BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **16**, No 4
Sofia • 2016 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2016-0066

An Approach of XML Query Evaluation Based Model Checking

Li Yan-Mei, Huang Shao-Bin, Li Ya, Xu Li

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
Emails: liyanmei@hrbeu.edu.cn huangshaobin@hrbeu.edu.cn liya@hrbeu.edu.cn Xuli@hrbeu.edu.cn

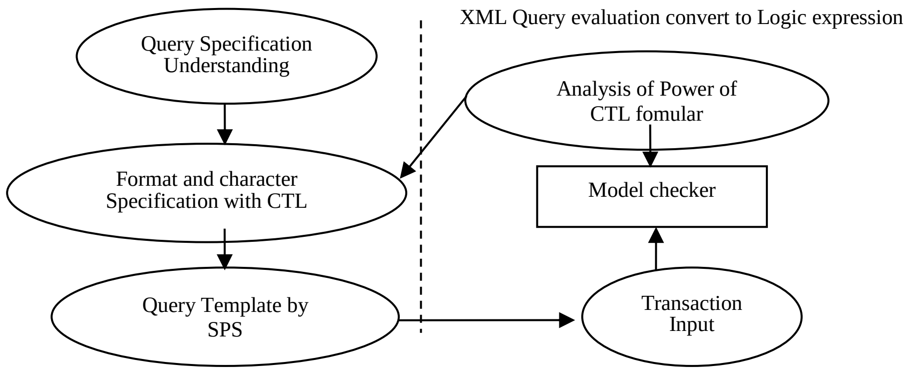
Abstract: *In this paper, we show the process inspired by model checking which integrate temporal logic to the application of semi-structured data query. We investigate the potential of a technique based on CTL (Computation Tree Logic) model checking for evaluating queries expressed in (a subset of) XPath. Our research consists of query algebra, constraint understanding and expression mapping. The core of research is mapping the XML query algebra to an expression collection of temporal logic. We try a new kind of query execution strategy to enhance the accuracy of semantic description of the XML query. For the purpose of supporting the generation of the formal specifications and reducing the mapping processing, the XML query constraint can be converted to a specification of SPS (Specification Pattern System) through which we get the formula set to evaluate path queries directly on CTL formula.*

Keywords: *XML, model checking, Xpath, temporal logic, SPS.*

1. Introduction

Twigs pattern is a common XML query strategy whose core ideas are decomposition and connection. We convert the query to binary relations contained child-parent relationship or ancestors-descendents, and merge the structural results with a connected algorithm after matching process. The relationship between XQuery and temporal logic has been investigated in [1, 2, 9]. The familiar correspondences between logic and XML languages are automata and regular language [3]. By using tree automata, pushdown automata, register automata and pebble automata, some researchers anaylzed the nature of XML language and established the contact between the sorts of automata and validate XML [4, 8]. In the aspect of temporal logic, we drew the conclusion that a query can be stated in query language Q if and only if it can be expressed in logic L. Of course, these results always refer to reasonably abstracted versions and precisely defined subsets of query or schema languages. In [9], it was shown that a node-selecting query can be stated in the navigational core of XPath if and only if it can be expressed in two-variable logic which restrict the formulas of the first-order logic with two variables.

Some researchers considered that the query in semi-structure data like XML might be regarded as the constraint satisfiability problem whose constraint specification is presented expression of temporal logic. In [5, 7], they processed the subset of Xpath which included vertical axis and label assignment with the temporal operator {true, ∧, EXϕ, EFϕ}. They proved that the tree patterns in XML query can be translated to the formula of Computation Tree Logic (CTL). The complexity of the problem of query containment is serious in [6,10]. So, in this paper, we try to figure out three problems in XML query evaluation based on the temporal logic. Firstly, one of the problems is how to analyze and characterize the expressiveness of XPath algebra by temporal logic. Secondly, we need to identify the restrictions understanding. Another aspect is the reduction of expression of temporal logic concerned about Xpath algebra. Finally, we applied the SPS (Specification Pattern System) to the translation and the study diagram is as the figure1. We devise a linear translation of the query evaluation for Simple XPath into model checking for CTL. The translation is sound: a query matches a XML document if and only if the translated formula matches the XML model.


