Claves XML: Una Implementación de Algoritmos de Implicación y Validación

Emir F. Muñoz Jiménez

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

Examen de Grado

Dr. Mauricio Marín - Dr. Flavio Ferrarotti

Lunes 30 de mayo de 2011

Introducción

Claves XML

- Las claves son fundamentales para la administración de bases de datos (independiente del modelo usado)
- En la última década, varias nociones de claves han sido propuestas
  - La más popular es la propuesta de Buneman et al.,2002 y 2003 basada en un árbol XML
- Las claves identifican nodos en árboles XML, en base a los valores de los nodos descendientes
- Su expresividad y propiedades computacionales han sido analizadas en la teoría

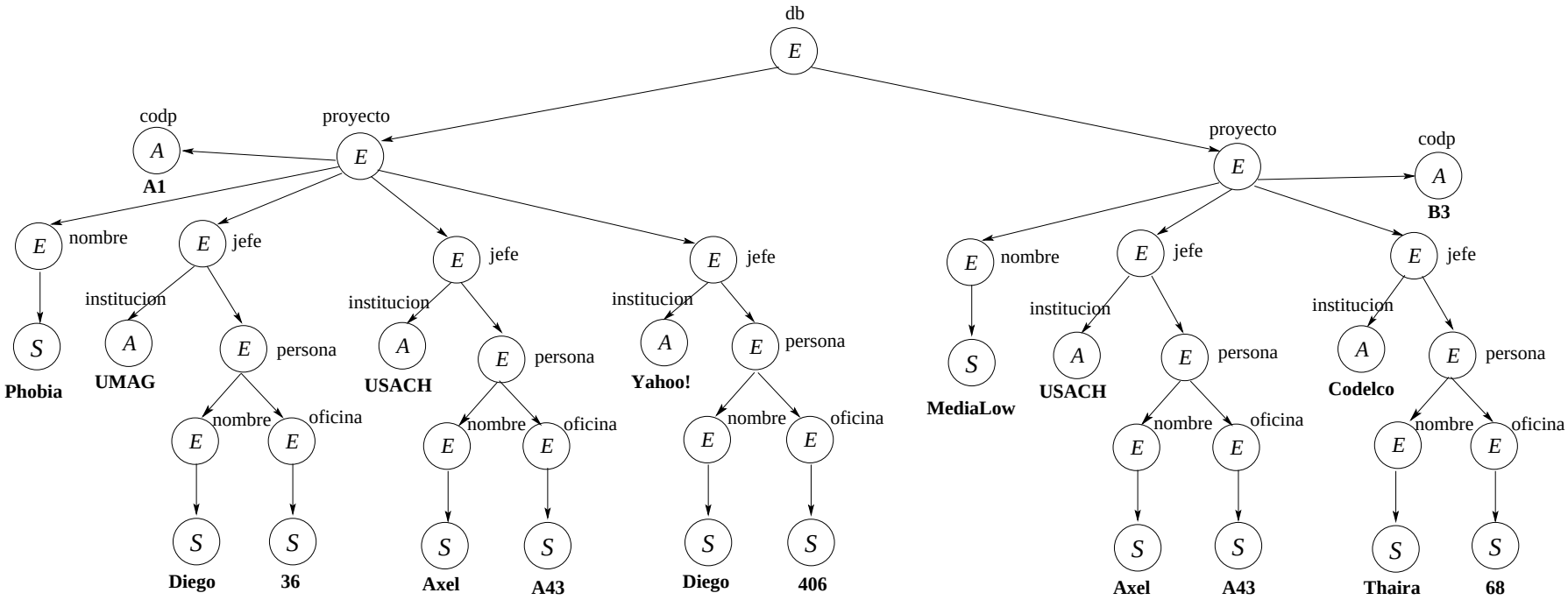
Introducción

Claves XML (cont.)

- Tanto la industria como la academia han reconocido la importancia de las claves en la administración de datos XML
- Por otro lado, en la práctica, las nociones de expresividad de las claves XML han sido **ignoradas**
- Se espera que las claves XML sean tan útiles como en el modelo relacional
- Así, las claves tendrán gran potencial en áreas como: diseño de esquemas, optimización de consultas, almacenamiento y actualización, intercambio de datos e integración

Introducción

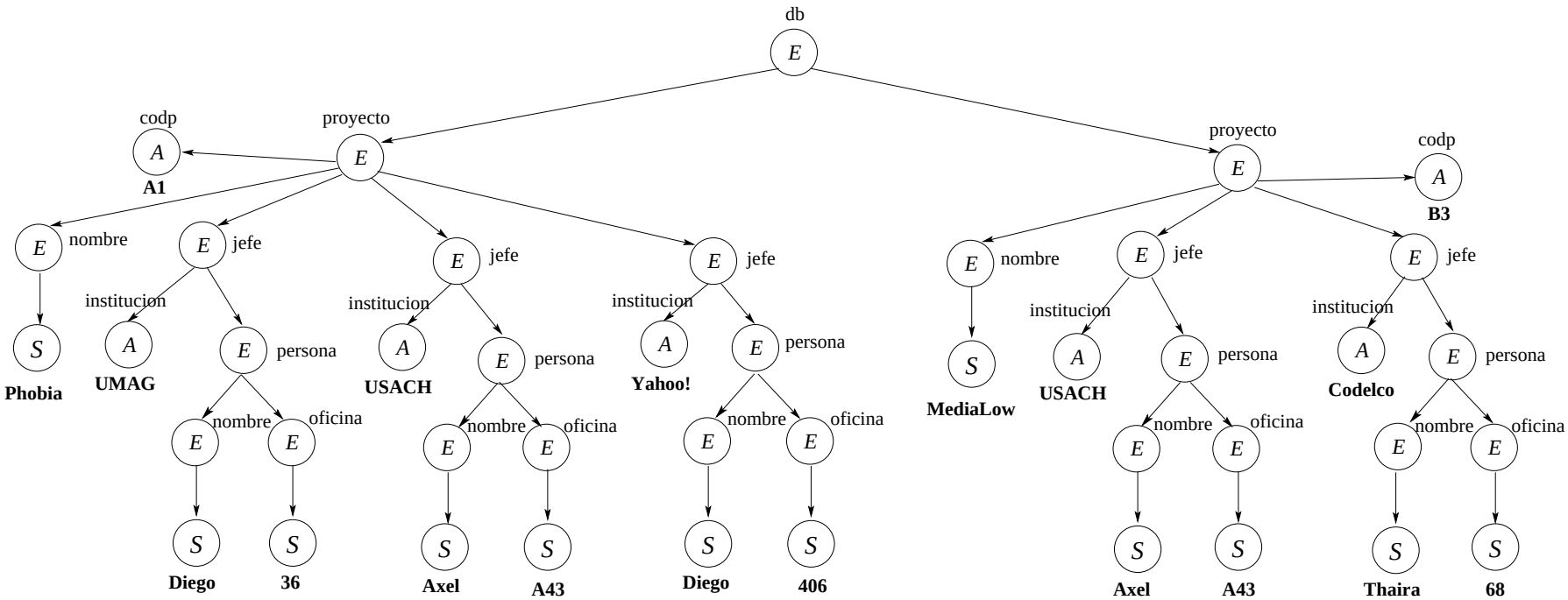
Claves XML (cont.)



