



HAL
open science

CONTRIBUTION À L'ANALYSE DE L'IMPACT DE MISES À JOUR SUR DES VUES ET SUR DES CONTRAINTES D'INTÉGRITÉ XML.

Hicham Idabal

► **To cite this version:**

Hicham Idabal. CONTRIBUTION À L'ANALYSE DE L'IMPACT DE MISES À JOUR SUR DES VUES ET SUR DES CONTRAINTES D'INTÉGRITÉ XML.. Interface homme-machine [cs.HC]. Université Panthéon-Sorbonne - Paris I, 2010. Français. NNT: . tel-00580926

HAL Id: tel-00580926

<https://theses.hal.science/tel-00580926>

Submitted on 29 Mar 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Centre de Recherche en Informatique

Contribution à l'analyse de l'impact de Mises à jour sur des Vues et sur des Contraintes d'Intégrité XML

Hicham Idabal

Encadré par **Pr. Françoise Gire**

25 novembre 2010



Sommaire

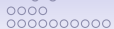
Introduction

Langages de sélection de nœuds

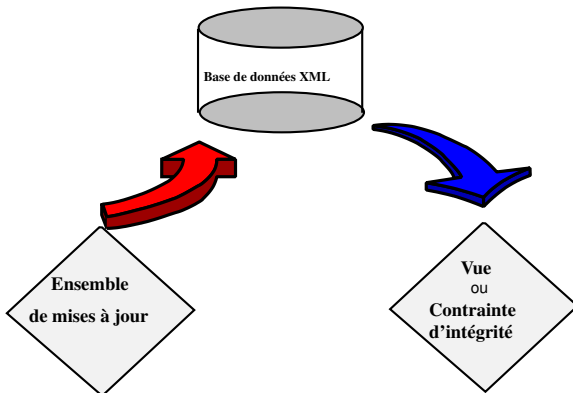
Analyse de dépendances entre Vues et MAJ

Analyse de dépendances entre CI et MAJ

Conclusion



Introduction : contexte général





Introduction : positionnement

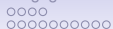
L'impact des mises à jours : un problème classique largement étudié.

Principales différences par rapport aux travaux antérieurs :

1. La plupart des travaux utilisent des informations :
 - provenant des sources
 - provenant de couches intermédiaires

⇒ On n'utilise que la définition des requêtes (Vue, MAJ, CI) et un schéma s'il existe.
2. Les langages utilisés sont différents :
 - La plupart des travaux utilisent des langages proches des standards du W3C.

⇒ On fait le choix d'un langage formel puissant à base de motifs réguliers arborescents : \mathcal{RAR}_S



Introduction : positionnement

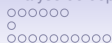
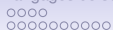
L'impact des mises à jours : un problème classique largement étudié.

Principales différences par rapport aux travaux antérieurs :

1. La plupart des travaux utilisent des informations :
 - provenant des sources
 - provenant de couches intermédiaires

⇒ On n'utilise que la définition des requêtes (Vue, MAJ, CI) et un schéma s'il existe.
2. Les langages utilisés sont différents :
 - La plupart des travaux utilisent des langages proches des standards du W3C.

⇒ On fait le choix d'un langage formel puissant à base de motifs réguliers arborescents : \mathcal{RAR}_S



Introduction : positionnement

L'impact des mises à jours : un problème classique largement étudié.

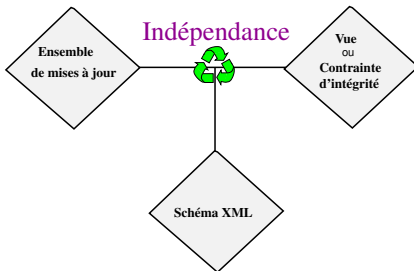
Principales différences par rapport aux travaux antérieurs :

1. La plupart des travaux utilisent des informations :
 - provenant des sources
 - provenant de couches intermédiaires

⇒ On n'utilise que la définition des requêtes (Vue, MAJ, CI) et un schéma s'il existe.
2. Les langages utilisés sont différents :
 - La plupart des travaux utilisent des langages proches des standards du W3C.

⇒ On fait le choix d'un langage formel puissant à base de motifs réguliers arborescents : \mathcal{RAR}_S

Introduction : problématique



But : détection de l'indépendance

Eviter :

- une réévaluation de la Vue, ou
- une revalidation de la Contrainte d'intégrité



Sommaire

Introduction

Langages de sélection de nœuds

Requête Arbre Régulière (\mathcal{RAR})

\mathcal{RAR} et XPath

Analyse de dépendances entre Vues et MAJ

Vue et Classe de Mises à jour

Problème PSPACE-difficile

Critère d'indépendance

Analyse de dépendances entre CI et MAJ

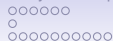
Exemple

Travaux reliés

\mathcal{RAR} : modèle uniforme pour les CI et les MAJ

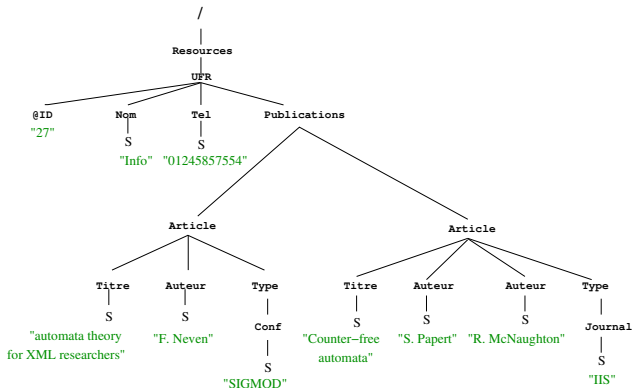
Résultat sur l'analyse de dépendance

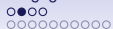
Conclusion



Un document XML

Une représentation en **arbres ordonnés étiquetés** des documents XML : $\mathcal{D}=(d, \lambda, val)$



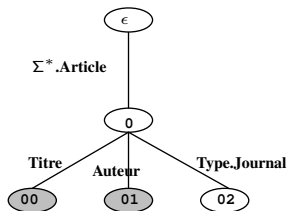


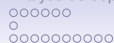
Requêtes Arbres Régulières (\mathcal{RAR}_S)

⇒ **Modéliser le processus de sélection de nœuds par un arbre**

Exemple (Vue)

- “Donner les (Titre,Auteur) des articles publiés dans un journal”





RAR : définition formelle

Définition : Une requête $RAR \mathcal{R}$ est définie par $\mathcal{R} = (\mathcal{T}, \vec{s})$ où $\mathcal{T} = (\Sigma, N, M, \mathcal{E})$ est appelé arbre template de \mathcal{R}

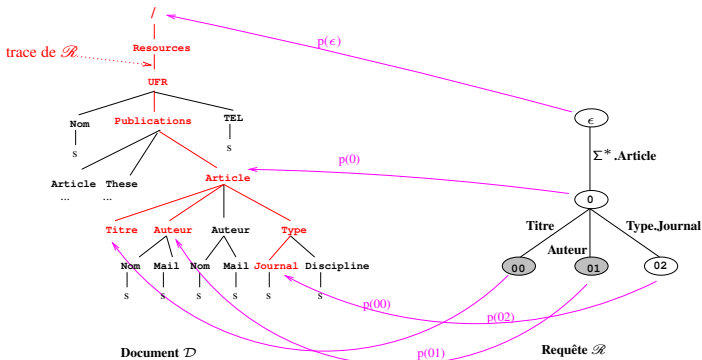
- $N \subseteq \mathbb{N}^*$: le domaine de l'arbre.
- $M \subseteq N \times N$: ensemble de ses arcs.
- $\mathcal{E} : M \rightarrow REG(\Sigma)$: application qui associe à chaque arc (w, w') de M une **expression** notée $\mathcal{E}_{(w, w')}$ qui est, soit vide, soit **régulière** propre.
- $\vec{s} = (w_1, \dots, w_n)$: n-uplet de nœuds de N à sélectionner



