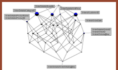
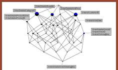


Triadic Approach to Conceptual Design of XML Data

Christian Săcărea
Viorica Varga

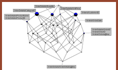
Babes-Bolyai University
Cluj Napoca

December 17, 2014

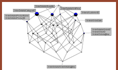


Outline

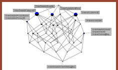
1. XML notions
2. FCA tool to detect XML FDs in Flat representation
3. Triadic approach for the hierarchical representation of XML data
4. Conclusions and Future Work



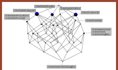
Outline



Outline

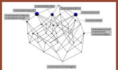


Outline



XML Design

- XML data design: choose an appropriate XML schema, which usually come in the form of DTD (Document Type Definition) or XML Scheme.
- Functional dependencies (FDs) are a key factor in XML design.
- The objective of normalization is to eliminate redundancies from an XML document, eliminate or reduce potential update anomalies.
- Arenas, M., Libkin, L.: A normal form for XML documents. TODS 29(1), 195-232 (2004)
- Yu, C., Jagadish, H. V.: XML schema refinement through redundancy detection and normalization. VLDB J. 17(2): 203-223 (2008)

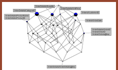


Data tree definition

Definition

An XML database is defined to be a rooted labeled tree $T = \langle N, \mathcal{P}, \mathcal{V}, n_r \rangle$, where:

- N is a set of labeled data nodes, each $n \in N$ has a label e and a node key that uniquely identifies it in T;
- $n_r \in N$ is the root node;
- $\mathcal{P}$ is a set of parent-child edges, there is exactly one $p = (n', n)$ in $\mathcal{P}$ for each $n \in N$ (except n_r), where $n' \in N, n \neq n', n'$ is called the parent node, n is called the child node;
- $\mathcal{V}$ is a set of value assignments, there is exactly one $v = (n, s)$ in $\mathcal{V}$ for each leaf node $n \in N$, where s is a value of simple type.



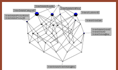
Schema definition

Definition

A schema is defined as a set $S = (E, T, r)$, where:

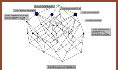
- *E is a finite set of element labels;*
- *T is a finite set of element types, and each $e \in E$ is associated with a $\tau \in T$, written as $(e : \tau)$, τ has the next form:*
  $\tau ::=$ **str** $|$ **int** $|$ **float** $|$ **SetOf** $\tau|$ **Rcd** $[e_1 : \tau_1, \ldots, e_n : \tau_n]$;
- *$r \in E$ is the label of the root element, whose associated element type can not be* **SetOf** *τ.*

- Types **str**, **int**, **float** are *simple types*
- **Rcd** for *complex scheme elements*,
- **SetOf** for *set schema elements*



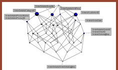
Path definition

- A schema element e_k can be identified through a path expression, $path(e_k) = /e_1/e_2/.../e_k$, where $e_1 = r$, and e_i is associated with type $\tau_i ::= \textbf{Rcd}\ [..., e_{i+1} : \tau_{i+1}, ...]$ for all $i \in [1, k-1]$.
- A path is *repeatable*, if e_k is a set element.
- We adopt XPath steps "." (self) and ".." (parent)