XML Query evaluation convert to Logic expression
Query Specification Understanding
Analysis of Power of CTL fomular
Format and character Specification with CTL
Model checker
Query Template by SPS
Transaction Input
Fig. 1. Query evaluation with model checking based Computation tree logic

Compared to the popular use of formal verification techniques in software engineers, it is imperative that automatically generate complex formal specifications in XML query instead of manually. Through by SPS and Prospec we automatically generate CTL formulas for character description in XPath. We use formal proofs to validate the correctness of the templates in XML query. We additional describes a novel approach to validating CTL formulas using model checker. Formal method make us to be confident that the formal specifications accurately reflect the intended query properties.

2. Computation tree logic and XML represents by SPS

In this section, we introduce the formal description and definition of model checking and the syntax and the semantics of Computation tree logic. Then we present a model-checking algorithm for CTL. Firstly, the CTL is formated as clearly tree structure, and their Backus-Naur normal form is as following.

$$\Phi ::= \perp |T|p|\neg\Phi|\Phi \wedge \Phi|\Phi \vee \Phi|\Phi \rightarrow \Phi|AX\Phi|EX\Phi|$$
$$AF\Phi|EF\Phi|AG\Phi|EG\Phi|A[\Phi U\Phi]|E[\Phi U\Phi].$$

CTL formulae is divided into state formulae and path formulae. Intuitively, state formulae express a property of a state, while path formulae expresses a property of a path, i.e., an infinite sequence of states, where *P* is the set of atomic proposition, $\perp$ is absurdity and *T* presents tautology. *A* and *E* are path quantifiers. *A* is "along all the path" and *E* is "at least one path". X, F, G, U is the temporal operators. X pronounced "next" and U pronounced "until". The until operator allows to derive the temporal modalities F and G, F is "eventually", sometimes in the future and G is "always", from now on forever. $F\varphi \triangleq \text{True } U\varphi$, and $G\varphi \triangleq \neg F \neg\varphi$.

Definition 2.1. The syntax of CTL formula, where φ is CTL formula, for state situation the satisfy relation is as follows:

$s| = a$ iff $a \in L(s)$
$s| = \neg\Phi$ iff $s| = \Phi$ is false
$s| = \Phi \wedge \Psi$ iff $(s| = \Phi)$ and $(s| = \Psi)$
$s| = \exists\varphi$ iff $\pi| = \Phi,$ exist $\pi \in \text{Paths}(s)$
$s| = \forall\varphi$ iff $\pi| = \Phi$ for all $\pi \in \text{Paths}(s)$.

For a Path π the satisfy relation =is design as follows.

$\pi| = X\Phi$ iff $\pi[1]| = \Phi$
$\pi| = \Phi \cup \Psi$ iff $\exists j \geq 0.(\pi[j]| = \Psi \wedge (\forall 0 \leq k < j.\pi[k]| = \Phi))$.

where, the path$\pi = s_0 s_1 s_2 ...$ and the integer $i \geq 0$, $\pi[i]$ presents the (i+1)-th element of π, that is $\pi[i] = s_i$.

Definition 2.2. CTL formula semantics

Let $\text{Sat}(\Phi)$ is the maches nodes for CTL state formula Φ,

$$\text{Sat}(\Phi) = \{s \in S|s| = \Phi\}.$$

For path $\pi = s_0 s_1 s_2 ...$:

$\pi| = F\Phi$ iff $s_j| = \Phi$, $\exists j \geq 0$.

These induced that:

$s| = \exists G\Phi$ iff $\exists\pi \in \text{Paths}(s).\pi[j]| = \Phi\ \forall j \geq 0$,
$s| = \forall G\Phi$ iff $\forall\pi \in \text{Paths}(s).\pi[j]| = \Phi\ \forall j \geq 0$.

Therefor, $G\Phi$ may viewed as the path formula with sementic:

$\pi = s_0 s_1 s_2 ...| = G\Phi$ iff $\forall j \geq 0, s_j| = \Phi$.

In the same way, it can conclude that $\exists G\Phi$, $\exists F\Phi$, $\forall F\Phi$.

That CTL-model checking can be performed by a recursive procedure that calculates the satisfaction set for all subformulae of Φ.

(a) $\text{Sat}(\text{true}) = S$
(b) $\text{Sat}(a) = \{s \in S \mid a \in L(s)\}$, for any $p \in P$,
(c) $\text{Sat}(\Phi \wedge \Psi) = \text{Sat}(\Phi) \cap \text{Sat}(\Psi)$,
(d) $\text{Sat}(\neg\Phi) = S \setminus \text{Sat}(\Phi)$,
(e) $\text{Sat}(\exists X\Phi) = \{s \in S \mid \text{Post}(s) \cap \text{Sat}(\Phi) \neq \emptyset\}$,
(f) $\text{Sat}(\exists(\Phi U\Psi))$is the smallest subset T of S, such that

$\text{Sat}(\Psi) \subseteq T$ and $s \in \text{Sat}(\Phi)$ and $\text{Post}(s) \cap T \neq \emptyset$ implies $s \in T$.

