* a *descendant* osi, kým náš algoritmus neobmedzuje počet výrazov a výrazy môžu používať všetky osi). S použitím strihania sa tieto pamäťové nároky môžu znížiť ešte viac. Tieto výsledky boli prezentované na seminári v Dagstuhle [54].

UNIVERZITA KOMENSKÉHO
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Zoznam publikačnej činnosti

Mgr. Pavel Labath

AFC Publikované príspevky na zahraničných vedeckých konferenciách

AFC01 Labath, Pavel 50% - Rovan, Branislav 50%: Simplifying DPDA using supplementary information
Lit. 4 zázn.
In: Language and Automata Theory and Applications. - Berlin : Springer, 2011. - S. 342-353. - ISBN 978-3-642-21253-6. - (Lecture Notes in Computer Science ; Vol. 6638)
[LATA 2011 : Language and Automata Theory and Applications : International Conference. 5th, Tarragona, 26.-31.5.2011]

AFC02 Labath, Pavel 100%: XSLT streamability analysis with recursive schemas
Lit. 13 zázn.
In: RCIS 2012 : Sixth International Conference on Research Challenges in Information Science Valencia [elektronický zdroj]. - Piscataway : IEEE, 2012. - S. 1-6 [PDF file]. - ISBN 978-1-4577-1938-7
[RCIS 2012 : Research Challenges in Information Science : International Conference. 6th, Valencia, 16.-18.5.2012]
URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6229822

AFG Abstrakty príspevkov zo zahraničných vedeckých konferencií

AFG01 Labath, Pavel 50% - Niehren, Joachim 50%: A functional language for hyperstreaming XSLT
Popis urobený 24.10.2013
Lit. 1 zázn.
In: Dagstuhl Reports [elektronický zdroj]. - Vol. 3, No. 5 (2013), s. 11 [online]
[Tree Transducers and Formal Methods : Dagstuhl Seminar. 13192, Dagstuhl, 2.-7.6.2013]
URL: http://drops.dagstuhl.de/opus/volltexte/2013/4301/pdf/dagrep-v003-i005-complete.pdfhttp://www.dagstuhl.de/mat/Files/13/13192/13192.LabathPavel.Slides.pdf

AFH Abstrakty príspevkov z domácich vedeckých konferencií

AFH01 Labath, Pavel 100%: Zjednodušenie výpočtov prídavnou infomáciou
Lit. 3 zázn.
In: Študentská vedecká konferencia FMFI UK, Bratislava 2010 : Zborník príspevkov. - Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2010. - S. 351. - ISBN 978-80-89186-68-6
[Študentská vedecká konferencia FMFI UK 2010. Bratislava, 28.4.2010]

AFK Postery zo zahraničných konferencií

AFK01 Labath, Pavel 100%: Stream processing of XML transformations
Popis urobený 24.2.2013
In: INQUEST 2012 [elektronický zdroj]. - [Oxford] : [Oxford University], 2012. - 28 slajdov [online]
[INQUEST 2012 : Innovative Querying of Streams : Workshop. Oxford, 14.-28.9.2012]
URL: http://games.cs.ox.ac.uk/inquest12/uploads/Main/labath-inquest12.pdfhttp://games.cs.ox.ac.uk/inquest12/pmwiki.php/Main/Abstracts
POZNÁMKA: Vyšlo aj ako abstrakt - INQUEST 2012 [elektronický zdroj]. - [Oxford] : [Oxford University], 2012. - s. 1 [online].

Štatistika kategórií (Záznamov spolu: 5):
AFC Publikované príspevky na zahraničných vedeckých konferenciách (2)
AFG Abstrakty príspevkov zo zahraničných vedeckých konferencií (1)
AFH Abstrakty príspevkov z domácich vedeckých konferencií (1)
AFK Postery zo zahraničných konferencií (1)

2. 7. 2014

Literatúra

[1] Simple API for XML. http://www.saxproject.org.

[2] Aho, A. V., and Ullman, J. D. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[3] Alon, N., Milo, T., Neven, F., Suciu, D., and Vianu, V. XML with data values: typechecking revisited. *J. Comput. Syst. Sci. 66*, 4 (2003), 688–727.