Évaluation d'une \mathcal{RAR}

Soit $\mathcal{R} = (\mathcal{T}, \vec{s})$ une \mathcal{RAR} où $\mathcal{T} = (\Sigma, N, M, \mathcal{E})$ une requête arbre et \mathcal{D} un document XML

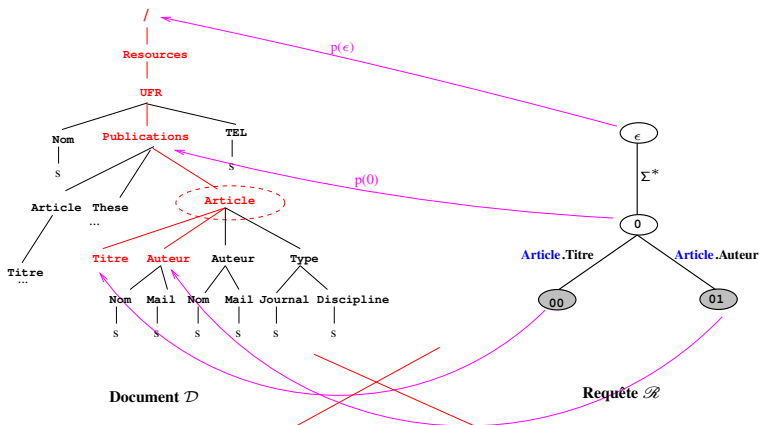
Définition : Un plongement de \mathcal{R} dans \mathcal{D} est une application $p : N \rightarrow \mathcal{N}(\mathcal{D})$



Évaluation d'une \mathcal{RAR}

Soit $\mathcal{R} = (\mathcal{I}, \vec{s})$ une \mathcal{RAR} où $\mathcal{I} = (\Sigma, N, M, \mathcal{E})$ une requête arbre et \mathcal{D} un document XML

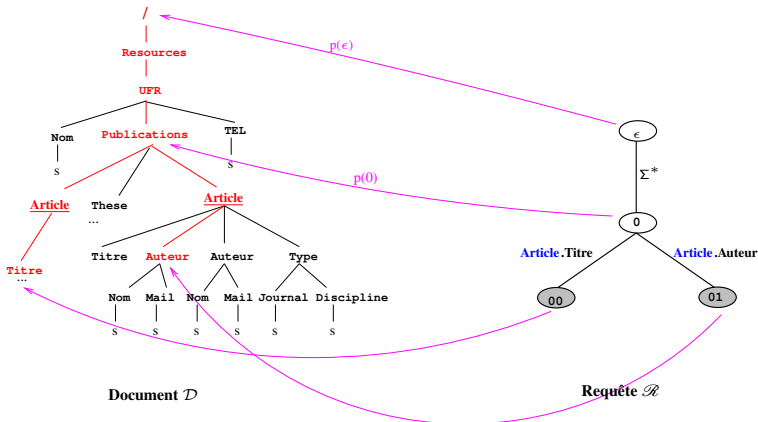
Définition : Un plongement de \mathcal{R} dans \mathcal{D} est une application $p : N \rightarrow \mathcal{N}(\mathcal{D})$



Évaluation d'une \mathcal{RAR}

Soit $\mathcal{R} = (\mathcal{I}, \vec{s})$ une \mathcal{RAR} où $\mathcal{I} = (\Sigma, N, M, \mathcal{E})$ une requête arbre et \mathcal{D} un document XML

Définition : Un plongement de \mathcal{R} dans \mathcal{D} est une application $p : N \rightarrow \mathcal{N}(\mathcal{D})$

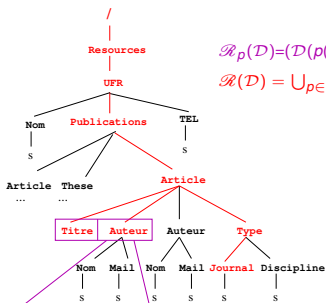




Évaluation d'une \mathcal{RAR}

Soit $\mathcal{R} = (\mathcal{T}, \vec{s})$ une \mathcal{RAR} où $\mathcal{T} = (\Sigma, N, M, \mathcal{E})$ une requête arbre et \mathcal{D} un document XML

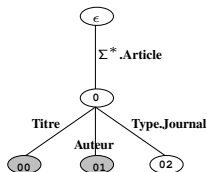
Définition : Un plongement de \mathcal{R} dans \mathcal{D} est une application $p : N \rightarrow \mathcal{N}(\mathcal{D})$



Document \mathcal{D}

$$\mathcal{R}_p(\mathcal{D}) = (\mathcal{D}(p(w_1)), \dots, \mathcal{D}(p(w_n)))$$

$$\mathcal{R}(\mathcal{D}) = \bigcup_{p \in \mathcal{P}} \mathcal{R}_p(\mathcal{D})$$



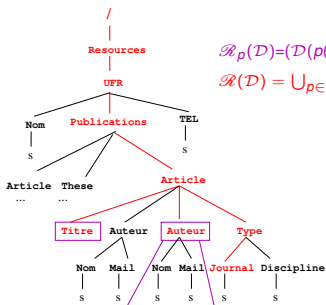
Requête \mathcal{R}



Évaluation d'une \mathcal{RAR}

Soit $\mathcal{R} = (\mathcal{T}, \vec{s})$ une \mathcal{RAR} où $\mathcal{T} = (\Sigma, N, M, \mathcal{E})$ une requête arbre et \mathcal{D} un document XML

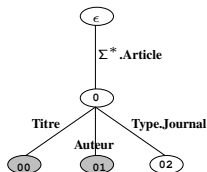
Définition : Un plongement de \mathcal{R} dans \mathcal{D} est une application $p : N \rightarrow \mathcal{N}(\mathcal{D})$



Document \mathcal{D}

$$\mathcal{R}_p(\mathcal{D}) = (\mathcal{D}(p(w_1)), \dots, \mathcal{D}(p(w_n)))$$

$$\mathcal{R}(\mathcal{D}) = \bigcup_{p \in \mathcal{P}} \mathcal{R}_p(\mathcal{D})$$



Requête \mathcal{R}



XPath

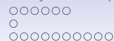
XPath :

Langage de requête, standard du W3C (1998).

⇒ sélectionne un ensemble d'éléments d'un document XML répondant à certaines contraintes structurelles ou textuelles.

Path = succession d'étapes (step).

Step = **axe::type**[**filtre**]



RAR et XPath

Modèle *RAR* :

- Modèle formel puissant.
- Exprime certaines requêtes que le langage standard *XPath* ne permet pas d'exprimer.



RAR et XPath

Modèle *RAR* :

- Modèle formel puissant.
- Exprime certaines requêtes que le langage standard *XPath* ne permet pas d'exprimer.

⇒ *XPath* et le modèle des requêtes *RAR_s* sont incomparables.



\mathcal{RAR} et XPath

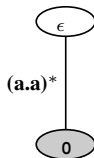
Modèle \mathcal{RAR} :

- Modèle formel puissant.
- Exprime certaines requêtes que le langage standard *XPath* ne permet pas d'exprimer.

⇒ *XPath* et le modèle des requêtes \mathcal{RAR}_s sont incomparables.