(g) $\mathrm{Sat}(\exists \mathrm{G}\Phi)$ is the largest subset T of S, such that $T \subseteq \mathrm{Sat}(\Phi)$ and $s \in T$ implies $\mathrm{Post}(s) \cap T \neq \emptyset$.

(h) $\mathrm{Sat}(\forall \mathrm{X}\Phi) = \{s \in S \mid \mathrm{Post}(s) \subseteq \mathrm{Sat}(\Phi)\}$.

(i) $\mathrm{Sat}\big(\forall(\Phi \mathrm{U} \Psi)\big)$ is the smallest set $T \subseteq S$ satisfying

$$\mathrm{Sat}(\Psi) \cup \{s \in \mathrm{Sat}(\Phi) \mid \mathrm{Post}(s) \subseteq T\} \subseteq T.$$

(j) $\mathrm{Sat}(\forall \mathrm{G}\Phi)$ is the largest set $T \subseteq S$ satisfying

$$T \subseteq \{s \in \mathrm{Sat}(\Phi) \mid \mathrm{Post}(s) \subseteq \mathrm{T}\}.$$

In [11], they provided patterns and scopes to assist the practitioner in formally specifying software properties with SPS. Each pattern describes the structure of specific behavior and defines the pattern's relationship with other patterns. In SPS, each pattern is associated with a scope. There are five types of scopes defined in SPS: Global, Before *R*, After *L*, Between *L* And *R*, and After *L* Until *R*.

Table 1. New CP class and XPath semanstic

New CP Class	**CTL Description for XPath Character**	**XML Query Semantic**
AtLeastOne	*Cluster and Single Node*	
AtLeastOne*C*	$EA, a_i \in A$	*Matching nodes(key-words)*
AtLeastOne*H*	$E\mathrm{F}A, a_i \in A$	*Matching path rooted the initial node*
Parallel	*Cluster*	
Parallel*C*	*AFA*	*Small Lowest Common Ancestor*
Parallel*H*	$E\ (\mathrm{F}a_1 \wedge \mathrm{F}a_2 \ldots \wedge \mathrm{F}a_n)$	*Lowest Common Ancestor*
Consecutive	*Continue parent-child Path*	
ConsecutiveC	$(a_1 \wedge \mathrm{X}(a_2 \wedge (\ldots(\wedge \mathrm{X}a_n))\ldots))$	*Strict parent-child sequence*
Consecutive*H* (*S*)	$(\neg S) \wedge (S_1 \wedge \mathrm{X}(S_2 \wedge (\ldots(\wedge \mathrm{X}S_n))\ldots))$	*Non-strict parent-child*
Eventual	*Continue ancestor-descendent Path*	
Eventual*C*	$(a_1 \wedge \mathrm{X}(\neg a_2 \mathrm{U}(a_2 \wedge \mathrm{X}(\ldots\wedge \mathrm{X}(\neg a_{n-1}\mathrm{U}(a_{n-1} \wedge \mathrm{X}(\neg a_n \mathrm{U} a_n))))\ldots)))$	*Strict ancestor-descendent sequence*
Eventual*H*(*S*)	$(\neg S) \wedge (S_1 \wedge \mathrm{X}(\neg S_2 \mathrm{U}(S_2 \wedge \mathrm{X}(\ldots\wedge \mathrm{X}(\neg S_{n-1}\mathrm{U}(S_{n-1} \wedge \mathrm{X}(\neg S_n \mathrm{U} S_n))))\ldots)))$	*Non-strict ancestor-descendent*

1) *Universality*(*P*): Only the states that have the desired property (*P*). Also known as Henceforth and Always.

2) *Absence*(*P*): Free of certain event or state (*P*).

3) *Existence*(*P*): Contains an instance of certain events or states (*P*). Also known as Eventually.

4) *Precedence*(*P*, *Q*): To describe relationships between a pair of events/states where the occurrence of the first (*Q*) is a necessary pre-condition for an occurrence of the second (*P*). We say that an occurrence of the second is enabled by an occurrence of the first.

5) *Response* (*P*, *Q*): To describe cause-effect relationships between a pair of events/states. An occurrence of the first (*P*), the cause, must be followed by an occurrence of the second (*Q*), the effect.