[4] Alur, R., and Madhusudan, P. Adding nesting structure to words. *Journal of the ACM 56*, 3 (2009), 1–43.

[5] Arroyuelo, D., Claude, F., Maneth, S., Mäkinen, V., Navarro, G., Nguyen, K., Sirén, J., and Välimäki, N. Fast in-memory xpath search using compressed indexes. In *ICDE* (2010), IEEE, pp. 417–428.

[6] Bar-Yossef, Z., Fontoura, M., and Josifovski, V. On the memory requirements of XPath evaluation over XML streams. *J. Comput. Syst. Sci. 73*, 3 (2007), 391–441.

[7] Becker, O. Transforming XML on the Fly. In *Proceedings of XML Europe* (2003).

[8] Benedikt, M., Fan, W., and Geerts, F. XPath satisfiability in the presence of DTDs. *J. ACM 55*, 2 (2008).

[9] Benedikt, M., and Vu, H. Higher-order functions and structured datatypes. In *15th International Workshop on the Web and Databases* (2012), Z. G. Ives and Y. Velegrakis, Eds., pp. 43–48.

[10] Benzaken, V., Castagna, G., and Frisch, A. CDuce: an XML-centric general-purpose language. *ACM SIGPLAN Notices 38*, 9 (2003), 51–63.

[11] Benzaken, V., Castagna, G., Nguyen, K., and Siméon, J. Static and dynamic semantics of NoSQL languages. In *POPL* (2013), R. Giacobazzi, R. Cousot, R. Giacobazzi, and R. Cousot, Eds., ACM, pp. 101–114.

[12] Berstel, J., and Boasson, L. XML Grammars. In *Mathematical Foundations of Computer Science 2000, 25th International Symposium, Bratislava, Slovakia* (Aug. 2000), M. Nielsen and B. Rovan, Eds., vol. 1893 of *Lecture Notes in Computer Science*, Springer, pp. 182–191.

[13] Berstel, J., and Boasson, L. Formal properties of xml grammars and languages. *Acta Informatica 38*, 9 (2002), 649–671.

[14] Bojanczyk, M., Muscholl, A., Schwentick, T., and Segoufin, L. Two-variable logic on data trees and xml reasoning. *J. ACM 56*, 3 (2009).

[15] Braaksma, A. Exselt: A concurrent streaming processor for XSLT 3.0. http://exselt.net.

[16] Castagna, G., Im, H., Nguyen, K., and Benzaken, V. A Core Calculus for XQuery 3.0. Unpublished manuscript, 2013.

[17] Chen, Y., Davidson, S. B., and Zheng, Y. An Efficient XPath Query Processor for XML Streams. In *Data Engineering* (Apr. 2006), L. Liu, A. Reuter, K.-Y. Whang, and J. Zhang, Eds., IEEE Computer Society, p. 79.

[18] Chen, Y., Davidson, S. B., and Zheng, Y. An Efficient XPath Query Processor for XML Streams. *Data Engineering, International Conference on* (2006), 79.

[19] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., and Tommasi, M. Tree automata techniques and applications. Oct. 2007.

[20] Debarbieux, D., Gauwin, O., Niehren, J., Sebastian, T., and Zergaoui, M. Early Nested Word Automata for XPath Query Answering on XML Streams. In *CIAA* (2013), pp. 292–305.

[21] Dvoráková, J., and Zavoral, F. Xord: An Implementation Framework for Efficient XSLT Processing. In *Intelligent Distributed Computing, Systems and Applications, Catania, Italy* (2008), C. Badica et al., Eds., vol. 162 of *Studies in Computational Intelligence*, Springer, pp. 95–104.

[22] Dvoráková, J., and Zavoral, F. Using Input Buffers for Streaming XSLT Processing. In *Advances in Databases, Knowledge, and Data Applications* (Mar. 2009), Q. Chen et al., Eds., IEEE Computer Society, pp. 50–55.

[23] Dvořáková, J. Automatic Streaming Processing of XSLT Transformations Based on Tree Transducers. *Informatica (Slovenia) 32*, 4 (2008), 373–382.