Contre-exemples :

1. "Donner tous les nœuds dont chaque ancêtre est étiqueté par un 'a' et le nombre des ancêtres est pair" : non exprimable par XPath





\mathcal{RAR} et XPath

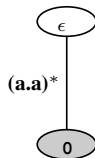
Modèle \mathcal{RAR} :

- Modèle formel puissant.
- Exprime certaines requêtes que le langage standard *XPath* ne permet pas d'exprimer.

⇒ *XPath* et le modèle des requêtes \mathcal{RAR}_s sont incomparables.

Contre-exemples :

1. "Donner tous les nœuds dont chaque ancêtre est étiqueté par un 'a' et le nombre des ancêtres est pair" : non exprimable par XPath
2. `//a[not(./b)]` : non exprimable par \mathcal{RAR}





CoreXPath positif

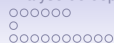
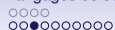
CoreXPath⁺ : Fragment navigationnel de *XPath* (sans négation, sans disjonction).

Formellement, un chemin absolu P_a de *CoreXPath⁺* est défini par la grammaire :

$$P_a \equiv /P$$

$$P \equiv axis :: \lambda \parallel P/axis :: \lambda \parallel P[F] \parallel P|P$$

$$F \equiv axis :: \lambda \parallel axis :: \lambda[F] \parallel /F$$



CoreXPath positif

CoreXPath⁺ : Fragment navigationnel de *XPath* (sans négation, sans disjonction).

Formellement, un chemin absolu P_a de *CoreXPath⁺* est défini par la grammaire :

$$P_a \equiv /P$$

$$P \equiv axis :: \lambda \parallel P/axis :: \lambda \parallel P[F] \parallel P|P$$

$$F \equiv axis :: \lambda \parallel axis :: \lambda[F] \parallel /F$$

axis : self, child, parent, descendant,
 descendant_or_self, ancestor, ancestor_or_self,
 following_sibling, following, preceding_sibling,
 preceding



CoreXPath positif

CoreXPath⁺ : Fragment navigationnel de *XPath* (sans négation, sans disjonction).

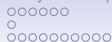
Formellement, un chemin absolu P_a de *CoreXPath*⁺ est défini par la grammaire :

$$P_a \equiv /P$$

$$P \equiv axis :: \lambda \parallel P/axis :: \lambda \parallel P[F] \parallel P|P$$

$$F \equiv axis :: \lambda \parallel axis :: \lambda[F] \parallel /F$$

► *CoreXPath* a la même expressivité que FO^2_{tree}



RAR et CoreXPath positif

Proposition 1

Tout chemin P_a de *CoreXPath*⁺ est exprimable par une union finie de requêtes *RAR* monadiques.



XPath conditionnel : CXPath

CXPath : (M. Marx, 2005)

prim_axis ::= *child* || *parent* || *right* || *left*.

P ::= *prim_axis* || *P/P* || *P ∪ P* || (*prim_axis/?F*)* || ?*F*

F ::= λ || ⊤ || < *P* > || *not F* || *F and F* || *F or F*



XPath conditionnel : CXPath

CXPath : (M. Marx, 2005)

prim_axis ::= *child* || *parent* || *right* || *left*.

P ::= *prim_axis* || *P/P* || *P ∪ P* || (*prim_axis*/*?F*)* || *?F*

F ::= λ || \top || $\langle P \rangle$ || *not F* || *F and F* || *F or F*

► CXPath est FO_{tree} -complet



XPath conditionnel : CXPath

CXPath : (M. Marx, 2005)

$prim_axis ::= child \parallel parent \parallel right \parallel left.$

$P ::= prim_axis \parallel P/P \parallel P \cup P \parallel (prim_axis/?F)^* \parallel ?F$

$F ::= \lambda \parallel \top \parallel \langle P \rangle \parallel not\ F \parallel F\ and\ F \parallel F\ or\ F$

► **CXPath est FO_{tree} -complet**

Exemple :

Soit la requête qui sélectionne les nœuds origines d'un chemin dont tous les nœuds sont étiquetés par b sauf le dernier qui est étiqueté par a.

En CXPath : $? \langle (child/?b)^* /child/?a \rangle$ (ou $//*[./b^*/a]$)



XPath conditionnel : CXPath

CXPath : (M. Marx, 2005)

$prim_axis ::= child \parallel parent \parallel right \parallel left.$

$P ::= prim_axis \parallel P/P \parallel P \cup P \parallel (prim_axis/?F)^* \parallel ?F$

$F ::= \lambda \parallel \top \parallel \langle P \rangle \parallel not\ F \parallel F\ and\ F \parallel F\ or\ F$

► **CXPath est FO_{tree} -complet**

Exemple :

Soit la requête qui sélectionne les nœuds origines d'un chemin dont tous les nœuds sont étiquetés par b sauf le dernier qui est étiqueté par a.

En CXPath : $? \langle (child/?b)^* /child/?a \rangle$ (ou $//*[./b^*/a]$)

En FO_{tree} : $\exists y(xR_{\downarrow}y \wedge O_a(y) \wedge \forall z((xR_{\downarrow}z \wedge zR_{\downarrow}y) \rightarrow O_b(z)))$.



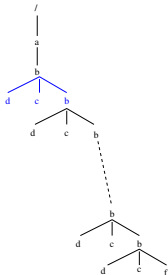
\mathcal{RAR} versus CXPath

Proposition 2

1. $CXPath^+ \not\subseteq \cup \mathcal{RAR}$
2. $\cup \mathcal{RAR} \not\subseteq CXPath^+$

Preuve par des contre-exemples :

1- $/a/(b[d \text{ and } c])^+/f$: non exprimable par $\cup \mathcal{RAR}$





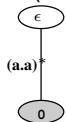
\mathcal{RAR} versus CXPath

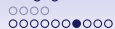
Proposition 2

1. $CXPath^+ \not\subseteq \cup \mathcal{RAR}$
2. $\cup \mathcal{RAR} \not\subseteq CXPath^+$

Preuve par des contre-exemples :

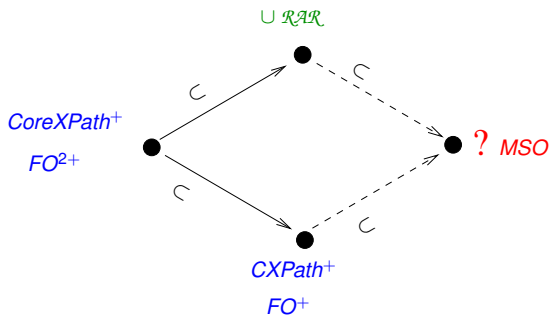
2-(a.a)* n'est pas sans étoile \Rightarrow non définissable par FO_{tree}

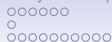
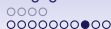




Récapitulatif de Comparaison

\mathcal{RAR} versus $CoreXPath$ et ses extensions





XPath régulier : RegularXPath

RegularXPath : (M. Marx, 2005)

$prim_axis ::= child \parallel parent \parallel right \parallel left.$

$P ::= prim_axis \parallel P/P \parallel P \cup P \parallel P^* \parallel ?F$

$F ::= \lambda \parallel < P > \parallel not F \parallel F and F \parallel F or F$

- La puissance expressive de ce langage est strictement incluse entre celle des logiques FO_{tree} et MSO

Proposition 3

Toute requête \mathcal{RAR} monadique \mathcal{R} est exprimable par une expression de chemin de **RegularXPath**.



\mathcal{RAR} versus Logique

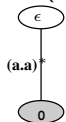
FOREG (F. Neven et T. Schwentick, 2000) est l'extension de *FO* par des expressions régulières de chemin verticales et horizontales.