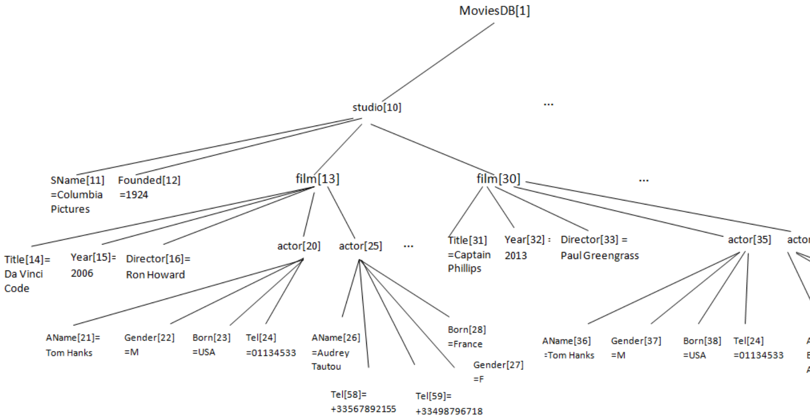


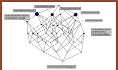
Descendant element definition

- We assign a node key, referred to as @key, to each data node in the data tree
- A data element n_k is a descendant of another data element n_1 if there exists a series of data elements n_i, such that $(n_i, n_{i+1}) \in \mathcal{P}$ for all $i \in [1, k-1]$.
- Element n_k is called a *direct descendant* of element n_a, if n_k is a descendant of n_a, $path(n_k) = \ldots/e_a/e_1/\ldots/e_{k-1}/e_k$, and e_i is not a set element for any $i \in [1, k-1]$.



Example XML data tree





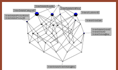
Element-value equality

Definition

Two data elements n_1 of $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1} \rangle$ and n_2 of $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2} \rangle$ are *element-value equal* (written as $n_1 =_{ev} n_2$) if and only if:

- n_1 and n_2 both exist and have the same label;
- There exists a set M, such that for every pair $(n'_1, n'_2) \in M$, $n'_1 =_{ev} n'_2$, where n'_1, n'_2 are children elements of n_1, n_2, respectively. Every child element of n_1 or n_2 appears in exactly one pair in M.
- $(n_1, s) \in \mathcal{V}_1$ if and only if $(n_2, s) \in \mathcal{V}_2$,where s is a simple value.

Example Data elements node 20 and 35 are *element value equal* if and only if the subtrees rooted at those two elements are identical when the order among sibling elements is ignored.



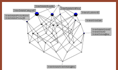
Path-value equality

Definition

Two data element paths p_1 on $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1} \rangle$ and p_2 on $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2} \rangle$ are *path-value equal* $(T_1.p_1 =_{pv} T_2.p_2)$ if and only if there is a set M' of matching pairs where

- For each pair $m' = (n_1, n_2)$ in M', $n_1 \in N_1$, $n_2 \in N_2$, $path(n_1) = p_1$, $path(n_2) = p_2$, and $n_1 =_{ev} n_2$;
- All data elements with path p_1 in T_1 and path p_2 in T_2 participate in M', and each such data element participates in only one such pair.

Two paths are value equal, if each node that is pointed to by one path have a corresponding node that is pointed to by the other path, where the two nodes are element-value equal.



Triadic Approach to Conceptual Design of XML Data

Christian Săcărea
Viorica Varga

XML notions

FCA tool to detect XML FDs in Flat representation

Triadic approach for the hierarchical representation of XML data

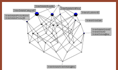
Conclusions and Future Work

Generalized tree tuple

Definition

A generalized tree tuple of data tree $T = \langle N, \mathcal{P}, \mathcal{V}, n_r \rangle$, with regard to a pivot data element n_p, is a tree $t^T_{n_p} = \langle N^t, \mathcal{P}^t, \mathcal{V}^t, n_r \rangle$, where:

- $N^t \subseteq N$ (nodes),$n_p \in N^t$; $\mathcal{P}^t \subseteq \mathcal{P}$ (parent-child edges);
- $\mathcal{V}^t \subseteq \mathcal{V}$ is the set of value assignments;
- n_r is the same root node in both $t^T_{n_p}$ and T;
- $n \in N^t$ if and only if:
  - n is a descendant or ancestor of n_p in T, or
  - n is a non-repeatable direct descendant of an ancestor of n_p in T ;
- $(n_1, n_2) \in \mathcal{P}^t$ if and only if $n_1, n_2 \in N^t$, $(n_1, n_2) \in \mathcal{P}$;
- $(n, s) \in \mathcal{V}^t$ if and only if $n \in N^t$, $(n, s) \in \mathcal{V}$.