Un nodo *proyecto* es identificado por *codp* en todo el documento

Introducción

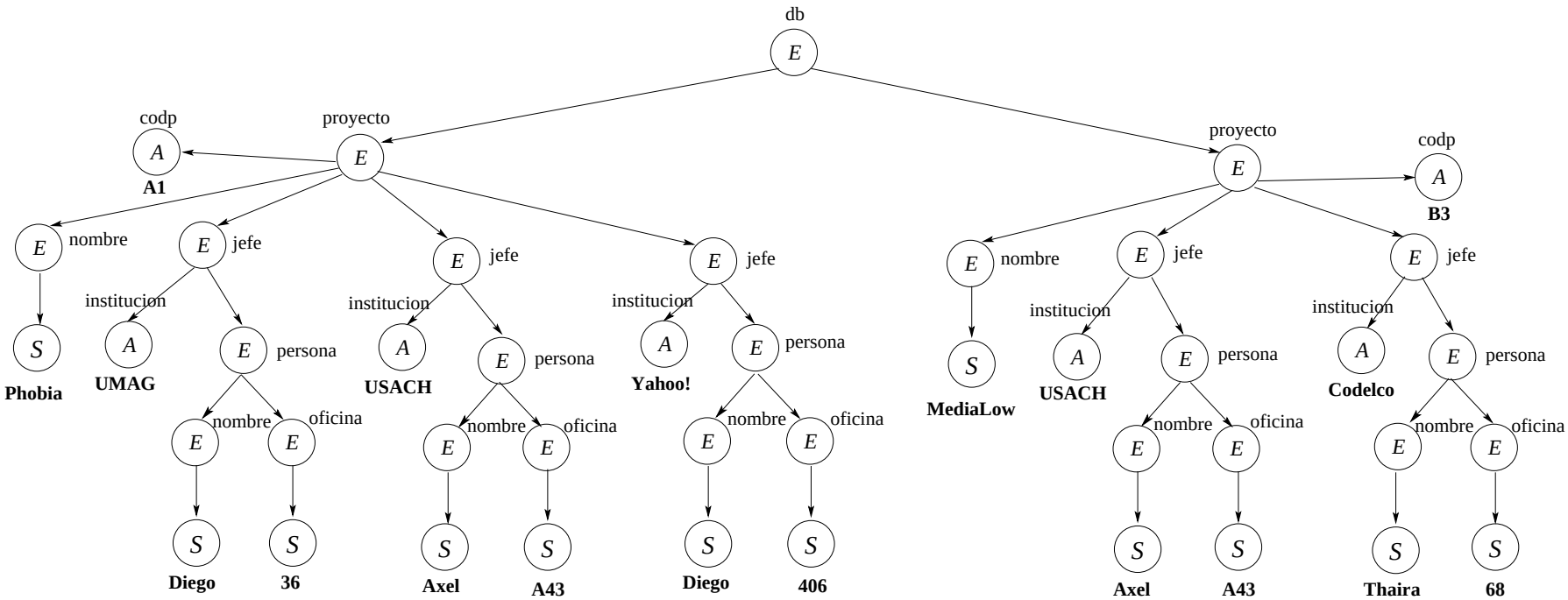
Claves XML (cont.)



Un nodo *jefe* puede ser identificado por *institucion*, relativo a un nodo *proyecto*

Introducción

Claves XML (cont.)



Un nodo *jefe* puede ser identificado por sus nodos hijos *institucion* y *persona*

Introducción

Claves XML (cont.)

Claves definidas

(a). Un nodo *proyecto* es identificado por *codp* en todo el documento (Clave Absoluta)

(b). Un nodo *jefe* puede ser identificado por *institucion*, relativo a un nodo *proyecto* (Clave Relativa)

(c). Un nodo *jefe* puede ser identificado por sus nodos hijos *institucion* y *persona* (Clave Relativa)

Nótese que

Hay claves que implican otras claves. Como la clave (b) que implica la (c). (superclave)

Introducción

Claves XML (cont.)

Claves definidas

(a). Un nodo *proyecto* es identificado por *codp* en todo el documento (Clave Absoluta)

(b). Un nodo *jefe* puede ser identificado por *institucion*, relativo a un nodo *proyecto* (Clave Relativa)

(c). Un nodo *jefe* puede ser identificado por sus nodos hijos *institucion* y *persona* (Clave Relativa)

Nótese que

Hay claves que implican otras claves. Como la clave (b) que implica la (c). (superclave)

Introducción

Claves XML (cont.)

Es importante investigar las claves XML

Las claves XML tienen un gran potencial para áreas como:

- Diseño de esquemas
- Optimización y reescritura de consultas
- Almacenamiento y actualización de datos
- Intercambio e integración de datos
- Indexación

Introducción

Claves XML (cont.)

¿Qué se ha hecho en el área?

- Propuestas de definición de restricciones en XML [Arenas et al., 2002, Fan, 2005, Fan and Siméon, 2003, Fan and Libkin, 2002, Suciu, 2001, Vianu, 2003]
- Las principales son las claves absolutas y relativas [Buneman et al., 2002, Buneman et al., 2003]
  - Se estudia el problema de implicación de claves XML [Buneman et al., 2002, Buneman et al., 2003]
  - Se logra un algoritmo $\mathcal{O}(n^7)$
- El conjunto de reglas presentadas no era válido [Hartmann and Link, 2009]
- Se propone un nuevo algoritmo $\mathcal{O}(n^2)$, para un fragmento de claves
- Es un área de investigación con gran complejidad teórica

Contribuciones

- Disminuir la brecha entre teoría y práctica de las claves XML
- Las contribuciones realizadas por este trabajo son:
  - (i). Una implementación eficiente para un algoritmo de implicación de claves XML, propuesto en [Hartmann and Link, 2009]
  - (ii). El diseño e implementación de un algoritmo de validación de documentos contra claves XML.
  - (iii). El diseño e implementación de un algoritmo para calcular *covers* no redundantes.

Objetivo General

El presente trabajo tuvo como objetivo general:

Demostrar empíricamente las capacidades semánticas que entregan las claves XML, mediante el desarrollo de *software* para los problemas de implicación y validación, para las claves definidas por [Buneman et al., 2003], y estudiadas en [Hartmann and Link, 2009].

Marco Teórico

Modelo de Árbol XML

- **E** denota el conjunto de etiquetas de elementos
- **A** denota el conjunto de nombres de atributos
- S el conjunto unitario que denota texto (PCDATA)
- Sea $\mathcal{L} = \mathbf{E} \cup \mathbf{A} \cup \{S\}$, el conjunto de *etiquetas*

Definición

Un árbol XML es una 6-tupla $T = (V, lab, ele, att, val, r)$

Marco Teórico

Modelo de Árbol XML

- **E** denota el conjunto de etiquetas de elementos
- **A** denota el conjunto de nombres de atributos
- S el conjunto unitario que denota texto (PCDATA)
- Sea $\mathcal{L} = \mathbf{E} \cup \mathbf{A} \cup \{S\}$, el conjunto de *etiquetas*

Definición

Un árbol XML es una 6-tupla $T = (V, lab, ele, att, val, r)$

Marco Teórico

Modelo de Árbol XML (cont.)

- Un *camino* en un árbol XML T es una secuencia finita $v_0, \ldots, v_m$, tal que (v_{i-1}, v_i) es una arista de T para todo $i = 1, \ldots, m$
- El camino p determina una palabra $lab(v_1), \ldots, lab(v_m)$ sobre el alfabeto $\mathcal{L}$
- Se llama p a un camino desde v_0 hasta v_m, y se dice que v_m es *alcanzable* desde v_0
- Para cada nodo $v \in V$, un nodo w alcanzable desde v es llamado *descendiente* de v

Marco Teórico

Igualdad en Valor

Dos nodos u y v se consideran iguales en valor $(u =_v v)$ si:

(a) $lab(u) = lab(v)$,

(b) si u, v son nodos atributo o texto, entonces $val(u) = val(v)$,

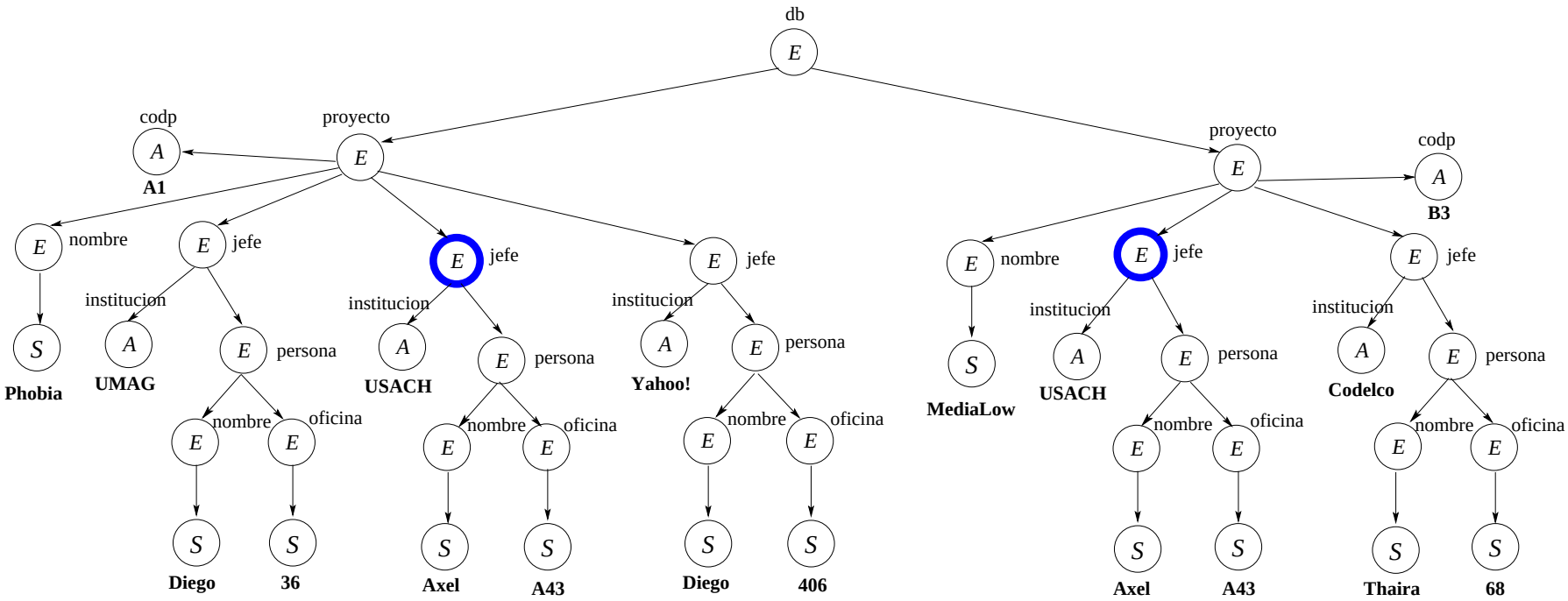
(c) si u, v son nodos elemento, entonces:

- (i) si $att(u) = \{a_1, \ldots, a_m\}$, entonces $att(v) = \{a'_1, \ldots, a'_m\}$ y existe una permutación π sobre $\{1, \ldots, m\}$ tal que $a_i =_v a'_{\pi(i)}$ para $i = 1, \ldots, m$, y
- (ii) si $ele(u) = [u_1, \ldots, u_k]$, entonces $ele(v) = [v_1, \ldots, v_k]$ y $u_i =_v v_i$ para $i = 1, \ldots, k$.

Determinar isomorfismo entre subárboles

Marco Teórico

Igualdad en Valor (cont.)



Nodos *jefe*, correspondientes a **Axel** con oficina **A43** son iguales en valor

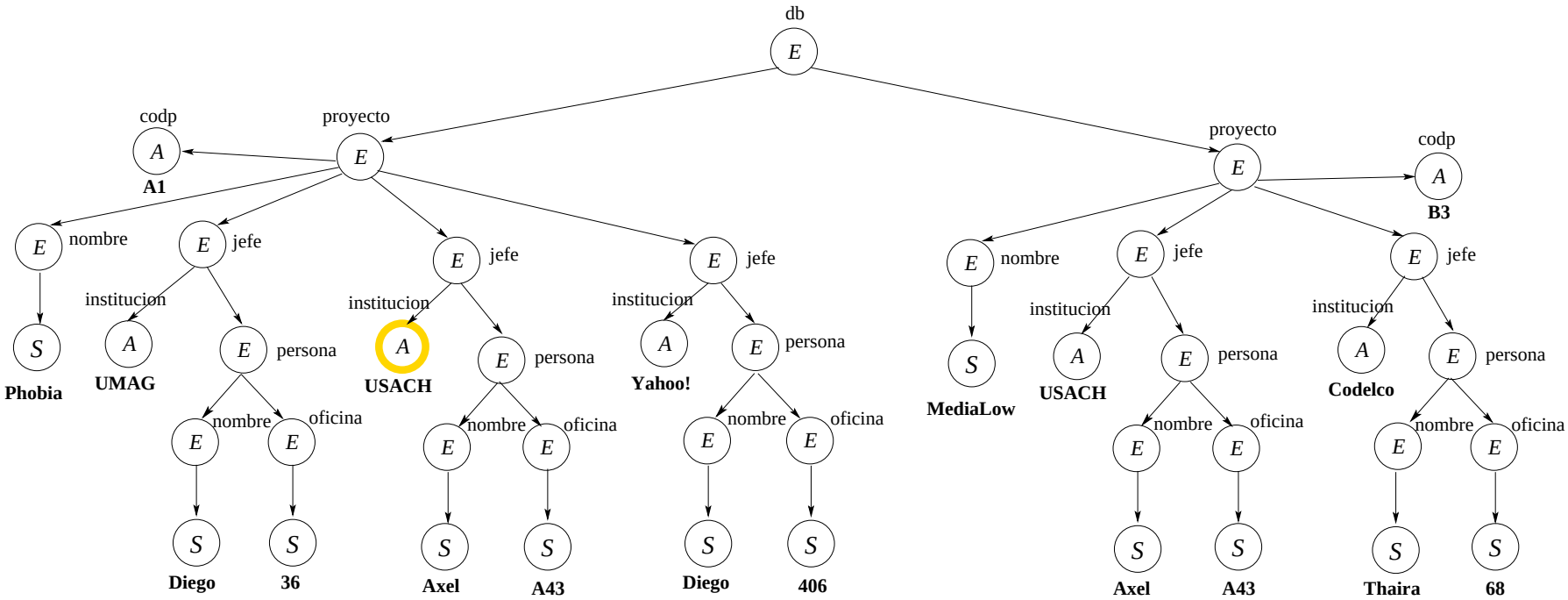
Marco Teórico

Expresiones de Camino

- Se construyen usando las etiquetas de los nodos como alfabeto, describiendo una secuencia de aristas padre-hijo
- Definimos un lenguaje PL a partir de la gramática $Q \rightarrow \ell \mid \varepsilon \mid Q.Q \mid \_^*$
- donde $\ell \in \mathcal{L}$ es alguna etiqueta, ε la palabra vacía, "." el operador concatenación, y "_*" el comodín
- Por ejemplo, una expresión de camino puede ser $proyecto.jefe.persona.nombre$

Marco Teórico

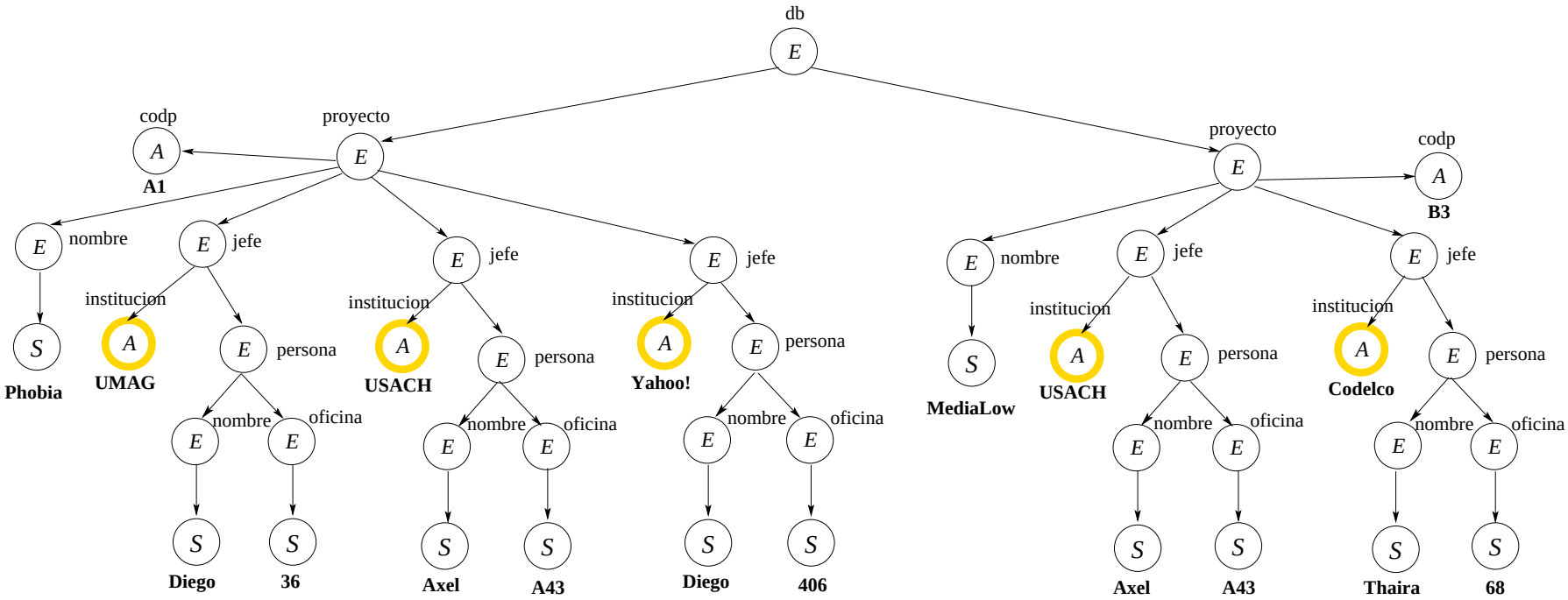
Expresiones de Camino (cont.)