3. XML and query constraint

For the sake of understanding the constraint about Xpath, we model the XML data with the new introduce definition in the view of temporal logic description.

Definition 3.1. Let XML module $M = (E,\ R)$. Where E is the elements set, R presents the edge related to the elements.

In the above definition, R actually reflect the relationships the nested structure of XML and classified with $R_{\downarrow}$and $R^{\uparrow}$ which respectively represents children and parents direction, e.g., $R_{\downarrow}{}^{*}$X ($R^{\uparrow^{*}}$X) means that it recursively get the parent(child) node, until it is empty, and X may be nodes or paths.

Definition 3.2. XML Instance $I = (V,\ \gamma,\ n,\ \text{Cons},\ \text{Alph})$, where V is the set of nodes, and γ represents the instance edges, n is the root. That $\text{Cons}: V \to 2^{V}$ presents the mapping between nodes and $\text{alph}(t) = E$ means that Albert for instance *I*.

Definition 3.3. Let path $h(v_1) = \langle v_1, v_2, \ldots, v_n \rangle$, where $v_i \in V$, *i*=1, …, *n*, satisfied condition:

(1) $\forall i,\ \ i = 1, \ldots, n,\ \exists v_i \to v_{i+1}$, and $v_{i+1} \in \text{cons}(v_i)$;

(2) v_n is the leaf of the path;

(3) $v_i \in V,\ \ i = 1, \ldots, n-1,$ means not branch or predicate.

Definition 3.4. Let path $h(v_1) = \langle v_1, v_2, \ldots, v_n \rangle$, $A(v_l) = a_1$, $A(v_{l+1}) = a_2, \ldots, A(v_{l+m}) = a_m$, $l = 1, \ldots, n,\ m < n$. We called $T(h)$ as the strict label trace of h and denoted by $A(a_1 a_2 \ldots a_n) \subseteq T(h)$. For the path $h(v_1)$ and $A(v_l) = a_1,\ A(v_{i>l}) = a_2,\ A\big(v_{j>i}\big) = a_3,\ i, j, l = 1, \ldots, n$, we called $\hat{T}(h)$ unstrict label trace of *h* and denoted by $A(a_1, a_2, \ldots, a_m) \subseteq \hat{T}(h)$. In the same way, let $L = \{a_1, a_2, \ldots, a_m\}$ and *s* presents all the string on *L*.

Definition 3.5. Let $H(V^H, T^H, L(r))$, where V^H presents set of nodes and T^H is the string collection based $L = \{a_1, a_2, \ldots, a_m\}$ and it satisfied conditions:

1) $T^H \subseteq \tilde{T}(h),\ T^H = \{s\}$ is all sort on alphabet L;

2) $T^H \subseteq \tilde{T}(h),\ \ s_i \in T^H$, the string s_i covered character r and all the element of set L/r.

3) Meet the condition (1) or meet (2);

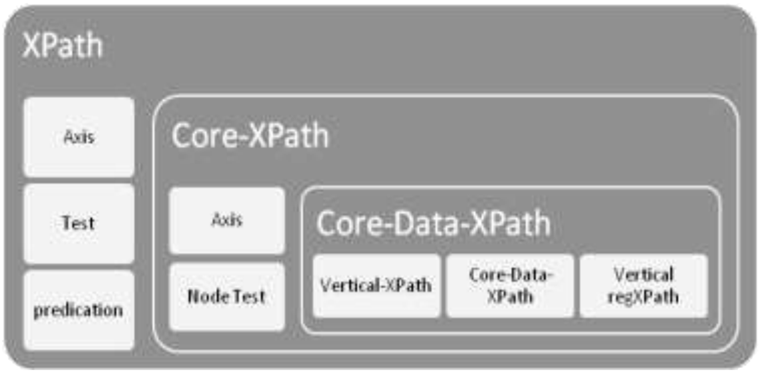
4) Exist no untrivial nodes, $H(V^H, T^H, L(r))$ is path connected.

Next definition gives the definition of continued parent-child path which have no nodes test and predicates excepted the last node.

Definition 3.6. Let $h(v_1) = \langle v_1, v_2, \ldots, v_n \rangle$, it means the each labels of path nodes from top to bottom is exactly $a_1 a_2 \ldots a_n$. Let $A(a_1, a_2, \ldots, a_m) \subseteq \hat{T}(h)$, it means that each labels of path nodes from top to bottom included in $a_1 a_2 \ldots a_n$. $S \subseteq \tilde{T}(h)$ expands that above two situation and it means that the trace of the path is on the alphabet $L = \{a_1, a_2, \ldots, a_n\}$ no matter its sort.