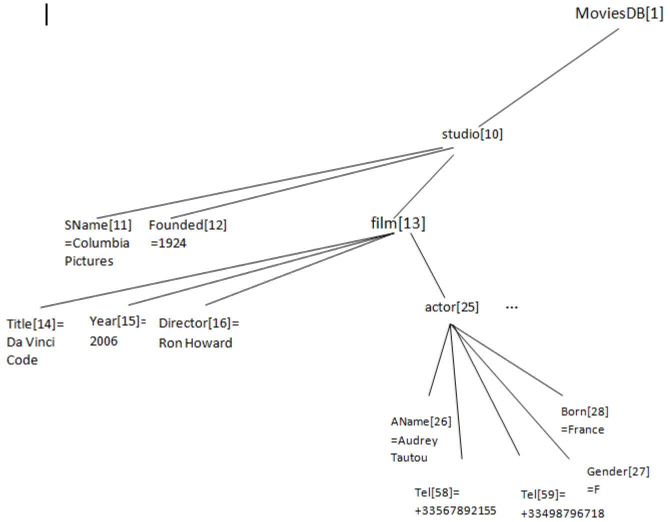
Tuple class

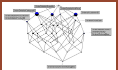
- A generalized tree tuple is a subtree of the XML data tree.
- It has an extra parameter called a pivot node.
- In contrast with tree tuple defined in Arenas and Libkin's article, which separate sibling nodes with the same path at all hierarchy levels, the generalized tree tuple separate sibling nodes with the same path only above the pivot node.
- Based on the pivot node, generalized tree tuples can be categorized into tuple classes:

Definition

A tuple class C_p^T of the data tree T is the set of all generalized tree tuples t_n^T, where $path(n) = p$. Path p is called the *pivot path*.

Generalized Tree Tuple MoviesDB/studio/film/actor[25]





XML Functional Dependency

Definition

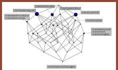
An XML FD is a triple $\langle C_p, LHS, RHS \rangle$, written as $LHS \rightarrow RHS$ w.r.t. C_p, where C_p denotes a tuple class, LHS is a set of paths $(P_{li}\ ,\ i = [1, n])$ relative to p, and RHS is a single path (P_r) relative to p.

An XML FD holds on a data tree T (or T satisfies an XML FD) if and only if for any two generalized tree tuples $t_1, t_2 \in C_p$

- $\exists i \in [1, n]$, $t_1.P_{li} = \perp$ or $t_2.P_{li} = \perp$, or

- If $\forall i \in [1, n]$, $t_1.P_{li} =_{pv} t_2.P_{li}$, then $t_1.P_r \neq \perp, t_2.P_r \neq \perp, t_1.P_r =_{pv} t_2.P_r$.

A null value, $\perp$, results from a path that matches no node in the tuple, and $=_{pv}$ is the path-value equality defined previous.

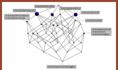


XML Functional Dependency Example

Example

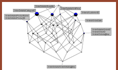
In MoviesDB Example whenever two actors agree on AName values, they have the same Gender.
This can be formulated as follows:
$./\mathrm{AName} \rightarrow ./\mathrm{Gender}$ w.r.t. $C_{MoviesDB/studio/film/actor}$
$./\mathrm{AName} \rightarrow ./\mathrm{Born}$ w.r.t $C_{MoviesDB/studio/film/actor}$
or
$\langle C_{actor}, ./AName, ./Tel \rangle$

This XML FD notion can effectively capture constraints involving set elements, like telephone.



Customer's Orders Example Scheme


```
CustOrder:Rcd
        Customers:SetOf Rcd
            CustomerID: str
            CompanyName: str
            Address: str
            City: str
            PostalCode:str
            Country: str
            Phone: str
            Orders: SetOf Rcd
                OrderID: int
                CustomerID: str
                OrderDate: str
                OrderDetails: SetOf Rcd
                    OrderID: int
                    ProductID: int
                    UnitPrice: float
                    Quantity: float
                    ProductName: str
                    CategoryID: int
```



XML Functional Dependency Example

Example

In Customer's Orders Example whenever two products agree on $\mathrm{ProductID}$ values, they have the same $\mathrm{ProductName}$.