[24] Dvořáková, J. Towards analyzing space complexity of streaming XML transformations. In *Research Challenges in Information Science* (June 2008), O. Pastor, A. Flory, and J.-L. Cavarero, Eds., IEEE, pp. 447–452.

[25] Dvořáková, J., and Zavoral, F. Schema-Based Analysis of XSLT Streamability. In *Advanced Engineering Computing and Applications in Sciences* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 187–192.

[26] Dvořaková, J. *A Formal Framework for Streaming XML Transformations.* PhD thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, 2010.

[27] Dvořáková, J., and Zavoral, F. A low-memory streaming algorithm for XSLT processing implemented in Xord framework. In *Applications of Digital Information and Web Technologies* (Aug. 2008), pp. 24–29.

[28] Ecma International. The JSON Data Interchange Format. http://ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 2013.

[29] Flanagan, C., and Felleisen, M. The semantics of future and an application. *J. Funct. Program. 9*, 1 (1999), 1–31.

[30] Florescu, D., Hillery, C., Kossmann, D., Lucas, P., Riccardi, F., Westmann, T., Carey, M. J., Sundararajan, A., and Agrawal, G. The BEA/XQRL streaming XQuery processor. In *Very Large Data Bases* (2003), VLDB Endowment.

[31] Franceschet, M. XPathMark: An XPath Benchmark for the XMark Generated Data. In *XSym* (Sept. 2005), S. Bressan et al., Eds., vol. 3671 of *Lecture Notes in Computer Science*, Springer, pp. 129–143.

[32] Frisch, A., and Nakano, K. Streaming XML Transformation Using Term Rewriting. In *Programming Language Technologies for XML, An ACM SIGPLAN Workshop, Nice, France* (Jan. 2007), pp. 2–13.

[33] Fülöp, Z., and Vogler, H. *Syntax-Directed Semantics: Formal Models Based on Tree Transducers*, 1 ed. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[34] Galiegue, F., et al. JSON Schema: Core definitions and terminology. Internet-Draft: http://json-schema.org/latest/json-schema-core.html, Jan. 2013.

[35] Galiegue, F., et al. JSON Schema: interactive and non interactive validation. Internet-Draft: http://json-schema.org/latest/json-schema-validation.html, Jan. 2013.

[36] Gauwin, O., and Niehren, J. Streamable Fragments of Forward XPath. In *Implementation and Application of Automata* (July 2011), B. Bouchou-Markhoff et al., Eds., vol. 6807 of *LNCS*, Springer, pp. 3–15.

[37] Gauwin, O., Niehren, J., and Tison, S. Earliest Query Answering for Deterministic Nested Word Automata. In *Fundamentals of Computation Theory, 17th Symposium* (Sept. 2009), vol. 5699 of *LNCS*, Springer, pp. 121–132.

[38] Gauwin, O., Niehren, J., and Tison, S. Queries on Xml streams with bounded delay and concurrency. *Inf. Comput. 209*, 3 (2011), 409–442.

[39] Gauwin, O., Niehren, J., and Tison, S. Queries on XML Streams with Bounded Delay and Concurrency. *Information and Computation 209*, 3 (Mar. 2011), 409–442.

[40] Gottlob, G., and Koch, C. Monadic queries over Tree-Structured data. In *17th Annual IEEE Symposium on Logic in Computer Science* (Copenhagen, 2002), pp. 189–202.

[41] Gou, G., and Chirkova, R. Efficient Algorithms for Evaluating XPath over Streams. In *SIGMOD Conference on Management of Data, Beijing, China* (June 2007), C. Y. Chan, B. C. Ooi, and A. Zhou, Eds., ACM, pp. 269–280.

[42] Guo, Z., Li, M., Wang, X., and Zhou, A. Scalable XSLT Evaluation. In *Advanced Web Technologies and Applications* (Apr. 2004), J. X. Yu, X. Lin, H. Lu, and Y. Zhang, Eds., vol. 3007 of *LNCS*, Springer, pp. 190–200.

[43] Gupta, A. K., and Suciu, D. Stream Processing of XPath Queries with Predicates. In *SIGMOD Conference on Management of Data* (June 2003), A. Y. Halevy, Z. G. Ives, and A. Doan, Eds., ACM, pp. 419–430.