In the Definition 3.5, it reflects that a cluster of result which either a linear path with no branch or r is its joint and the braches covered all the element of alphabet L.

XPath query which expressed long path structure as the main characteristics covers a complex branching structure. Based on XPath, Xquery queries integrate the nested structure of programming which adds the conditional statement. Fig. 2 shows the XPath syntax structure which is the intermediate products in XML query researches. Core-XPath is a fragment of XPath, which only covers the navigation part, but not express any attribute contained data; It is the core of the XPath language which is able to keep the navigation of a language and give up the arithmetic and string operations. Core-Data-XPath is an expansion of the Core-XPath language which contains equal and unequal test concerned Data, and it is undecidable. Vertical-XPath* defines the fragments of Core-Data-XPath, which applies only forward axis like child and descendant and reverse axis like parent and ancestor, which do not allow the siblings axis and permit any XPath expression of asterisks *.



Node Constraint $p::= p \wedge p \mid p \vee p \mid \neg p \mid h(p)$

Path Constraint $h ::= r \mid r/h\,,\ r \in R\{R^{\uparrow}, R_{\downarrow}\}$

$r = r::l \mid r::l\,[p]$

Semantics for M $[[R^{\uparrow}]]^{M} = \{r \mid r \in R^{\uparrow},\}$

$[[R_{\downarrow}]]^{M} = \{r \mid r \in R_{\downarrow},\}$

$[[r/h]]^{M,x} = \{y \mid \exists k.k \in [[r]]^{M,x}, y \in [[h]]^{M,x}\}$

$[[p1 \wedge p2]]^{M,x} = [[p1]]^{M,x} \cap [[p2]]^{M,x}$

$[[p1 \vee p2]]^{M,x} = [[p1]]^{M,x} \cup [[p2]]^{M,\mathrm{x}}$

$[[\neg p]]^{M,x} = X \setminus [[p]]^{M,x}$

$[[R^{\uparrow}\,[l]]]^{M,x} = \{y \mid (x,y) \in [R^{\uparrow}]^{M}, y \in L(l)\}$

$[[R_{\downarrow}\,[l]]]^{M,x} = \{x \mid (x,y) \in [R_{\downarrow}]^{M}, x \in L(l)\}$

$[[\mathrm{r} :: l[p]]]^{M,x} = \{y \mid (x,y) \in [\mathrm{axis}]^{M},\ y \in L(l),\ [[p]]^{M}, y \neq \emptyset\}$

Fig. 2. XPath schematic diagram and recursive syntax and semantics

4. Embedding query constraint into CTL

In practical applications, we often need to describe properties where one or more patterns or scope parameters are made of multiple propositions, i.e., Composite Propositions (CP). To describe such patterns, Mondragon and Gates (2004) extended SPS by introducing a classification for defining sequential and concurrent behavior to describe pattern and scope parameters. Specifically, the work formally described several types of CP classes and provided their formal descriptions. CP classes are categorized to be either of condition type (denoted with a subscript *C*) or event type (denoted with a subscript *H*). They respectively defined eight CP classes to describe sequential and concurrent behavior. A condition is a proposition that holds over multiple consecutive states, where an event represents a change in the truth value of a proposition in two consecutive states.

The four CP classes of condition type are defined as follows:

$\mathrm{AtLeastOne}_C(a_1,\cdots,a_n)$**,** $H|\mathrm{root}(V), \forall v \in V^H, \exists h(v), T^H \subseteq T(h),$

$$T^H = a_i, 1 \leq i \leq m, L = \{a_1, a_2, .., a_m\}$$

$$\mathrm{Parallel}_C(a_1,\cdots,a_n),$$

$$H|\mathrm{root}(V), \forall h(v_i) \in H, \sum_{i=1}^{n} h(v_i) = L, L = \{a_1, a_2, \ldots . a_n\},$$

$\mathrm{Consecutive}_C(a_1,\cdots,a_n)$**,**

$$H|\mathrm{root}(V), \forall h(v_i) \in H, v_i \in V^H, A(a_1 a_2, \ldots, a_n) \subseteq T^H,$$

$\mathrm{Eventual}_C(\{a_1,\cdots,a_n\})$**,** $\exists H, \forall h(v_i) \in H, v_i \in V^H, A(a_1 a_2, \ldots, a_n) \subseteq \hat{T}(h).$

The four CP classes of type event are as follows:

1. $\mathrm{AtLeastOne}_H(a_1,\cdots,a_n)$ and $\mathrm{AtLeastOne}_C(a_1,\cdots,a_n)$, $\mathrm{Parallel}_H(\{a_1,\cdots,a_n\})$ and $\mathrm{Parallel}_C(a_1,\cdots,a_n)$ is as the same.