- Para los nodos $v, w \in V$, escribimos $T \models Q(v, w)$ si w es alcanzable desde v, siguiendo un camino-Q en T
- Si el camino-Q fuese $proyecto.jefe.institucion$ y v el nodo raíz

Marco Teórico

Expresiones de Camino (cont.)



- Para un nodo $v \in V$, sea $v[\![Q]\!]$ el conjunto de nodos en T alcanzables desde v siguiendo un camino-Q
- Si el camino-Q fuese $proyecto.jefe.institucion$ y v el nodo raíz

Marco Teórico

Expresiones de Camino (cont.)

- Denotamos PL_s al subconjunto de expresiones PL que contienen todas las palabras sobre el alfabeto $\mathcal{L}$ (no contienen comodines)
- $Q \in PL$ es *válida* si no posee etiquetas $\ell \in \mathbf{A}$ o $\ell = S$ en una posición que no sea la última
- Para los nodos v y v' de un árbol XML T, la *intersección en valor* de $v[\![Q]\!]$ y $v'[\![Q]\!]$ es dada por
$v[\![Q]\!] \cap_v v'[\![Q]\!] = \{(w, w') \mid w \in v[\![Q]\!], w' \in v'[\![Q]\!], w =_v w'\}$

Marco Teórico

Claves para XML

Definición (Clave XML)

Una clave XML φ en una clase $\mathcal{K}(PL_1, PL_2, PL_3)$ de claves XML, es una expresión de la forma $(Q, (Q', \{Q_1, \ldots, Q_k\}))$ donde,

- *Q es una expresión PL_1 (Camino Contexto)*
- *Q' es una expresión PL_2 (Camino Objetivo)*
- *Para $1 \leq i \leq k$, Q_i es una expresión PL_3 (Caminos Clave)*
- *Tal que $Q.Q'.Q_i$ es una expresión válida*
- *Si $Q = \varepsilon$, φ es absoluta; en otro caso, φ es relativa*

Marco Teórico

Claves para XML (cont.)

- Un árbol XML T *satisface* una clave $(Q, (Q', \{Q_1, \ldots, Q_k\}))$ si y sólo si, para todo nodo $q \in [\![Q]\!]$ y todo par de nodos $q'_1, q'_2 \in q[\![Q']\!]$, tal que existen los nodos $x_i \in q'_1[\![Q_i]\!]$, $y_i \in q'_2[\![Q_i]\!]$, con $x_i =_v y_i$ para todo $i = 1, \ldots, k$, se tiene que $q'_1 = q'_2$
- Se trabajó concretamente con la clase $\mathcal{K}(PL, PL, PL_s^+)$ de claves XML mostrada en [Hartmann and Link, 2009]
- Esta clase representa un buen balance entre expresividad de claves XML y complejidad de su problema de implicación
- Donde "+" indica que el conjunto de caminos clave es finito y no vacío

Marco Teórico

Claves para XML (cont.)

Formalización de claves

En particular, las claves descritas informalmente en la introducción, pertenecen a esta clase y pueden ser expresadas como:

-Clave (a): $(\varepsilon, (proyecto, \{codp\}))$

-Clave (b): $(proyecto, (jefe, \{institucion\}))$

-Clave (c): $(proyecto, (jefe, \{institucion, persona\}))$

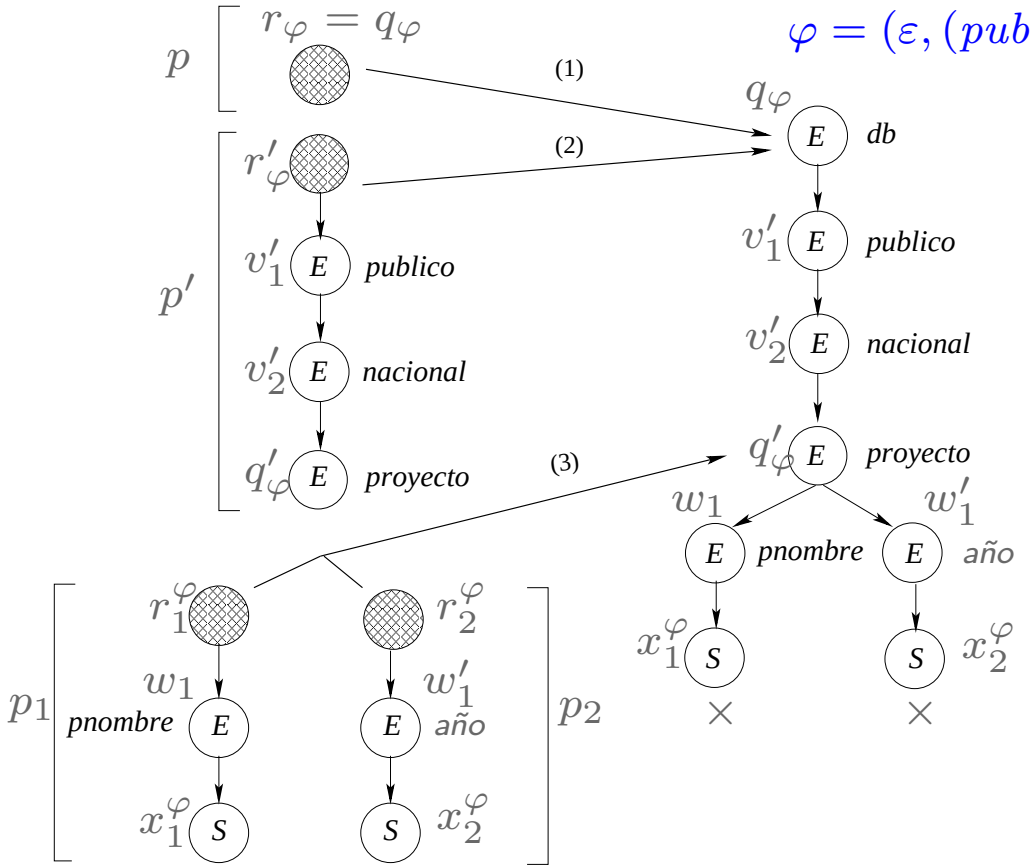
Marco Teórico

Implicación de claves y Axiomatización

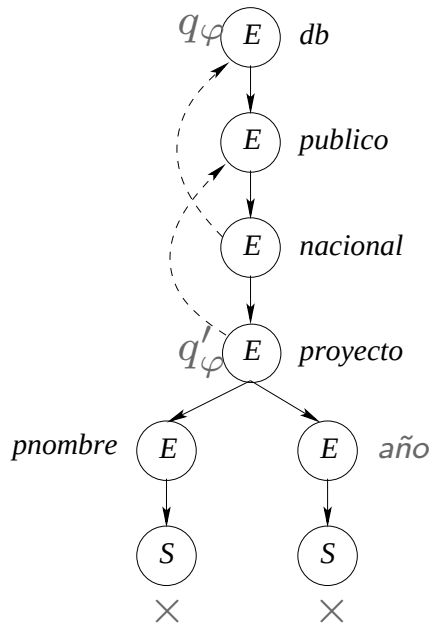
- Decimos que Σ implica φ, denotado $\Sigma \models \varphi$, si todo árbol XML T que satisface toda $\sigma \in \Sigma$ también satisface φ
- El problema de implicación es decidir para algún conjunto de claves $\Sigma \cup \{\varphi\}$ en una clase $\mathcal{C}$, si $\Sigma \models \varphi$
- Definimos *clausura semántica* como: $\Sigma^* = \{\varphi \in \mathcal{C} \mid \Sigma \models \varphi\}$
- Definimos *clausura sintáctica* como: $\Sigma^+_{\Re} = \{\varphi \mid \Sigma \vdash_{\Re} \varphi\}$
- Un conjunto de reglas $\Re$ es *válido (completo)* si $\Sigma^+_{\Re} \subseteq \Sigma^* (\Sigma^* \subseteq \Sigma^+_{\Re})$
- A un conjunto válido y completo de reglas se le llama *axiomatización*