Proposition 4

Toute requête $\mathcal{RAR} \mathcal{R}$ est exprimable par une propriété définie par une formule $\phi_{\mathcal{R}}$ de **FOREG** ($Eval_{\mathcal{D}}(\mathcal{R}) = \{ n \in \mathcal{D} \mid \mathcal{D} \models \phi_{\mathcal{R}}(n) \}$).

Exemple

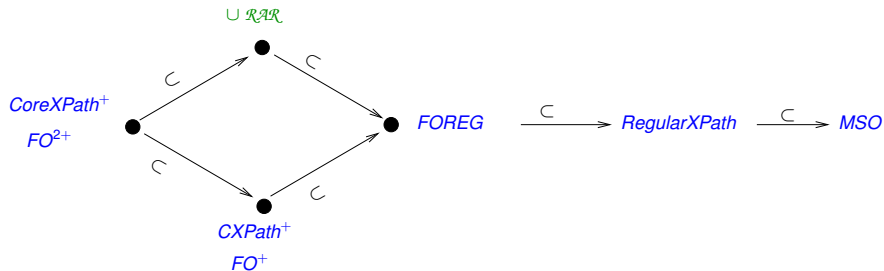
$\phi(x, y) = [(\theta_{aa}(s, t)\theta_{aa}(s, t))^* \theta_{aa}(s, t)]_{s,t}^{\downarrow}(x, y)$ avec
 $\theta_{aa}(s, t) = O_a(s) \wedge O_a(t)$





Récapitulatif de Comparaison

\mathcal{RAR} versus CoreXPath et ses extensions





Sommaire

Introduction

Langages de sélection de nœuds

Requête Arbre Régulière (*RAR*)

RAR et XPath

Analyse de dépendances entre Vues et MAJ

Vue et Classe de Mises à jour

Problème PSPACE-difficile

Critère d'indépendance

Analyse de dépendances entre CI et MAJ

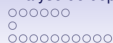
Exemple

Travaux reliés

RAR : modèle uniforme pour les CI et les MAJ

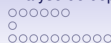
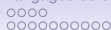
Résultat sur l'analyse de dépendance

Conclusion



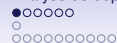
État de l'art

- **Travaux utilisant les sources :**
 - [S. Abiteboul & al, 1998] "*Incremental Maintenance for Materialized Views over Semistructured Data*"
 - [M. Onizuka & al, 2003] "*Incremental Maintenance for Materialized XPath/XSLT Views*"
 - [A. Sawires & al, 2005-2006] "*Incremental maintenance of path-expression views*"
 - [V. Sans 2007] "*Maintenance de vues XML matérialisées à partir de sources web*"
- **Travaux n'utilisant pas les sources :**
 - [M. Raghavachari and O. Shmueli, 2006] "*Conflicting XML Updates*"
 - [G. Ghelli & al, 2007] "*Commutativity Analysis in XML Update Languages*"
 - [M. Benedikt and J. Cheney, 2009] "*Schema-Based Independence Analysis for XML Updates*"
 - [N. Bidoit & al, 2010] "*Detecting XML Query-Update Independence*"



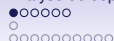
État de l'art

- **Travaux utilisant les sources :**
 - [S. Abiteboul & al, 1998] "*Incremental Maintenance for Materialized Views over Semistructured Data*"
 - [M. Onizuka & al, 2003] "*Incremental Maintenance for Materialized XPath/XSLT Views*"
 - [A. Sawires & al, 2005-2006] "*Incremental maintenance of path-expression views*"
 - [V. Sans 2007] "*Maintenance de vues XML matérialisées à partir de sources web*"
- **Travaux n'utilisant pas les sources :**
 - [M. Raghavachari and O. Shmueli, 2006] "*Conflicting XML Updates*"
 - [G. Ghelli & al, 2007] "*Commutativity Analysis in XML Update Languages*"
 - [M. Benedikt and J. Cheney, 2009] "*Schema-Based Independence Analysis for XML Updates*"
 - [N. Bidoit & al, 2010] "*Detecting XML Query-Update Independence*"



Vue, Mise à jour

- Requête de Vue n-aire : Extraction de tuples de nœuds (E) + personnalisation (P)
 $\Rightarrow \mathcal{V} = P \circ E$, dans ce travail \mathcal{V} sera identifiée à E
- Mise à jour : sélection de nœuds (C) + Remplacement (R)
 $\Rightarrow q = R \circ C$



Vue, Mise à jour

- Requête de Vue n-aire : Extraction de tuples de nœuds (E) + personnalisation (P)
 $\Rightarrow \mathcal{V} = P \circ E$, dans ce travail \mathcal{V} sera identifiée à E
- Mise à jour : sélection de nœuds (\mathcal{C}) + Remplacement (R)
 $\Rightarrow q = R \circ \mathcal{C}$

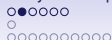
Remarque

\mathcal{C} et E : processus commun de sélection de tuples de nœuds



Classe de Mises à jour

- Une mise à jour $q = R \circ C$ où C sélectionne les nœuds à modifier.
- C représente une classe de mises à jour : toutes les mises à jour qui modifient les nœuds sélectionnés par C .



Classe de Mises à jour

- Une mise à jour $q = R \circ C$ où C sélectionne les nœuds à modifier.
- C représente une classe de mises à jour : toutes les mises à jour qui modifient les nœuds sélectionnés par C .

Exemple

1. “**Modifier les auteurs d’articles** en y ajoutant un N° de téléphone (Tel)”
2. “**Modifier les auteurs d’articles** en changeant Nom en (NomFamille, Prénom)”.



Indépendance : définition

Définition 1 : impact

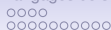
Une mise à jour q **impacte** une vue \mathcal{V} sur \mathcal{D} ssi
 $\mathcal{V}(\mathcal{D}) \neq \mathcal{V}(q(\mathcal{D}))$

Définition 2 : indépendance

\mathcal{V} est indépendante par rapport à la classe \mathcal{C} ssi
 $\forall \mathcal{D}, \forall q \in \mathcal{C}, q$ n'impacte pas \mathcal{V} sur \mathcal{D}

La problématique est alors :

Étant données une vue \mathcal{V} et une classe de mises à jour \mathcal{C} ,
 \mathcal{V} est-elle indépendante par rapport à \mathcal{C} ?



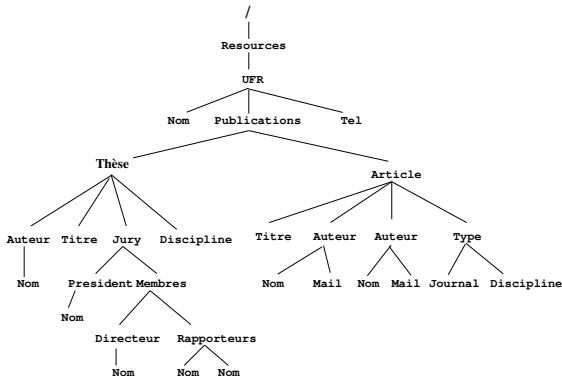
Exemple d'indépendance

Soit la vue \mathcal{V} :

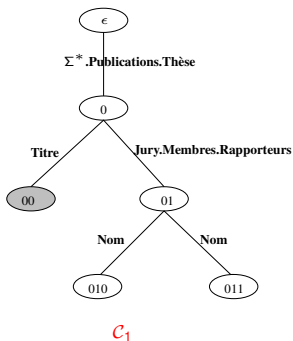
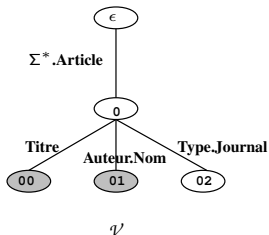
“Donner les (Titre, Nom d’auteur) des articles publiés dans un journal”