2. $\mathrm{Consecutive}_H(L)$ for
$H(V^H,T^H)$: $H(V^H,T^H), \forall h(v_i) \in H, v_i \in V, L = \{a_1, a_2, \ldots, a_n\}, S \subseteq T(h).$

3. $\mathrm{Eventual}_H(L)$ for
$H(V^H,T^H)$: $H(V^H,T^H), \forall h(v_i) \in H, v_i \in V, L = \{a_1, a_2, \ldots, a_n\}, S \subseteq \hat{T}(h).$

Within global scope, the pattern of *Absence of P* presents as $A\mathrm{G}\neg P^{\mathrm{CTL}}$, the pattern of *Existence of P* presents as $A\mathrm{F}P^{\mathrm{CTL}}$, and the other patterns is as the Table 2.

Table 2. Template CTL formulas for patterns within Global scope for Xpath

Pattern	CTL Formula	Pattern	CTL Formula
Q responds to *P*	$\mathrm{G}(P^{\mathrm{CTL}} \rightarrow (P^{\mathrm{CTL}} \wedge_l \mathrm{F}Q^{\mathrm{CTL}}))$	*Q* Precedes P_{C*}	$\neg[(\neg Q^{\mathrm{CTL}})\mathrm{U}(P^{\mathrm{CTL}} \wedge \neg Q^{\mathrm{CTL}})]$
Q strict recedes P_C	$\neg\left[\left(\neg(Q^{\mathrm{CTL}} \wedge_r \neg P^{\mathrm{CTL}})\right)\mathrm{U}P^{\mathrm{CTL}}\right.$	*Q* Precedes P_{C+}	$\neg[(\neg(Q^{\mathrm{CTL}} \wedge_{\bar{l}} \neg P^{\mathrm{CTL}}))\mathrm{U}P^{\mathrm{CL}}$
Q strict recedes P_E	$\neg\left[\left(\neg\left(Q^{\mathrm{CTL}} \wedge_r \neg(\neg A \wedge \mathrm{X}{P_H}^{\mathrm{CTL}})\right)\right)\mathrm{U}(\neg A \wedge \mathrm{X}{P_H}^{\mathrm{CTL}})\right]$		
Q precedes P_{E*}	$\neg[(\neg(Q^{\mathrm{CTL}} \wedge \neg(\neg A \wedge \mathrm{X}{P_H}^{\mathrm{CTL}})))\mathrm{U}(\neg A \wedge \mathrm{X}{P_H}^{\mathrm{CTL}} \wedge \neg Q^{\mathrm{CTL}})]$		
Q precedes P_{E+}	$\neg[(\neg(Q^{\mathrm{CTL}} \wedge_{\bar{l}} \neg(\neg A \wedge A\mathrm{X}{P_H}^{\mathrm{CTL}})))\mathrm{U}(\neg A \wedge \mathrm{X}{P_H}^{\mathrm{CTL}})]$		

Table 3. Template CTL formulas for patterns within global scope for XPath forward axis