Implicación de Claves XML

Mini-tree y Witness-graph [Hartmann and Link, 2009]



(a) Mini-tree $T_{\Sigma,\varphi}$



(b) Witness-graph $G_{\Sigma,\varphi}$

Figure: Herramientas de la implicación.

Implicación de Claves XML

Algoritmo de Implicación de claves XML

Algoritmo 1: $\Sigma \models \varphi$: Implicación de claves XML

Entrada: Conjunto finito de claves XML $\Sigma \cup \{\varphi\}$ en $\mathcal{K}(PL, PL, PL_s^+)$
Salida : Sí, si $\Sigma \models \varphi$; No, en otro caso

```
1 Construir G_{Σ,φ} para Σ y φ;
2 if q_φ es alcanzable desde q'_φ en G_{Σ,φ} then
3 |     return si;
4 else
5 |     return no;
```

Implicación de Claves XML

Teorema ([Hartmann and Link, 2009])

Sea $\Sigma \cup \{\varphi\}$ un conjunto finito de claves en la clase $\mathcal{K}(PL, PL, PL_s^+)$. Se tiene que $\Sigma \models \varphi$ si y sólo si, q_φ es alcanzable desde q'_φ en $G_{\Sigma,\varphi}$.

Teorema ([Hartmann and Link, 2009])

Sea $\Sigma \cup \{\varphi\}$ un conjunto finito de claves XML en $\mathcal{K}(PL, PL, PL_s^+)$. El problema de implicación $\Sigma \models \varphi$ se puede decidir en tiempo $\mathcal{O}(|\varphi| \times (||\Sigma|| + |\varphi|))$.

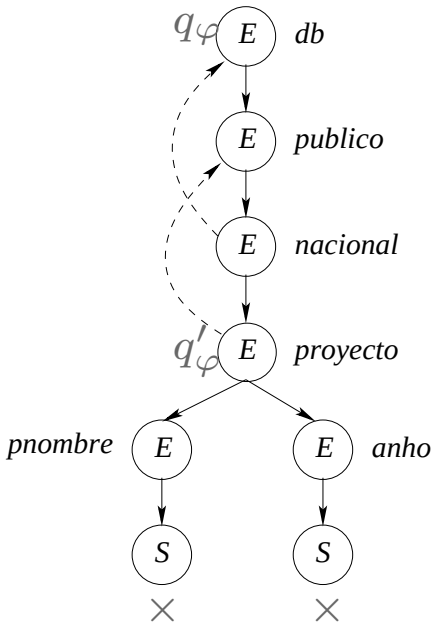
Implicación de Claves XML

Implementación

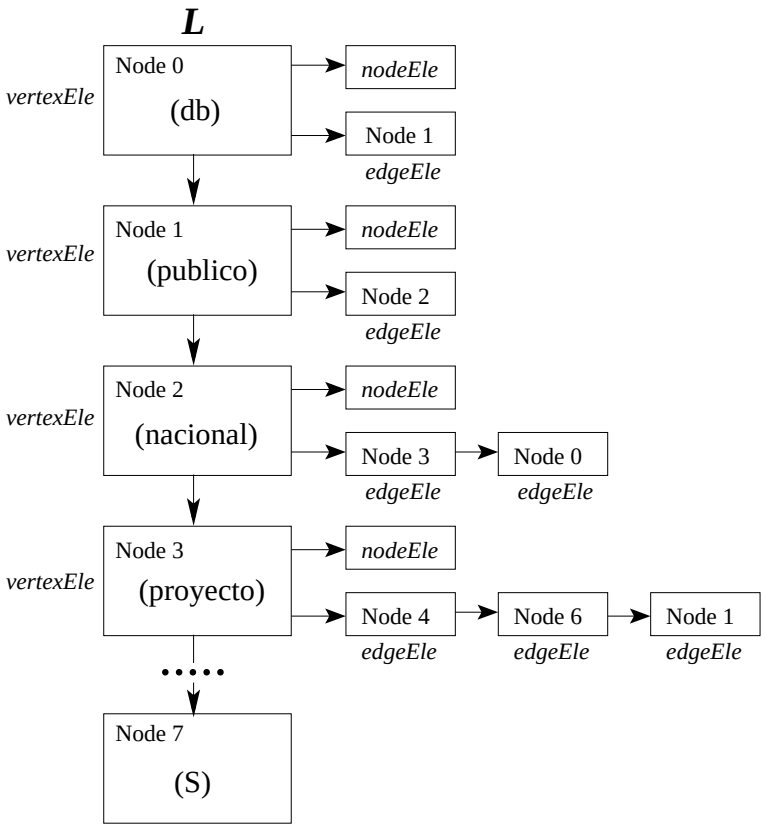
- Definir estructuras de datos para representar un grafo (árbol)
  - Listas de adyacencia
- Definir un algoritmo de alcanzabilidad, en base a una búsqueda en profundidad
  - Se ejecuta en tiempo lineal, en el número de aristas de $G_{\Sigma,\varphi}$
- Se logra una implementación que puede decidir el problema de implicación $\Sigma \models \varphi$ en tiempo $\mathcal{O}(|\varphi| \times (\sum_{\sigma_i \in \Sigma} |\sigma_i| \times |\varphi|))$

Implicación de Claves XML

Implementación (cont.)



(a) Witness-graph $G_{\Sigma,\varphi}$



(b) Lista de adyacencia

Figure: Representación mediante listas de adyacencia.

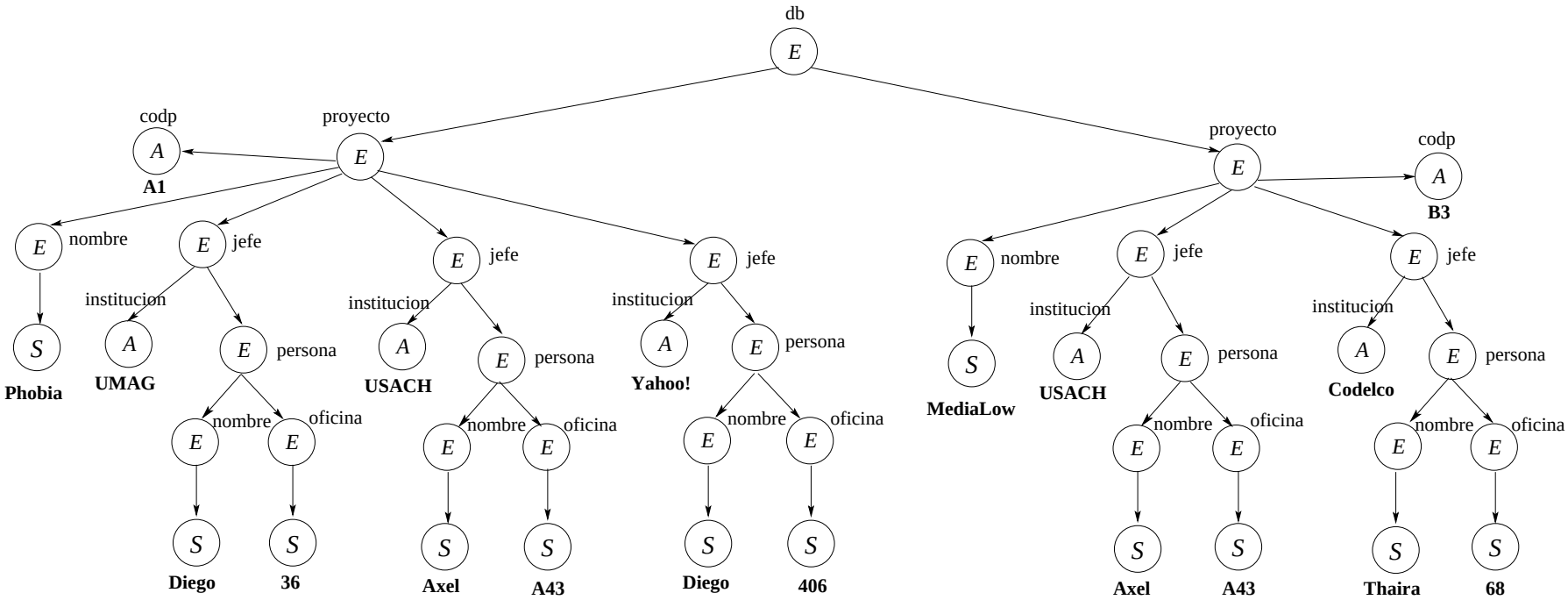
Validación de documentos contra claves XML