[44] Hakuta, S., Maneth, S., Nakano, K., and Iwasaki, H. XQuery Streaming by Forest Transducers. In *30th IEEE International Conference on Data Engineering* (2014), IEEE Computer Society.

[45] Hakuta, S., Maneth, S., Nakano, K., and Iwasaki, H. XQuery streaming by Forest Transducers. In *30th International Conference on Data Engineering* (Apr. 2014), I. F. Cruz et al., Eds., IEEE, pp. 952–963.

[46] Halstead, R. H. Multilisp: A language for concurrent symbolic computation. *ACM Trans. Program. Lang. Syst. 7*, 4 (Oct. 1985), 501–538.

[47] Innovimax. QuiXProc. http://www.quixproc.com.

[48] Kay, M. SAXON: XSLT and XQuery processor. http//www.saxonica.com.

[49] Kepser, S. A Simple Proof for the Turing-Completeness of XSLT and XQuery. In *Proceedings of the Extreme Markup Languages Conference, Montréal, Quebec, Canada* (Aug. 2004).

[50] Kirchner, C., Moreau, P. E., and Tavares, C. A type system for tom. In *RULE* (2009), I. Mackie, A. M. Moreira, I. Mackie, and A. M. Moreira, Eds., vol. 21 of *EPTCS*, pp. 51–63.

[51] Kuikka, E., and Penttonen, M. Transformation of structured documents with the use of grammar. *Electronic Publishing 6*, 4 (1993), 373–383.

[52] Labath, P. Stream processing of xml transformations. In *Innovative Querying of Streams* (2012), INQUEST, Oxford University.

[53] Labath, P. XSLT streamability analysis with recursive schemas. In *RCIS* (2012), C. Rolland, J. Castro, and O. Pastor, Eds., IEEE, pp. 1–6.

[54] Labath, P., and Niehren, J. A functional language for hyperstreaming XSLT. In *Tree Transducers and Formal Methods* (2013), vol. 3, Dagstuhl Reports, p. 11.

[55] Lille, I., and Innovimax. FXP and QuiXPath tools for XML streams, 2012. http://fxp.lille.inria.fr/.

[56] Ludäscher, B., Mukhopadhyay, P., and Papakonstantinou, Y. A transducer-based XML query processor. In *Proceedings of the 28th conference on Very Large Data Bases* (2002), VLDB Endowment, pp. 227–238.

[57] Maneth, S., Berlea, A., Perst, T., and Seidl, H. XML type checking with macro tree transducers. In *SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (2005), C. Li, Ed., ACM, pp. 283–294.

[58] Martens, W., and Neven, F. Typechecking Top-Down Uniform Unranked Tree Transducers. In *Database Theory - ICDT* (2003), D. Calvanese, M. Lenzerini, and R. Motwani, Eds., vol. 2572 of *LNCS*, Springer, pp. 64–78.

[59] Milner, R. A proposal for standard ml. In *Symposium on LISP and Functional Programming* (New York, NY, USA, 1984), ACM, pp. 184–197.

[60] Mlynkova, I., Toman, K., and Pokorny, J. Statistical Analysis of Real XML Data Collections. In *Proceedings of the 13th Conference on Management of Data* (2006), Tata McGraw-Hill Publishing Company Limited, pp. 20–31.

[61] Murata, M. Hedge automata: a formal model for XML schemata. `http://xml.coverpages.org/hedge20000224.html`, 1999.

[62] Murata, M., Lee, D., Mani, M., and Kawaguchi, K. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol. 5* (November 2005), 660–704.

[63] Nakano, K., and Mu, S.-C. A Pushdown Machine for Recursive XML Processing. In *Programming Languages and Systems*, N. Kobayashi, Ed., LNCS. Springer Berlin Heidelberg, 2006, pp. 340–356.

[64] Neumann, A., and Seidl, H. Locating matches of tree patterns in forests. In *18-th Conference on Foundations of Software Technology and Theoretical Computer Science* (1998).

[65] Niehren, J., Schwinghammer, J., and Smolka, G. A concurrent lambda calculus with futures. *Theoretical Computer Science 364*, 3 (Nov. 2006), 338–356.