This can be formulated as follows:

$./\mathrm{ProductID} \rightarrow ./\mathrm{ProductName}$ w.r.t. $C_{OrderDetails}$

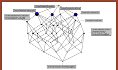
$./\mathrm{ProductID} \rightarrow ./\mathrm{CategoryID}$ w.r.t $C_{OrderDetails}$

other examples

$$\langle C_{Orders}, ./OrderID, ./CustomerID\rangle$$

$$\langle C_{Orders}, ./Orders@key, ./CustomerID\rangle$$

$$\langle C_{Orders}, ./OrderDetail/OrderID, ./CustomerID\rangle$$

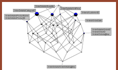


XML key

Definition (XML key) An XML Key of a data tree T is a pair $\langle C_p, LHS\rangle$, where T satisfies the XML FD $\langle C_p, LHS, ./@key\rangle$.

Example

We have the XML FD: $\langle C_{Orders}, ./OrderID, ./@key\rangle$, which implies that $\langle C_{Orders}, ./OrderID\rangle$ is an XML key.

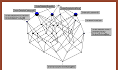


Interesting XML FD

Tuple classes with repeatable pivot paths are called *essential tuple classes*.

Definition An XML FD $\langle C_p, LHS, RHS\rangle$ is *interesting* if it satisfies the following conditions:

- RHS $\notin$ LHS;
- C_p is an essential tuple class;
- RHS matches to descendent(s) of the pivot node.



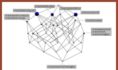
XML data redundancy

Definition

A data tree T contains a redundancy if and only if T satisfies an interesting XML FD $\langle C_p, LHS, RHS\rangle$, but does not satisfy the XML Key $\langle C_p, LHS\rangle$.

Intuitively:

- if $\langle C_p, LHS\rangle$ is not a key for T, then there exists two distinct tuples in C_p that share the same LHS.
- T satisfies $\langle C_p, LHS, RHS\rangle$, so RHS of these two tuples must be value equal
- so: data is redundantly stored

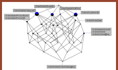


GTT-XNF

Definition

An XML schema *S* is in GTT-XNF given the set of all satisfied interesting XML FDs if and only if for each such XML FD $(\langle C_p, LHS, RHS\rangle)$, $\langle C_p, LHS\rangle$ is an XML key.

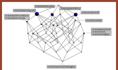
Intuitively: GTT-XNF disallows any satisfied interesting XML FD that indicates data redundancies.



Data representation for XML data

- flat representation as a single relational table
- hierarchical, one relational table for every level of the tree

Our recent work:
K.T. Janosi-Rancz, V. Varga, T. Nagy: *Detecting XML Functional Dependencies through Formal Concept Analysis*, 14th East European Conference on Advances in Databases and Information Systems (ADBIS), Novi Sad, Serbia, LNCS 6295, pp. 595-598 (2010).
K.T. Janosi-Rancz, V. Varga.: *XML Schema Refinement Through Formal Concept Analysis*, Studia Univ. Babes-Bolyai Cluj-Napoca, Informatica, vol. LVII, No. 3, pp. 49-64 (2012)



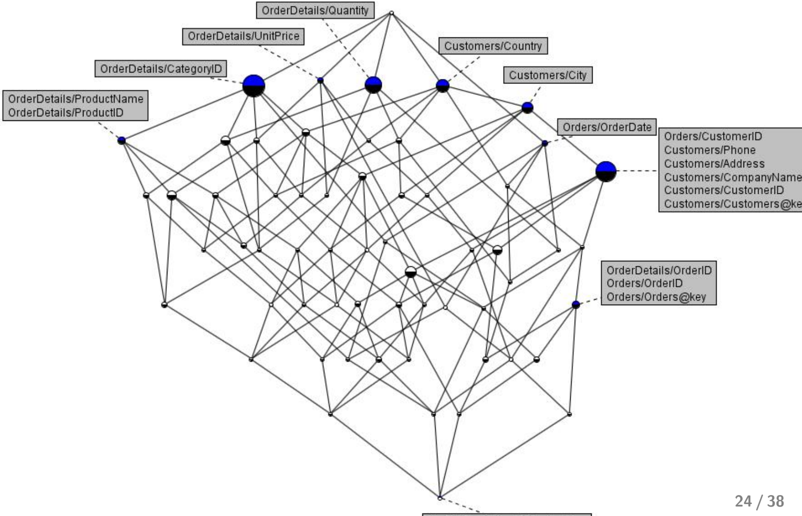
FCA tool to detect XML FDs in Flat representation