Et la classe de mises à jour : \mathcal{C}_1

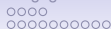
“Mettre à jour les titres des thèses qui ont au moins deux rapporteurs”



Exemple d'indépendance



Il n'existe pas de mise à jour de \mathcal{C}_1 qui impacte \mathcal{V}
 \Rightarrow **Indépendance** de \mathcal{V} par rapport à \mathcal{C}_1



Exemple de dépendance

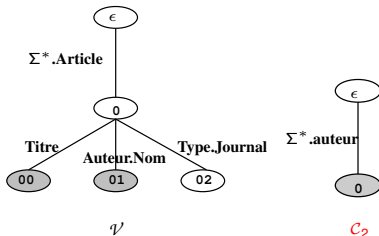
Avec la même requête \mathcal{V}

Mise à jour : $q_1 \in \mathcal{C}_2$

“**Modifier les auteurs d’articles** en y ajoutant un N° de téléphone (Tel)”

Mise à jour : $q_2 \in \mathcal{C}_2$

“**Modifier les auteurs d’articles** en changeant Nom en (NomFamille, Prénom)”





Exemple de dépendance

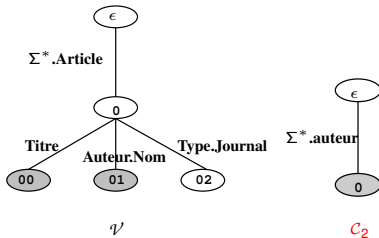
Avec la même requête \mathcal{V}

Mise à jour : $q_1 \in \mathcal{C}_2$

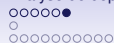
“**Modifier les auteurs d’articles** en y ajoutant un N° de téléphone (Tel)”

Mise à jour : $q_2 \in \mathcal{C}_2$

“**Modifier les auteurs d’articles** en changeant Nom en (NomFamille, Prénom)”



q_1 : pas d’impact mais q_2 : impact . \Rightarrow dépendance de \mathcal{V} par rapport à \mathcal{C}_2



Indépendance par rapport à un schéma

Notons $\text{valid}(Sc)$ l'ensemble des documents valides par rapport à Sc

Indépendance par rapport à un schéma

Une vue \mathcal{V} est indépendante par rapport à une classe de mises à jour \mathcal{C} dans le contexte Sc ssi $\forall \mathcal{D} \in \text{valid}(Sc), \forall q \in \mathcal{C}$ avec $q(\mathcal{D}) \in \text{valid}(Sc)$, q n'impacte pas \mathcal{V} sur \mathcal{D}

La problématique devient :

Etant données une vue \mathcal{V} , une classe de mises à jour \mathcal{C} et un schéma Sc , \mathcal{V} est-elle indépendante par rapport à \mathcal{C} dans le contexte Sc ?



Résultats sur la dépendance

Proposition 1

Décider si $\mathcal{V} = (\mathcal{T}_{\mathcal{V}}, \vec{s}_{\mathcal{V}})$ est indépendante par rapport à $\mathcal{C} = (\mathcal{T}_{\mathcal{C}}, \vec{s}_{\mathcal{C}})$ est un problème **PSPACE-difficile**.

Preuve réduire le problème d'inclusion des expressions régulières qui est PSPACE-difficile au problème de la dépendance.



Analyse d'indépendance

Dans la suite nous exhibons une condition suffisante testable en temps polynomial d'indépendance.



Analyse d'indépendance

Dans la suite nous exhibons une condition suffisante testable en temps polynomial d'indépendance.

Une vue \mathcal{V} est dépendante d'une classe de mises à jour \mathcal{C} dans le contexte \mathcal{Sc} ssi

$\exists \mathcal{D} \in \mathbf{valid}(\mathcal{Sc}), \exists q \in \mathcal{C}$ tel que $q(\mathcal{D}) \in \mathbf{valid}(\mathcal{Sc})$ et $\mathcal{V}(q(\mathcal{D})) \neq \mathcal{V}(\mathcal{D})$



Analyse d'indépendance

Dans la suite nous exhibons une condition suffisante testable en temps polynomial d'indépendance.

Une vue \mathcal{V} est dépendante d'une classe de mises à jour \mathcal{C} dans le contexte \mathcal{Sc} ssi

$\exists \mathcal{D} \in \mathbf{valid}(\mathcal{Sc}), \exists q \in \mathcal{C}$ tel que $q(\mathcal{D}) \in \mathbf{valid}(\mathcal{Sc})$ et $\mathcal{V}(q(\mathcal{D})) \neq \mathcal{V}(\mathcal{D})$

i.e

- $\vec{n} \in \mathcal{V}(\mathcal{D})$ et $\vec{n} \notin \mathcal{V}(q(\mathcal{D}))$



Analyse d'indépendance

Dans la suite nous exhibons une condition suffisante testable en temps polynomial d'indépendance.

Une vue \mathcal{V} est dépendante d'une classe de mises à jour \mathcal{C} dans le contexte \mathcal{Sc} ssi

$\exists \mathcal{D} \in \mathbf{valid}(\mathcal{Sc}), \exists q \in \mathcal{C}$ tel que $q(\mathcal{D}) \in \mathbf{valid}(\mathcal{Sc})$ et $\mathcal{V}(q(\mathcal{D})) \neq \mathcal{V}(\mathcal{D})$

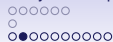
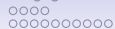
i.e

1. $\vec{n} \in \mathcal{V}(\mathcal{D})$ et $\vec{n} \notin \mathcal{V}(q(\mathcal{D}))$
2. ou $\vec{n} \notin \mathcal{V}(\mathcal{D})$ et $\vec{n} \in \mathcal{V}(q(\mathcal{D}))$



Analyse d'indépendance

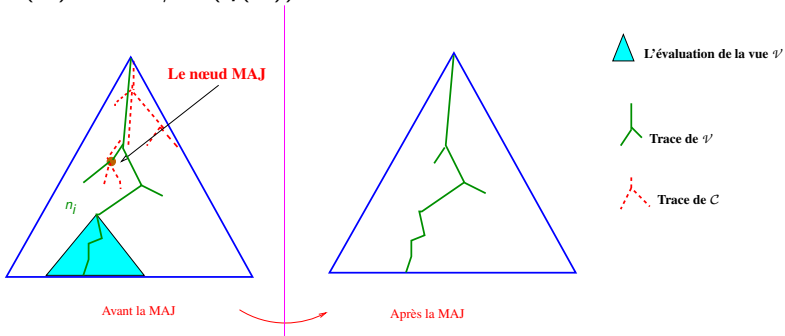
1- $\vec{n} \in \mathcal{V}(\mathcal{D})$ et $\vec{n} \notin \mathcal{V}(q(\mathcal{D})) \Rightarrow$

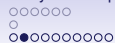
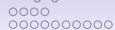


Analyse d'indépendance

1- $\vec{n} \in \mathcal{V}(\mathcal{D})$ et $\vec{n} \notin \mathcal{V}(q(\mathcal{D})) \Rightarrow$