[66] Nizar, A., and Kumar, P. S. Efficient Evaluation of Forward XPath Axes over XML Streams. In *Management of Data* (Dec. 2008), G. Das et al., Eds., Computer Society of India / Allied Publishers, pp. 222–233.

[67] Olteanu, D., Kiesling, T., and Bry, F. An Evaluation of Regular Path Expressions with Qualifiers against XML Streams. In *Data Engineering* (Mar. 2003), U. Dayal et al., Eds., IEEE Computer Society, pp. 702–704.

[68] Olteanu, D., Meuss, H., Furche, T., and Bry, F. XPath: Looking Forward. In *XML-Based Data Management and Multimedia Engineering* (Mar. 2002), A. B. Chaudhri et al., Eds., vol. 2490 of *LNCS*, Springer, pp. 109–127.

[69] Peng, F., and Chawathe, S. S. XPath Queries on Streaming Data. In *International Conference on Management of Data, San Diego, USA* (June 2003), A. Y. Halevy, Z. G. Ives, and A. Doan, Eds., ACM, pp. 431–442.

[70] Peyton Jones, S. L. *The Implementation of Functional Programming Languages*. Prentice-Hall, 1987.

[71] Schmidt, M., Scherzinger, S., and Koch, C. Combined static and dynamic analysis for effective buffer minimization in streaming XQuery evaluation. In *23rd Conference on Data Engineering* (2007), IEEE, pp. 236–245.

[72] Smolka, G. Objects in a higher-order concurrent constraint model with state. In *LMO* (1995), A. Napoli and A. Napoli, Eds., INRIA, pp. 69–74.

[73] Stefan Gössner. JSONPath - XPath for JSON. `http://goessner.net/articles/JsonPath/`, 2007.

[74] W3C. *Document Object Model (DOM) Level 1.* W3C Recommendation, 1998. http://www.w3.org/TR/REC-DOM-Level-1.

[75] W3C. *XML Path Language (XPath) Version 1.0.* W3C Recommendation, 1999. http://www.w3.org/TR/xpath.

[76] W3C. *XSL Transformations (XSLT) Version 1.0.* W3C Recommendation, 1999. http://www.w3.org/TR/xslt.

[77] W3C. *XML Schema Part 2: Datatypes*, 2 ed. W3C Recommendation, 2004. http://www.w3.org/TR/xmlschema-2.

[78] W3C. *XML Schema Part 2: Structures*, 2 ed. W3C Recommendation, 2004. http://www.w3.org/TR/xmlschema-1.

[79] W3C. *Extensible Stylesheet Language (XSL) Version 1.1.* W3C Recommendation, 2006. http://www.w3.org/TR/xsl.

[80] W3C. *Extensible Markup Language (XML) 1.0*, 5 ed. W3C Recommendation, 2008. http://www.w3.org/TR/xml.

[81] W3C. *Namespaces in XML 1.0*, 3 ed. W3C Recommendation, 2009. http://www.w3.org/TR/REC-xml-names.

[82] W3C. Requirements and Use Cases for XSLT 2.1, 2010. `http://www.w3.org/TR/xslt-21-requirements/`.

[83] W3C. *XML Syntax for XQuery 1.0 (XQueryX)*, 2 ed. W3C Recommendation, 2010. http://www.w3.org/TR/xqueryx.

[84] W3C. *XQuery 1.0: An XML Query Language*, 2 ed. W3C Recommendation, 2010. http://www.w3.org/TR/xquery.

[85] W3C. *XPath and XQuery Functions and Operators 3.0.* W3C Working Draft, Dec. 2011. http://www.w3.org/TR/xpath-functions-30.

[86] Walsh, N. XML Calabash. http://xmlcalabash.com.

[87] Walsh, N., Milowski, A., and Thompson, H. S. XProc: An XML Pipeline Language, May 2010.

[88] Zhang, Y., Cao, Y., and Li, X. A Decision Procedure for XPath Satisfiability in the Presence of DTD Containing Choice. In *Progress in WWW Research and Development* (2008), vol. 4976 of *LNCS*, Springer, pp. 202–213.