Estrategia

- Algoritmos rápidos para la validación de claves son cruciales para asegurar la consistencia de los datos en el intercambio [Arenas and Libkin, 2008]
- La estrategia es traducir el problema de validación al problema de satisfacción de claves en un árbol XML
- Se busca la primera intersección en valor no vacía
- En particular, nuestro algoritmo funciona con igualdad en valor no limitada a texto
- Proponemos transformar las claves en Σ en expresiones XPath, las que se aplican sobre un árbol XML (DOM)
- Útil para verificar la optimización con *covers* no-redundantes

Validación de documentos contra claves XML

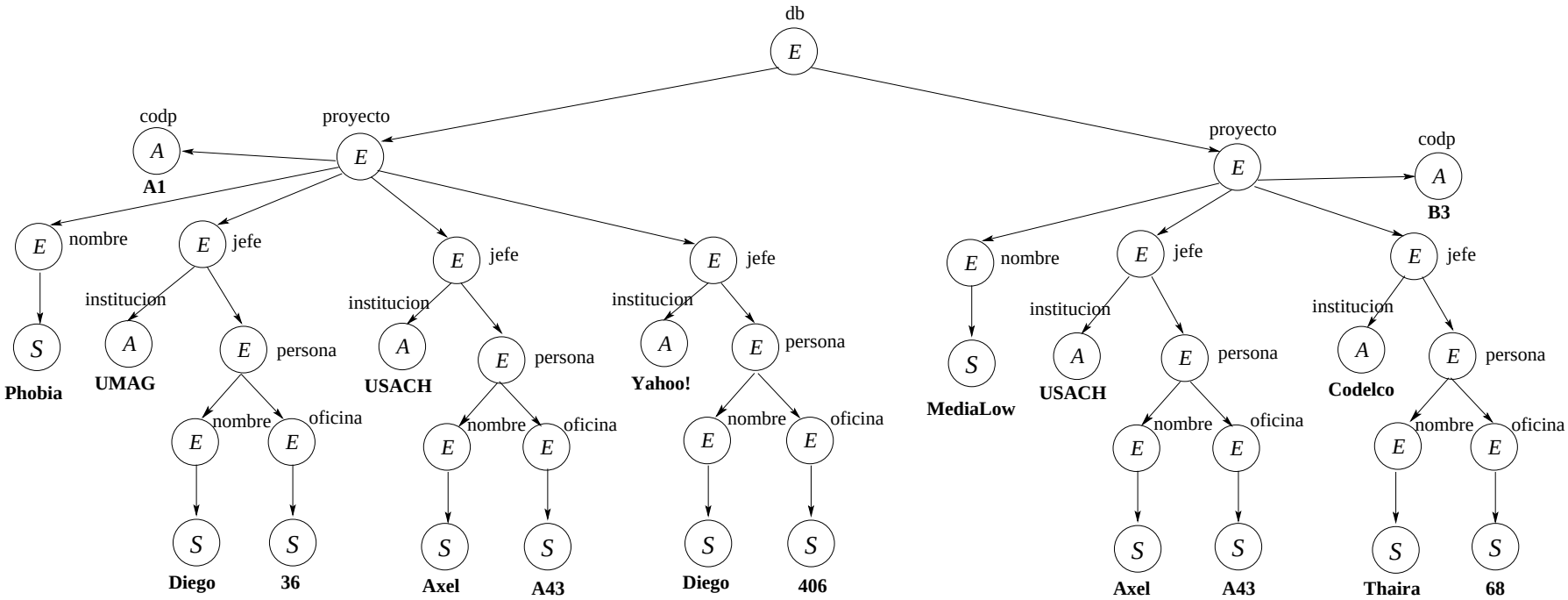
Algoritmo de Validación de documentos XML



La clave $(\varepsilon, (proyecto, \{codp\}))$ es satisfecha por el árbol XML T

Validación de documentos contra claves XML

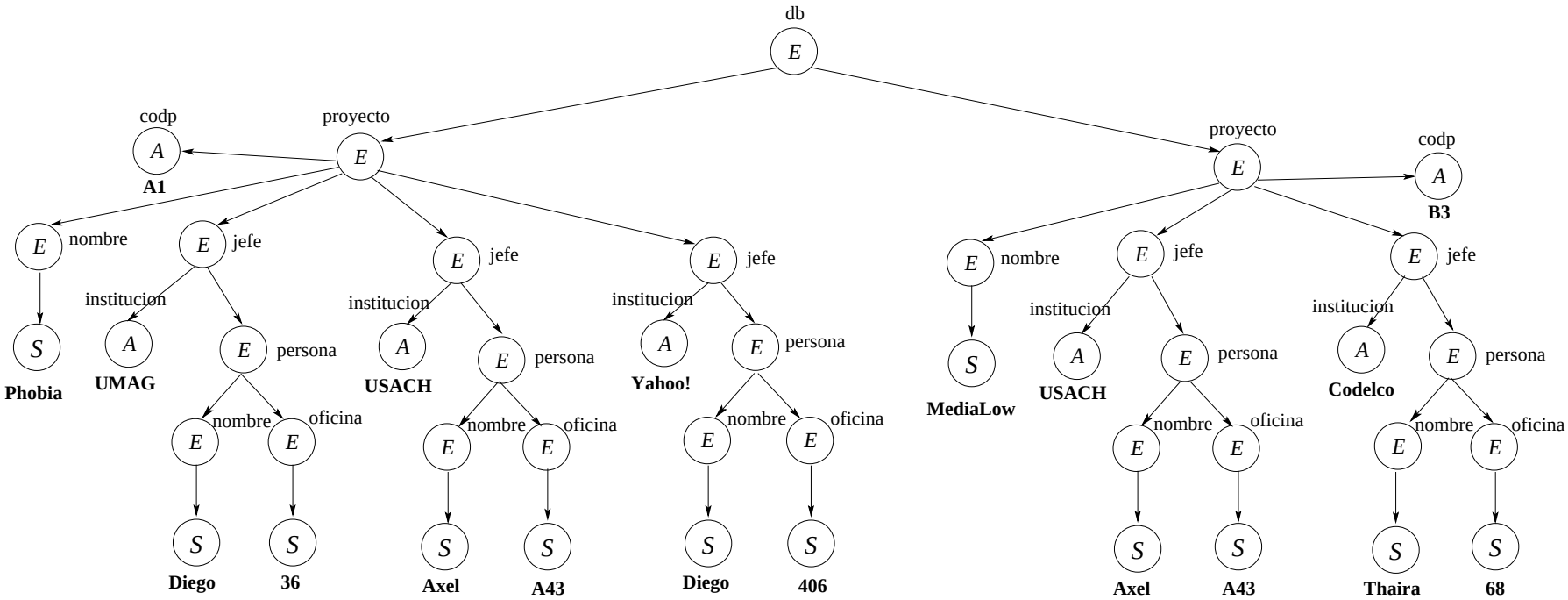
Algoritmo de Validación de documentos XML (cont.)