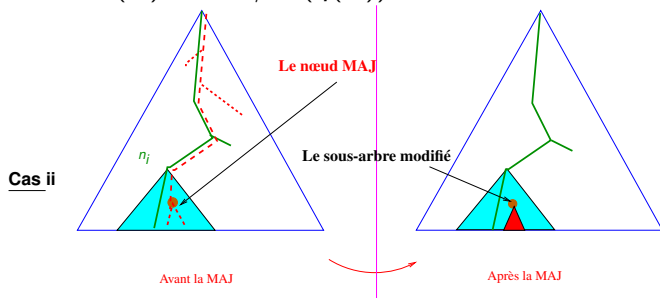
Cas i

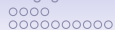




Analyse d'indépendance

1- $\vec{n} \in \mathcal{V}(\mathcal{D})$ et $\vec{n} \notin \mathcal{V}(q(\mathcal{D})) \Rightarrow$

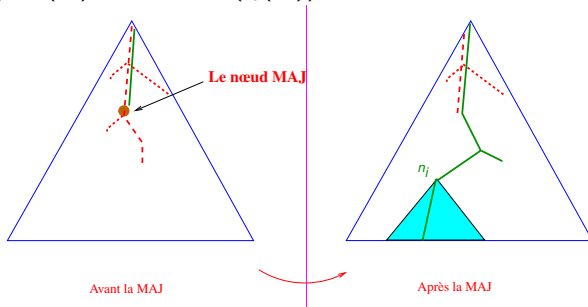




Analyse d'indépendance

2- $\vec{n} \notin \mathcal{V}(\mathcal{D})$ et $\vec{n} \in \mathcal{V}(q(\mathcal{D})) \Rightarrow$

Cas iii





Analyse d'indépendance

plus formellement, nous écrivons :

- (1) $\implies \hat{p}_c(s_c) \in \mathcal{N}(\text{trace}_{p_v}(\mathcal{V}, \mathcal{D})) \cup \mathcal{N}(\mathcal{V}_{p_v}(\mathcal{D}))$
- (2) $\implies \hat{p}_c(s_c) \in \mathcal{N}(\text{trace}_{p_v}(\mathcal{V}, q(\mathcal{D})))$

p_v : un plongement de \mathcal{V}

\hat{p}_c : un plongement partiel de \mathcal{C} (plongement de $\hat{\mathcal{R}} = (\hat{\mathcal{T}}, \vec{s})$ dans \mathcal{D}).

et $\mathcal{N}(\vec{n}) = \cup_{i=1}^n \mathcal{N}(\mathcal{D}_i)$ est l'ensemble des nœuds de \mathcal{D}_i avec $\vec{n} = (\mathcal{D}_1, \dots, \mathcal{D}_n)$ un n-uplet d'arbres.



Un langage régulier pour tester la dépendance

Définition

Soit \mathcal{L} l'ensemble des arbres \mathcal{D} vérifiant :

(i) $\mathcal{D} \in \text{valide}(\mathcal{S}_C)$

(ii) il existe **une trace** $\text{trace}_{p_V}(\mathcal{V}, \mathcal{D})$ de \mathcal{V} dans \mathcal{D} selon un plongement p_V et

il existe une **trace partielle** $\text{trace}_{\hat{p}_C}(\hat{\mathcal{C}}, \mathcal{D})$ de \mathcal{C} dans \mathcal{D} selon un plongement partiel \hat{p}_C et un nœud s_C de \vec{s}_C tel que

$\hat{p}_C(s_C) \in \mathcal{N}(\text{trace}_{p_V}(\mathcal{V}, \mathcal{D})) \cup \mathcal{N}(\mathcal{V}_{p_V}(\mathcal{D}))$.



Automate d'arbre et indépendance

Soit \mathcal{V} une requête de vue, et \mathcal{C} une classe de mises à jour

Proposition 2

Si $\mathcal{L} = \emptyset$ alors \mathcal{V} est indépendante par rapport \mathcal{C} dans le contexte \mathcal{S}_c

$\implies \mathcal{L} = \emptyset$ est une condition suffisante pour l'indépendance de \mathcal{V} par rapport \mathcal{C}



Cas particulier : problème d'indépendance polynomial

Dans le cas où

1. les nœuds de sélection de la requête de vue $\mathcal{V} = (\mathcal{I}_V, \vec{S}_V)$ vérifient : “tout chemin de la racine à une feuille de \mathcal{I}_V contient au moins un nœud de \vec{S}_V ”.
2. et les nœuds de mises à jour sont tous des feuilles.



Cas particulier : problème d'indépendance polynomial

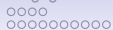
Dans le cas où

1. les nœuds de sélection de la requête de vue $\mathcal{V} = (\mathcal{T}_V, \vec{s}_V)$ vérifient : “tout chemin de la racine à une feuille de \mathcal{T}_V contient au moins un nœud de \vec{s}_V ”.
2. et les nœuds de mises à jour sont tous des feuilles.

On a :

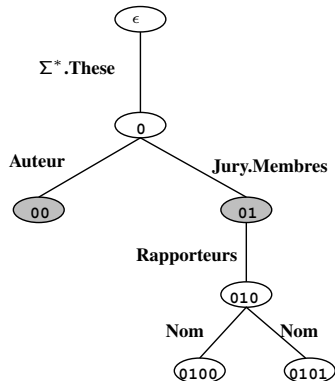
Proposition 3

\mathcal{V} est indépendante de la classe \mathcal{C} si et seulement si \mathcal{L} est vide.

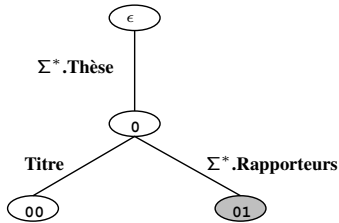


Cas particulier : problème d'indépendance polynomial

Exemple : “Donner les couples (Auteur, Membres) des thèses ayant au moins deux rapporteurs”



\mathcal{V}



\mathcal{C}

○○○
○○○○○○○○○○

○○○○○
○
○○○○○○○●○

○○
○
○○○
○○

Construction de \mathcal{A}

Pour vérifier la vacuité de \mathcal{L} , nous définissons l'automate d'arbre \mathcal{A} qui reconnaît tous les arbres de \mathcal{L} .

○○○
○○○○○○○○○○

○○○○○
○
○○○○○○○●○

○○
○
○○○
○○

Construction de \mathcal{A}

Pour vérifier la vacuité de \mathcal{L} , nous définissons l'automate d'arbre \mathcal{A} qui reconnaît tous les arbres de \mathcal{L} .

Étapes de construction de \mathcal{A} :

1. Construire l'automate $\mathcal{A}_{\hat{c}}$ qui reconnaît l'ensemble des arbres contenant au moins une trace de \hat{c}

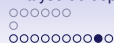


Construction de \mathcal{A}

Pour vérifier la vacuité de \mathcal{L} , nous définissons l'automate d'arbre \mathcal{A} qui reconnaît tous les arbres de \mathcal{L} .

Étapes de construction de \mathcal{A} :

1. Construire l'automate $\mathcal{A}_{\hat{c}}$ qui reconnaît l'ensemble des arbres contenant au moins une trace de \hat{c}
2. Construire l'automate $\mathcal{A}_{\mathcal{V}}$ qui reconnaît l'ensemble des arbres contenant au moins une trace de \mathcal{V} + le modifier en $\bar{\mathcal{A}}_{\mathcal{V}}$ en changeant qlq transitions pour identifier les nœuds descendants des nœuds de sélection.



Construction de \mathcal{A}

Pour vérifier la vacuité de \mathcal{L} , nous définissons l'automate d'arbre \mathcal{A} qui reconnaît tous les arbres de \mathcal{L} .