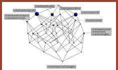
Context of functional dependencies:
(G, M, I), where $G = \{(t_1, t_2) | t_1, t_2 \in r, t_1 \neq t_2\}$ and
$\forall m \in M$, $(t_1, t_2) Im \Leftrightarrow t_1[m] = t_2[m]$.

- in the flat table we insert non-leaf and leaf level elements (or attributes) from the tree
  - for non-leaf level nodes: the name of the attribute is constructed as: <ElementName>+"@key" and its value will be the associated key value
  - for non-leaf level nodes: the element names of the leaves and its value.
- the incidence relation of the context shows which attributes of this tuple pairs have the same value.

A functional dependency $X \rightarrow Y$ $(X, Y \subset M)$ holds in a relation r over M iff the implication $X \rightarrow Y$ holds in the context (G, M, I).

Concept Lattice of functional dependencies' Formal Context for tuple class $C_{OrderDetail}$





The functional dependencies found by software FCAMineXFD

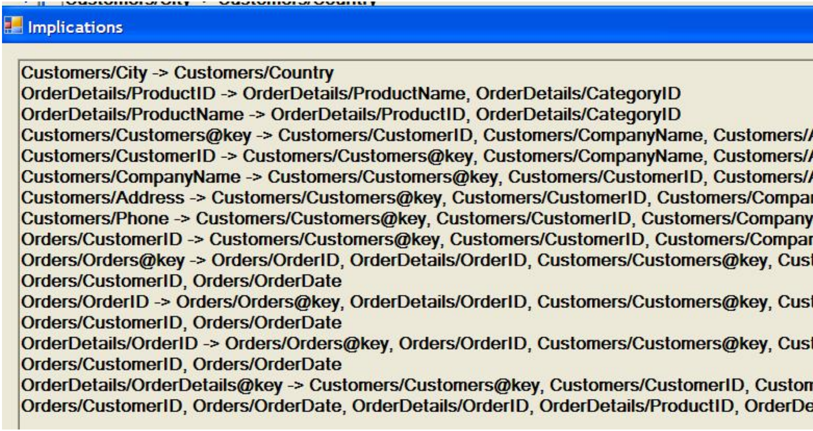
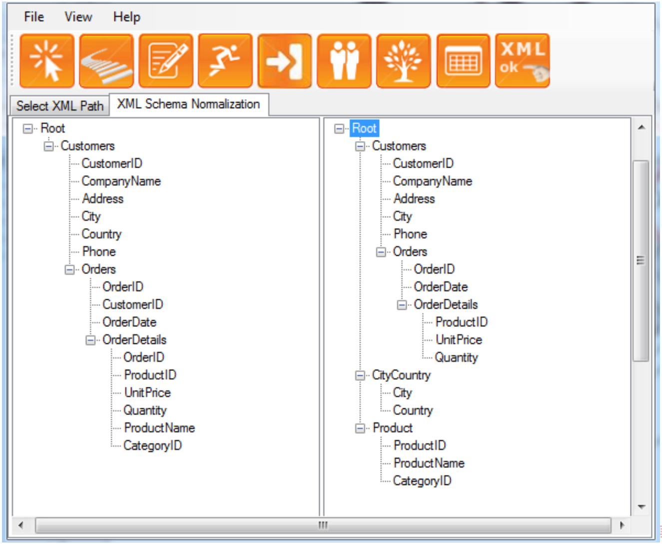
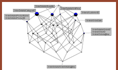