Forward axis	XPath	Mapping CTL formula
Child::	Selected nodes	
	XPath"=p /child:: *q*" (*p*/*q*)	$\Phi_{\text{child}}(p,q) = p \wedge EXq, P = R(p), Q = R(q)$
	XPath"=*p*1/*p*2/…/*pn*/ child:: *Q*"	$\Phi_{\text{child}}(P) = {P_c}^{\text{CTL}} \wedge_l EX{Q_c}^{\text{CTL}}$ $P = R(p_1, p_2, \ldots, p_n), Q = R(q_1, q_2, \ldots, q_n)$, $P, Q \in \text{Consecutive}_C$;
Descendant::	The destination node labeled with *P* which ancestors concluded in Q	
	XPath"=*p*/descendant::*q*"	$\Phi_{\text{desc}}(p,q) = E\text{F}q \wedge p, P = R(p)$, $Q = R(q)$
	XPath="*p*1//*p*2//…//*pn*/ descendant::*q*1//*q*2//…//*qn*"	$\Phi_{\text{desc}}(P,Q) = E\text{F}{Q_c}^{\text{CTL}} \wedge_l {P_c}^{\text{CTL}}$ $P = R(p_1, p_2, \ldots, p_n), Q = R(q_1, q_2, \ldots, q_n)$, $P, Q \in \text{Eventual}_C$;
Descendant-or-self::	Descendent node and labeled with *P* and themselves	
	XPath="//*p*1/*p*2/…/*pn* "	$\Phi_{\text{descself}}(p) = P_c, P = R(p), Q = R(q)$
	XPath="*p*1/*p*2/…/*pn*// *q*1/*q*2/…/*qn*"	$\Phi_{\text{descself}}(P,Q) = E\text{F}{Q_c}^{\text{CTL}} \wedge_l {P_c}^{\text{CTL}}$ $P = R(p_1, p_2, \ldots, p_n), \quad Q = R(q_1, q_2, \ldots, q_n)$, $P, Q \in \text{Consecutive}_C$;
Following-sibling::	Destination nodes *Q* which are the preceding of *P*	
	XPath="*p* / following-sibling::*q*"	$\Phi_{\text{flwsibling}}(p,q) = \gamma(EXp) \wedge q \wedge (N_p < N_q)$, $P = R(p), \ Q = R(q)$
	XPath="*p*1/*p*2/…/*pn*/ following-sibling::*Q*"	$\Phi(p,q)_{\text{flwsibling}} =$ $=\gamma(EX{P_c}^{\text{CTL}}) \wedge_r {Q_c}^{\text{CTL}} \wedge_l (N_P < N_Q)P =$ $= R(p_1, p_2, \ldots, p_n)$, $Q = R(q_1, q_2, \ldots, q_n), \ P, Q \in \text{Consecutive}_C$;

In Table 3, we maped the XPath expression to template of CTL formula with SPS and CP. The result may be nodes, nodes set and the path along to root.

5. An applied query instance and validation of CTL templates

For the purpose of evident of a CTL Templates instance for XPath, We applied the CTL templates for Xpath query to model checking. We validate the template from the two respects. On the one hand, the smaller number of templates for the Global was formally proved the correctness. On the other hand the larger number of templates was validated through testing in model checker. Queries of the test are interpreted over an disease XML document, obeying to the following DTD, whose skeleton represents the respiratory disease shown in Fig. 3.

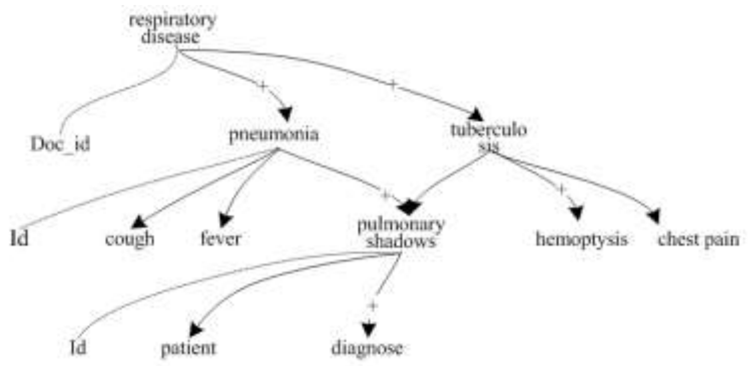


Fig. 3. Generating DTD from an sample XML of the running example

We used the following Simple XPath queries:

Q_1 – /[descendant:: pulmonary shadows]

Q_2 – /[descendant:: tuberculosis/child:: hemoptysis]

Q_3 – //[descendant:: pneumonia[fever]/child:: cough]

Observed that Q_1, Q_2, while Q_3 are the relative path. We apply model checking to validate these query, which gives us a positive answer if the query is successful and a negative answer, otherwise. All the queries are designed to have non-empty answer sets. According to the SPS template for CTL formula, the logical semantic of query Q_1 is "Atleast pulmonary shadows" and the CTL formula is *E* (pulmonary shsdows). Q_2 is continual parent-child and presented by Consecutive(tuberculosis, hemoptysis) and so on.

There is an additional necessary content which is the validation of the defined CTL templates for *X*path. The formal proofs is the main used method to validate the correctness of the templates for patterns within the Global scope and CP classes combinations.

Theorem 1. The CTL formula $\mathrm{G}(P^{\mathrm{CTL}} \rightarrow (P^{\mathrm{CTL}} \wedge_l \mathrm{F}Q^{\mathrm{CTL}}))$ is equivalent to the formal definition of the pattern "Q Responds to P" in Global scope.