La clave $(\varepsilon, (\_^*.persona, \{nombre.S, oficina.S\}))$ no es satisfecha por el árbol XML T

Validación de documentos contra claves XML

Algoritmo de Validación de documentos XML (cont.)



La clave $(\varepsilon, (proyecto.\_^*.persona, \{nombre.S, oficina.S\}))$ no es satisfecha por el árbol XML T

Validación de documentos contra claves XML

Implementación

- Usando Java SDK y la biblioteca `javax.xml.parsers.*`
- Se obtiene una estructura utilizando la biblioteca `org.w3c.dom`
- Para las consultas XPath se utiliza la biblioteca `javax.xml.xpath.*`

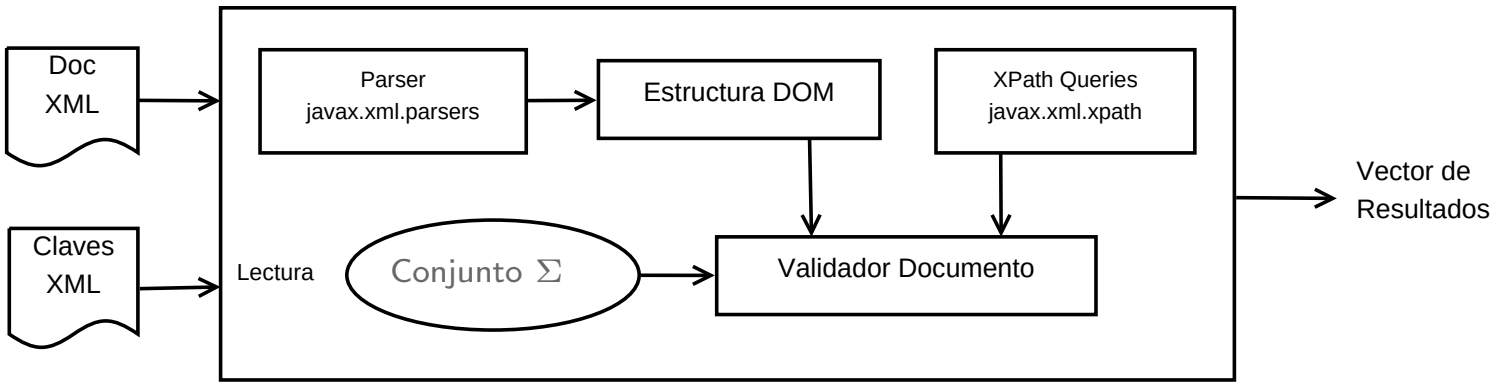


Figure: Arquitectura del validador.

Conjuntos Cover no-redundantes

Definiciones

Definición (Equivalencia)

Dos conjuntos Σ_1 y Σ_2 de claves XML son equivalentes , denotado $\Sigma_1 \equiv \Sigma_2$, si $\Sigma_1^* = \Sigma_2^* \Rightarrow$ ambos son cover del otro.

Definición (No-redundante)

Un conjunto Σ_2 de claves XML es no-redundante si no hay un subconjunto equivalente. Σ_2 es un cover no-redundante para un conjunto Σ_1, si Σ_2 es no-redundante y cover para Σ_1.

Conjuntos Cover no-redundantes

Caracterización

¿Podemos utilizar la implicación de claves?

- Σ es no-redundante si no hay una clave ψ en Σ tal que $\Sigma - \{\psi\} \models \psi$
- Si Σ_1 es redundante, entonces existe una clave ψ que puede ser descartada, obteniendo $\Sigma_2 = \Sigma_1 - \{\psi\}$
- Si Σ_2 es redundante, entonces existe una clave ϕ que puede ser descartada, obteniendo $\Sigma_3 = \Sigma_2 - \{\phi\}$
- etc...
- Nótese, que $\Sigma_3^* = \Sigma_2^* = \Sigma_1^*$

Conjuntos Cover no-redundantes

Caracterización

¿Podemos utilizar la implicación de claves?

- Σ es no-redundante si no hay una clave ψ en Σ tal que $\Sigma - \{\psi\} \models \psi$
- Si Σ_1 es redundante, entonces existe una clave ψ que puede ser descartada, obteniendo $\Sigma_2 = \Sigma_1 - \{\psi\}$
- Si Σ_2 es redundante, entonces existe una clave ϕ que puede ser descartada, obteniendo $\Sigma_3 = \Sigma_2 - \{\phi\}$
- etc...
- Nótese, que $\Sigma_3^* = \Sigma_2^* = \Sigma_1^*$

Conjuntos Cover no-redundantes

Algoritmo cálculo de Covers no-redundantes

Algoritmo 2: Cálculo de covers no-redundantes

Entrada: Conjunto finito Σ de claves XML en $\mathcal{K}(PL, PL, PL_s^+)$
Salida : Un cover no-redundante para Σ

1 $\Theta = \Sigma$;
2 **foreach** *clave* $\psi \in \Sigma$ **do**
3 **if** $\Theta - \{\psi\} \models \psi$ **then**
4 $\Theta = \Theta - \{\psi\}$;
5 **return** Θ;

Conjuntos Cover no-redundantes

Teorema definido

A partir de lo anterior, obtenemos el siguiente resultado:

Teorema

Un conjunto cover no-redundante para Σ puede ser calculado en tiempo $\mathcal{O}(|\Sigma| \times (max\{|\psi| : \psi \in \Sigma\})^2)$

Conjuntos Cover no-redundantes

Implementación

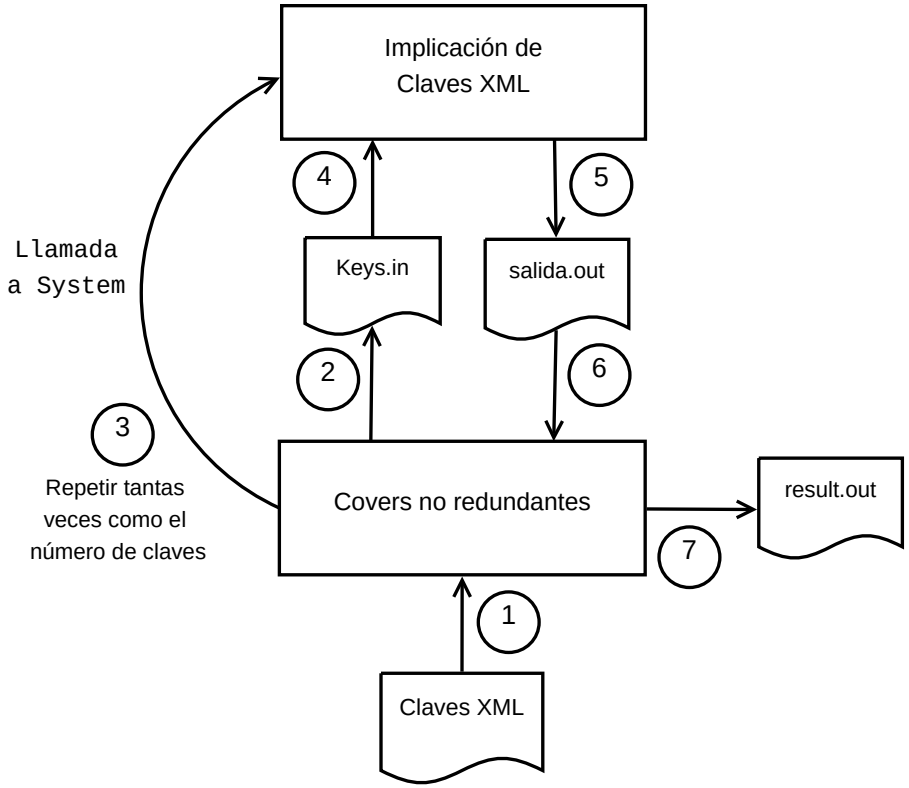
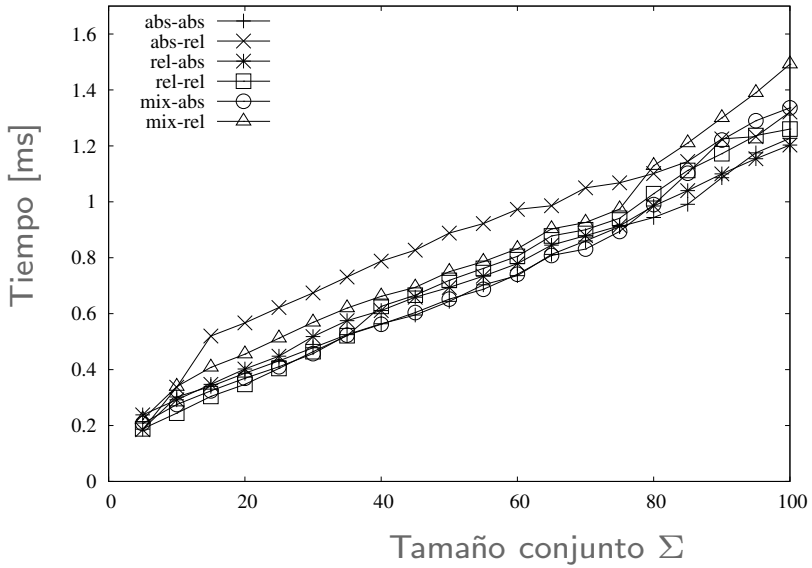