Étapes de construction de \mathcal{A} :

1. Construire l'automate $\mathcal{A}_{\hat{c}}$ qui reconnaît l'ensemble des arbres contenant au moins une trace de \hat{c}
2. Construire l'automate \mathcal{A}_v qui reconnaît l'ensemble des arbres contenant au moins une trace de v + le modifier en $\bar{\mathcal{A}}_v$ en changeant qlq transitions pour identifier les nœuds descendants des nœuds de sélection.
3. Définir $\mathcal{B} = \mathcal{A}_{\hat{c}} \times \bar{\mathcal{A}}_v$ + le changer pour vérifier la condition $\exists p_v, \hat{p}_c$ tel que $\hat{p}_c(s_c) \in \mathcal{N}(\text{trace}_{p_v}(v, \mathcal{D})) \cup \mathcal{N}(v_{p_v}(\mathcal{D}))$



Construction de \mathcal{A}

Pour vérifier la vacuité de \mathcal{L} , nous définissons l'automate d'arbre \mathcal{A} qui reconnaît tous les arbres de \mathcal{L} .

Étapes de construction de \mathcal{A} :

1. Construire l'automate $\mathcal{A}_{\hat{c}}$ qui reconnaît l'ensemble des arbres contenant au moins une trace de \hat{c}
2. Construire l'automate \mathcal{A}_v qui reconnaît l'ensemble des arbres contenant au moins une trace de v + le modifier en $\bar{\mathcal{A}}_v$ en changeant qlq transitions pour identifier les nœuds descendants des nœuds de sélection.
3. Définir $\mathcal{B} = \mathcal{A}_{\hat{c}} \times \bar{\mathcal{A}}_v$ + le changer pour vérifier la condition $\exists p_v, \hat{p}_c$ tel que $\hat{p}_c(s_c) \in \mathcal{N}(\text{trace}_{p_v}(v, \mathcal{D})) \cup \mathcal{N}(v_{p_v}(\mathcal{D}))$
4. Définir $\mathcal{A} = \mathcal{B} \times \mathcal{A}_s$ où \mathcal{A}_s représente l'automate du schéma s_c



Test polynomial du critère d'indépendance

Proposition 4 : Taille de l'automate \mathcal{A}

Si a_c et a_v désignent les arités maximales de \mathcal{C} et \mathcal{V} alors il existe une constante K tel que $|\mathcal{A}| \leq K|\Sigma|^3|\mathcal{A}_{Sc}||\mathcal{C}||\mathcal{V}|a_c a_v$.

Proposition 5 : Test de vacuité

Le test de vacuité de $L(\mathcal{A})$ est polynomial (en $O(|\Sigma|^6|\mathcal{A}_{Sc}|^2|\mathcal{C}|^2|\mathcal{V}|^2 a_c^2 a_v^2)$).



Sommaire

Introduction

Langages de sélection de nœuds

Requête Arbre Régulière (*RAR*)

RAR et XPath

Analyse de dépendances entre Vues et MAJ

Vue et Classe de Mises à jour

Problème PSPACE-difficile

Critère d'indépendance

Analyse de dépendances entre CI et MAJ

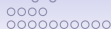
Exemple

Travaux reliés

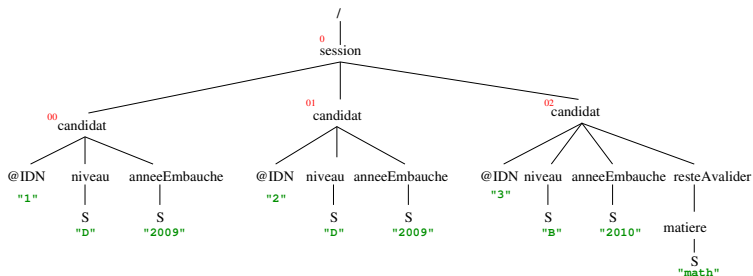
RAR : modèle uniforme pour les CI et les MAJ

Résultat sur l'analyse de dépendance

Conclusion

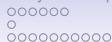
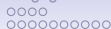


Introduction : exemple

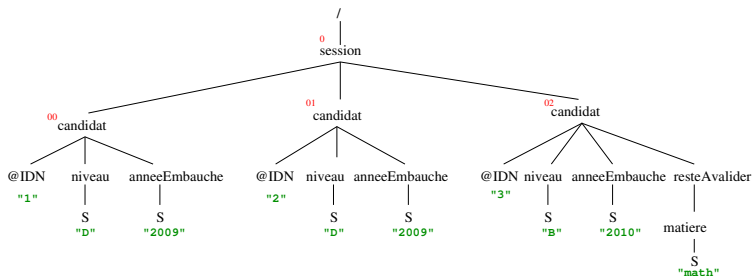


\mathcal{D} un document XML, $\mathcal{D}=(D, \lambda, val)$

- $D \subset \mathbb{N}^*$: un domaine d'arbre noté par $\mathcal{N}(D)$
- $\lambda : D \rightarrow \Sigma = E \cup A \cup \{S\}$
- $val : D \rightarrow D \cup I^*$



Introduction : exemple

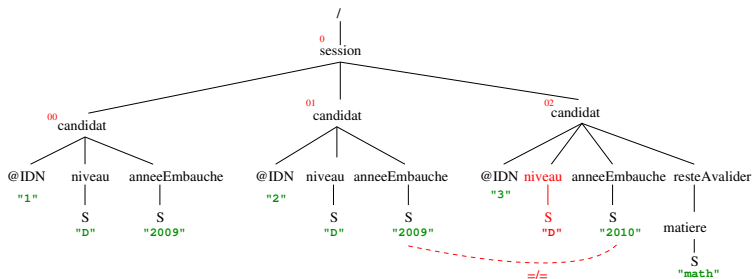


Une dépendance fonctionnelle (satisfaite avant la MAJ)

"Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job".



Introduction : exemple



Une MAJ

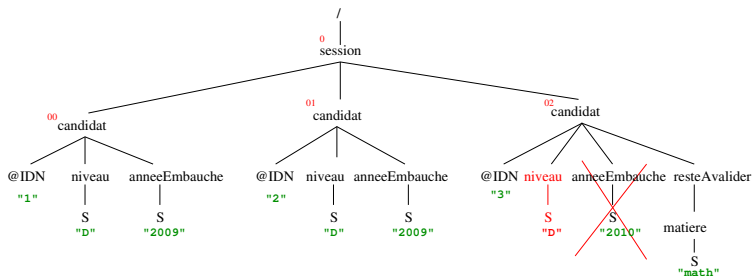
"Mettre à jour le niveau de tous les candidats auxquels il reste des épreuves à valider"

La dépendance fonctionnelle (violée après la MAJ)

"Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job".



Introduction : exemple



Schema Sc

candidat : (niveau, (anneeEmbauche | resteAvalider))

La dépendance fonctionnelle : "Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job" **reste satisfaite après la MAJ**



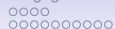
Travaux reliés

- **Travaux sur la spécification :**
 - [W. Fan & al, 2000 - 2002] "*Integrity constraints for XML*"
 - [P. Buneman & al, 2003] "*Reasoning about Keys for XML*"
 - [S. Hartmann & S. Link, 2003] "*More Functional Dependencies for XML*"
 - [M. Arenas & L. Libkin, 2004] "*A normal form for XML documents*"
- **Travaux traitant les mises à jour :**
 - [Y. Chen & al, 2002] "XKvalidator : a constraint validator for XML"
 - [M. A. Lima, 2007] "Maintenance incrémentale des contraintes d'intégrité en XML"



Travaux reliés

- **Travaux sur la spécification :**
 - [W. Fan & al, 2000 - 2002] "*Integrity constraints for XML*"
 - [P. Buneman & al, 2003] "*Reasoning about Keys for XML*"
 - [S. Hartmann & S. Link, 2003] "*More Functional Dependencies for XML*"
 - [M. Arenas & L. Libkin, 2004] "*A normal form for XML documents*"
- **Travaux traitant les mises à jour :**
 - [Y. Chen & al, 2002] "XKvalidator : a constraint validator for XML"
 - [M. A. Lima, 2007] "Maintenance incrémentale des contraintes d'intégrité en XML"



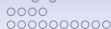
Expression des dépendances fonctionnelles

Le modèle le plus utilisé est le modèle des chemins linéaires simples.