Proof: According to the means of "Q responds to P", if P holds at some moment s, then Q holds at some moment s' for which $b_Q(t') \geq e_P(t)$. Formally, we can describe this property as follows:

$$\forall t \left(\mathrm{P}(t) \rightarrow \forall \pi \ \exists t' \left(Q(t') \wedge b_Q(t') \geq e_P(t) \right) \right).$$

The equivalence is proven and hence Theorem 1 is proven.

Theorem 2. The CTL formula $\mathrm{not}\, A[\,\mathrm{not}\, Q^{\mathrm{CTL}}] \cup P^{\mathrm{CTL}} \wedge \mathrm{not}\,(Q^{\mathrm{CTL}})]$ is equivalent to the formal definition of the pattern "Q Precedes P" in Global scope.

Proof: According to the means of "Q responds to P", if Q holds at some moment s, then P holds at some moment s' for which $e_Q(s') \geq b_P(s)$. Formally, we can describe this property as follows:

$$\forall s \left(P(s) \rightarrow \forall \exists s' \left(Q(s') \wedge e_Q(s') \geq b_P(s) \right) \right).$$

The equivalence and Theorem 2 are proven.

6. Conclusion

Through survey of automata and logic motivated by XML, we review the logic imposed by XML. There is a close connection between the query processing problem for XPath and model checking. In this paper, we processed a simple fragment of XPath and established the relationship between XML navigation and temporal logics, in particular CTL and XPath navigation based on regular expressions. We explore the potential of model-checking techniques applied in the field of XML. Here we proposed a technique for combining temporal logics to capture XPath queries expressible in CTL formula with SPS which support the generation of formal specifications. SPS has defined a set of patterns and scopes that allows a query to generate formal specifications by using direct substitution of propositions into parameters of selected patterns and scopes. Tools of Prospec extended SPS to support the definition of patterns and scopes that include the ability to specify parameters with composite propositions(CPs).

R e f e r e n c e s

1. S u r i n x, D., G. H. L. F l e t c h e r et al. Relative Expressive Power of Navigational Querying on Graphs Using Transitive Closure. – Logic Journal of IGPL, Vol. **23**, 2015, No 5, pp. 759-788.
2. Z h a n g, X., J. V. d e n B u s s c h e. On the Power of SPARQL in Expressing Navigational Queries. – The Computer Journal, Vol. **58**, 2015, No 11, pp. 2841-2851.
3. A n t o n o p o u l o s, T., D. H o v l a n d, W. M a r t e n s, F. N e v e n. Deciding Twig-Definability of Node Selecting Tree Automata. – Theory of Computing Systems, Vol. **57**, 2015, No 4, pp. 967-1007.
4. D e b a r b i e u x, D., O. G a u w i n et al. Early Nested Word Automata for XPath Query Answering on XML Streams. – Theoretical Computer Science, Vol. **578**, 2015, pp. 100-125.
5. N i e l a n d t, J., A. B r o n s e l a e r, G. d e T r é. Predicate Enrichment of Aligned XPaths for Wrapper Induction. – Expert Systems with Applications, Vol. **51**, 2016, pp. 259-275.
6. W a n g, Y., B. W a n g, M. L i u. A Component Retrieval Tree Matching Algorithm Based on a Faceted Classification Scheme. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 1, pp. 14-23.
7. Z h e n g, C., Y. Y a o, S. H u a n g, Z. R e n. Modeling Workflow Systems Constrained by Inputs and Outputs – An Approach Based on Petri Nets. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 4, pp. 27-41.
8. G i r e, F., J.-M. T a l b o t. Nested Sibling Tree Automata. – RAIRO – Theoretical Informatics and Applications, Vol. **43**, No 2, 2009, pp. 379-402.
9. L i b k i n, L., C. S i r a n g e l o. Reasoning about XML with Temporal Logics and Automata. – Journal of Applied Logic, Vol. **8**, No 2, 2010, pp. 210-232.
10. K o s t y l e v, E. V., J. L. R e u t t e r, D. V r g o č. Static Analysis of Navigational XPath over Graph Databases. – Information Processing Letters, Vol. **116**, No 7, 2016, pp. 467-474.
11. S a l a m a h, S., A. G a t e s, V. K r e i n o v i c h. Validated Templates for Specification of Complex LTL Formulas. – Journal of Systems and Software, Vol. **85**, No 8, 2012, pp. 1915-1929.