Figure : Functional dependencies in tuple class $C_{OrderDetails}$

Correct XML Scheme





Hierarchical representation of Movies XML data

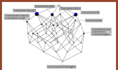
```
MoviesDB (studio*)
    studio (SName, Founded, film*)
        film (Title, Year, Director, actor*)
            actor (AName, Gender, Born, Tel*, BornY?)
```

Table : R_{root}

@key	parent
1	$\perp$

Table : R_{Studio}

@key	parent	SName	Founded
10	1	Columbia Pictures	1924
50	1	Warner Bros. Pictures	1923



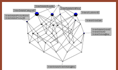
Hierarchical representation of Movies XML data

Table : R_{Film}

@key	parent	Title	Year	Director
13	10	Da Vinci Code	2006	Ron Howard
30	10	Captain Phillips	2013	Paul Greengrass
55	50	Extremely Loud & Incredibly Close	2011	Stephen Daldry
80	50	Gravity	2013	Alphonso Cuarón

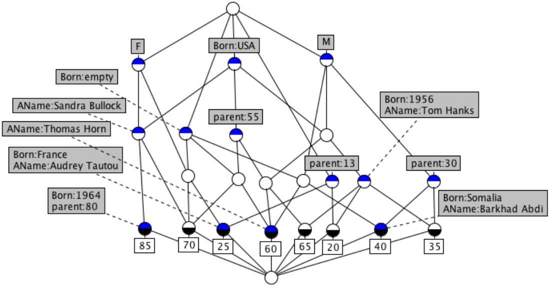
Table : R_{Actor}

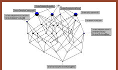
@key	parent	AName	Gender	Born	BornYear
20	13	Tom Hanks	M	USA	1956
25	13	Audrey Tautou	F	France	$\perp$
35	30	Tom Hanks	M	USA	1956
40	30	Barkhad Abdi	M	Somalia	$\perp$
60	55	Thomas Horn	M	USA	$\perp$
65	55	Tom Hanks	M	USA	1956
70	55	Sandra Bullock	F	USA	$\perp$
85	80	Sandra Bullock	F	USA	1964



Conceptual Structure of the Movie Multicontext

- every nested table is considered as a many-valued context
- through conceptual scaling a multi-context is obtained wherefrom we can construct a tricontext
- multicontext $\mathbb{K}_{Movie}$ formed by four contexts





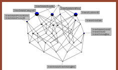
Triadic Formal Context

Definition

A triadic formal context is a quadruple $\mathbb{K} := (K_1, K_2, K_3, Y)$ where K_1, K_2 and K_3 are sets, and Y is a ternary relation between them, i. e., $Y \subseteq K_1 \times K_2 \times K_3$. An element $(g, m, b) \in Y$ is read *object* g *has attribute* m *under condition* b.

Definition

A multicontext of signature $\sigma \colon P \to I^2$, where I and P are non-empty sets, is be defined as a pair (S_I, R_P) consisting of a family $S_I := (S_i)_{i \in I}$ of sets and a family $R_P := (R_p)_{p \in P}$ of binary relations with $R_p \subseteq S_i \times S_j$ if $\sigma p = (i, j)$. A multicontext $\mathbb{K} := (S_I, R_P)$ can be understood as a *network* of formal contexts $\mathbb{K}_p := (S_i, S_j, R_p)$, with $p \in P$ and $\sigma p = (i, j)$.



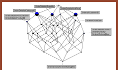
Triadic Formal Context for Hierarhical XML data

For a multicontext $\mathbb{K} := (S_I, R_P)$ of signature $\sigma\colon P \to I^2$, let $I_1 := \{i \in I \mid \sigma p = (i,j) \text{ for some } p \in P\}$ and $I_2 := \{j \in I \mid \sigma p = (i,j) \text{ for some } p \in P\}$. Let $G_{\mathbb{K}} := \bigcup_{i \in I_1} S_i$ and $M_{\mathbb{K}} := \bigcup_{j \in I_2} S_j$. We can define a *triadic context* by $\mathbb{T}_{\mathbb{K}} := (G_{\mathbb{K}}, M_{\mathbb{K}}, P, Y_{\mathbb{K}})$ with $Y_{\mathbb{K}} := \{(g,m,p) \in G_{\mathbb{K}} \times M_{\mathbb{K}} \times P \mid (g,m) \in R_p\}$.