Exemple :

"Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job".

(/session ,({candidat/niveau} → candidat/anneeEmbauche))



Expression des dépendances fonctionnelles

Le modèle le plus utilisé est le modèle des chemins linéaires simples.

Exemple :

"Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job".

(/session, ({ candidat/niveau } → candidat/anneeEmbauche))

contexte condition cible

Expression des dépendances fonctionnelles

Le modèle le plus utilisé est le modèle des chemins linéaires simples.

Exemple :

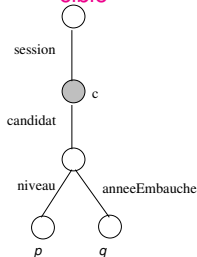
"Deux candidats qui ont le même niveau académique et qui ont trouvé du travail, ont été embauchés la même année dans leur premier job".

$(\underbrace{/session}_{\text{contexte}}, (\underbrace{\{candidat/niveau\}}_{\text{condition}} \rightarrow \underbrace{candidat/anneeEmbauche}_{\text{cible}}))$

Le modèle choisi :

REQUÊTES ARBRES RÉGULIÈRES

(\mathcal{RAR}_S)





Modélisation des dfs par les \mathcal{RAR}_S

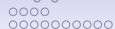
Définition

Une XDF- \mathcal{RAR} est une expression $fd = (\mathcal{FD}, c)$ où :

- $\mathcal{FD} = (\mathcal{T}, \vec{s} = \{p_1[E_1], p_2[E_2], \dots, p_n[E_n], q[E_{n+1}]\})$ est une \mathcal{RAR} .
 p_1, \dots, p_n and q sont associés à des types d'égalités $E_i \in \{V, N\}$ ($i=1, \dots, n+1$)
- c (nœud contexte) est un nœud ancêtre des p_1, p_2, \dots, p_n (nœuds conditions) et de q (nœud cible)

V est l'égalité par valeur : ($w_1 \equiv_v w_2 \Leftrightarrow \mathcal{D}(w_1)$ et $\mathcal{D}(w_2)$ ont la même valeur.)

N est l'égalité de nœud : $w_1 \equiv_N w_2$ ssi $w_1 = w_2$



Satisfaction d'une dépendance fonctionnelle

Définition

Un document \mathcal{D} satisfait la dépendance fonctionnelle (\mathcal{FD}, c) ssi :



Satisfaction d'une dépendance fonctionnelle

Définition

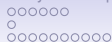
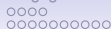
Un document \mathcal{D} satisfait la dépendance fonctionnelle (\mathcal{FD}, c) ssi :

Si pour deux traces, $\tau_1 = \text{trace}_{\pi_1}(\mathcal{FD}, \mathcal{D})$ et

$\tau_2 = \text{trace}_{\pi_2}(\mathcal{FD}, \mathcal{D})$, avec

(a) $\pi_1(c) =_N \pi_2(c)$

(b) $\forall i = 1, \dots, n, \pi_1(p_i) =_{E_i} \pi_2(p_i)$,



Satisfaction d'une dépendance fonctionnelle

Définition

Un document \mathcal{D} satisfait la dépendance fonctionnelle (\mathcal{FD}, c) ssi :

Si pour deux traces, $\tau_1 = \text{trace}_{\pi_1}(\mathcal{FD}, \mathcal{D})$ et

$\tau_2 = \text{trace}_{\pi_2}(\mathcal{FD}, \mathcal{D})$, avec

$$(a) \pi_1(c) =_N \pi_2(c)$$

$$(b) \forall i = 1, \dots, n, \pi_1(p_i) =_{E_i} \pi_2(p_i),$$

$$\text{Alors } \pi_1(q) =_{E_{n+1}} \pi_2(q)$$

Résultat sur l'analyse de dépendance

Proposition

Décider si une dépendance fonctionnelle (\mathcal{FD}, c) est indépendante par rapport à une classe de MAJ \mathcal{C} est un problème PSPACE-difficile



Un critère d'indépendance

Définition

Soit \mathcal{L} le langage des documents XML \mathcal{D} satisfaisant :

(i) $\mathcal{D} \in \text{valid}(Sc)$

(ii) $\exists \tau_{\mathcal{F}D} = \text{trace}_{\pi}(\mathcal{F}D, \mathcal{D})$, par rapport à un plongement π de $\mathcal{F}D$ dans \mathcal{D} ,
et $\exists \tau_{\mathcal{C}^p} = \text{trace}_{\pi'}(\mathcal{C}^p, \mathcal{D})$, par rapport à un plongement partiel π' de \mathcal{C} dans
 \mathcal{D} , tel que :

$$\mathcal{N}(\pi'(\vec{s}_c)) \cap (\mathcal{N}(\text{trace}_{\pi}(\mathcal{F}D, \mathcal{D})) \cup \mathcal{N}(\mathcal{F}D_{\pi}(\mathcal{D}))) \neq \emptyset$$

Proposition 1

Si \mathcal{L} est vide alors $(\mathcal{F}D, c)$ est indépendante par rapport à \mathcal{C}
dans le contexte Sc

Proposition 2

La vacuité du langage \mathcal{L} est testable en temps

$$O(a_c^2 a_{\mathcal{F}D}^2 \times |\Sigma|^4 \times |\mathcal{A}_{Sc}|^2 \times |\mathcal{C}|^2 \times |\mathcal{F}D|^2)$$

○○○
○○○○○○○○○○

○○○○○
○
○○○○○○○○○

○○
○
○○○
○○

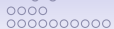
Conclusion

- Définition d'un langage formel de requêtes \mathcal{RAR}_S .
- Les requêtes \mathcal{RAR}_S expriment les requêtes de $CoreXPath^+$.
- Modélisation des Vues, MAJ et DFs par les requêtes \mathcal{RAR}_S .
- Problème de l'indépendance PSPACE-difficile.
- Condition suffisante d'indépendance entre Vues et MAJ.
- Condition suffisante d'indépendance entre DFs et MAJ.

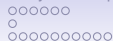


Perspectives

- Évaluation des \mathcal{RAR}_S .
- Étude de sous fragments par simplification des expressions régulières.
- Extension de l'analyse d'indépendance.
- Implémentation du critère d'indépendance.
- \mathcal{RAR}_S et autres contraintes d'intégrités.



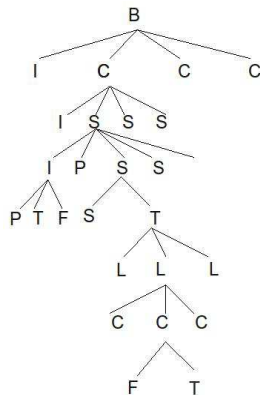
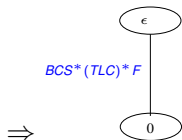
Merci de votre attention



Exemple d'une RAR

C : chapitre, I : introduction, S :
section, T : tableau, L : ligne, C :
colonne, F : figure

Exemple de requête : "Extraire toutes
les figures des tableaux"



Book document