Example

For the triadic context generated by the *Movies* multicontext

- $G := \{1, 10, 50, 13, 30, 55, 80, 20, 25, 35, 40, 60, 65, 70, 85\}$,
- the attribute set is obtained by taking all nominally scaled attributes of the four many-valued contexts
- the condition set contains the labels of the hierarchical levels from root to levels.



Derivation Operator

Definition

For $\{i, j, k\} = \{1, 2, 3\}$ with $j < k$ and for $X \subseteq K_i$ and $Z \subseteq K_j \times K_k$, the $(-)^{(i)}$-derivation operators are defined by

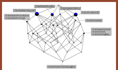
$$X \mapsto X^{(i)} = \{(a_j, a_k) \in K_j \times K_k \mid (a_i, a_j, a_k) \in Y \forall a_i \in X\},$$

$$Z \mapsto Z^{(i)} = \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \forall (a_j, a_k) \in Z\}.$$

Example

In the tricontext *Movies*, if g is an object, i.e., is a key of a node in the XML dataset at level l, then $i = 1, j = 2, k = 3$ and $g^{(1)}$ is the set of nominally scaled attributes at level l.

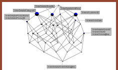
$$g^{(1)} = \{(a, l) \in K_2 \times K_3 \mid (g, a, l) \in Y\}.$$



Generalized Tree Tuple

We will denote $g^{(1)}$ by $g_{(1)}$.

- In $g_{(1)}$ we have the parent value between scaled attributes.
- In order to obtain the generalized tree tuple with pivot node p at level l we have to traverse the tree up from p, by parent value until we reach the root, where the parent has NULL value.



Generalized Tree Tuple with pivot node at level l

We apply the derivation operator for object g at level l until its parent value is different from NULL as follows :

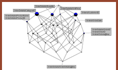
$$g'_{(1)} = \{(a, l-1) \in K_2 \times K_3 \mid (g', a, l-1) \in Y, g[parent] = g'[@key]\}$$

$$g''_{(1)} = \{(a, l-2) \in K_2 \times K_3 \mid (g'', a, l-2) \in Y, g'[parent] = g''[@key]\}$$

$$\dots$$

$$g^{(l-1)}_{(1)} = \{(a, 1) \in K_2 \times K_3 \mid (g^{(l-1)}, a, 1) \in Y,$$
$$g^{(l-2)}[parent] = g^{(l-1)}[@key]\}$$

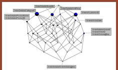
$$GTT_p = g'_{(1)} \bigcup g''_{(1)} \dots \bigcup g^{(l-1)}_{(1)}$$



Inter-relational Functional Dependencies

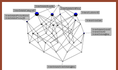
Definition

Let $\mathbb{K}$ be the tricontext resulting from the canonical translation of an XML database. Let C_p be a tuple class. Then, the *formal context of functional dependencies with respect to* C_p is defined as $\mathrm{XMLFD}(\mathbb{K}) := (C_p \times C_p, M, I)$, where M is the set of paths, and $((g,h),q) \in I$ if and only if the path values of g and h on path q are equal with regard to path-value equality.



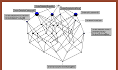
Inter-relational Functional Dependencies

- XML FDs that involve a single relation of the hierarchical representation are intra-relation FDs,
- XML FDs that involve multiple relations inter-relation FDs.
- The inter-relational functional dependencies of an XML database are exactly the attribute implications in $\mathrm{XMLFD}(\mathbb{K})$.



Conclusions

- FCA proves to be a valuable tool for the conceptual design of XML data.
- XML data can be represented in hierarchical or flat form.
- In a recent work we give an FCA based approach for mining functional dependencies for flat XML data representation.
- In this paper we define a triadic FCA approach for a conceptual model of hierarchical XML data representation.



Future Work

- We propose to develop a software which will build the tricontext of an XML tree
- and will give the functional dependencies from XML tree.