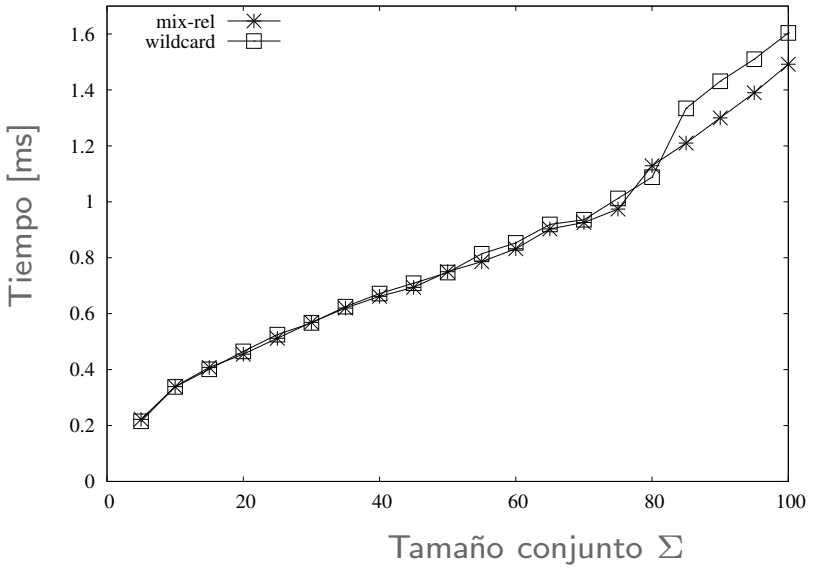
Figure: Flujo del cálculo de *covers* no-redundantes.

Resultados Experimentales

Experimentos de Implicación



(a) Casos de implicación de claves



(b) Efecto de los comodines

Figure: *Performance* de la implicación de claves XML.

Resultados Experimentales

Experimentos de Cover no-redundantes

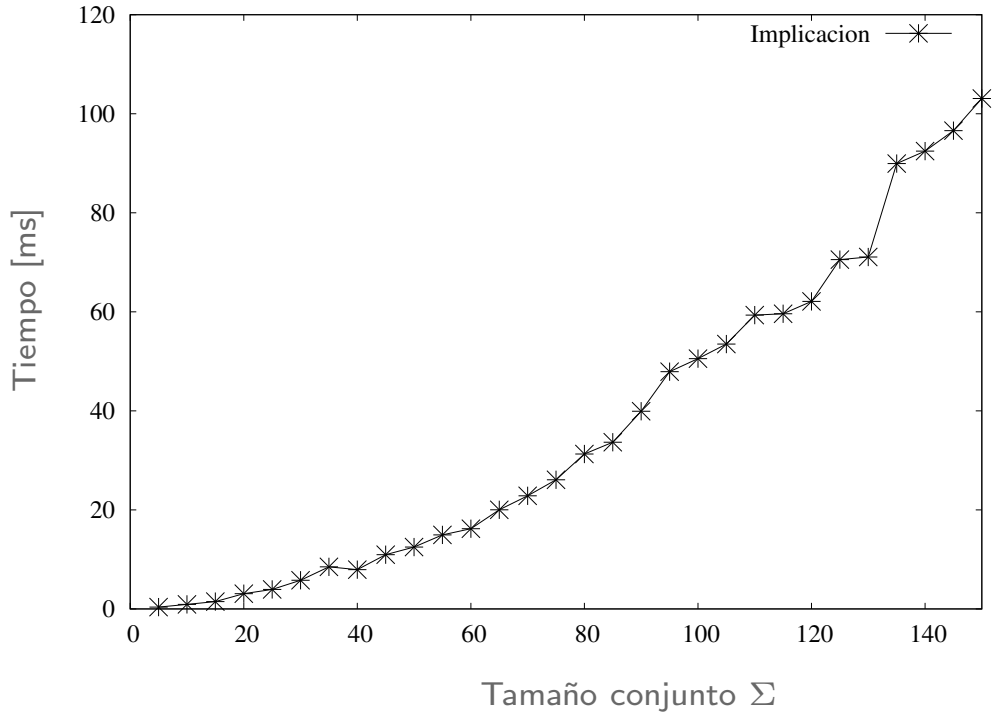
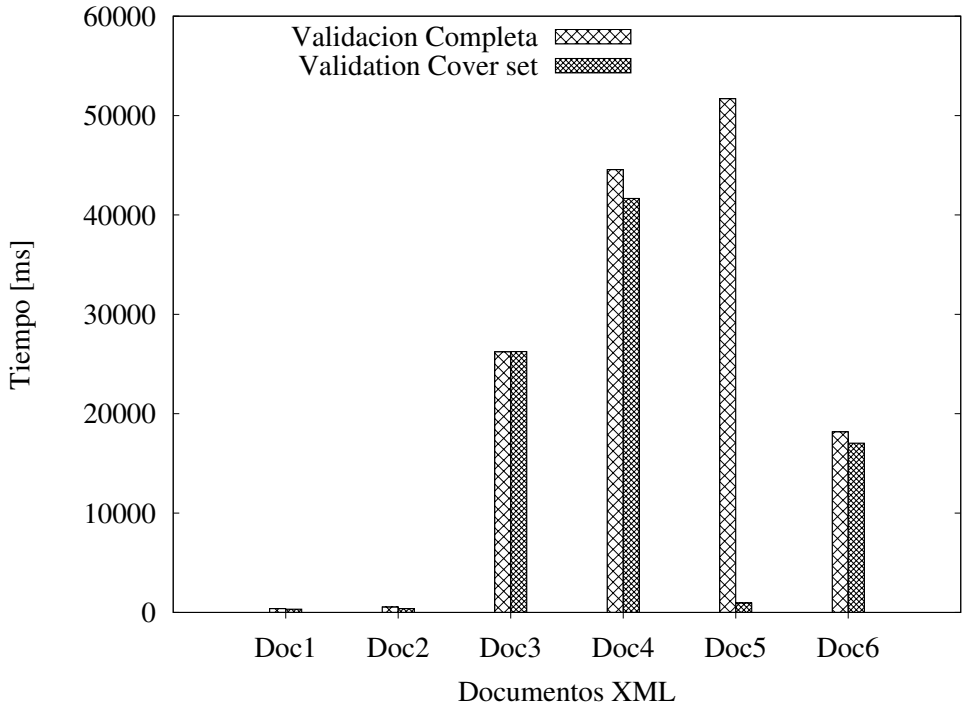


Figure: *Performance* del cálculo de covers no-redundante.

Resultados Experimentales

Validación contra Covers no-redundantes



Documentos XML

- 321gone
- Yahoo
- DBLP
- Nasa
- SIGMOD Record
- Mondial

Figure: *Performance* de la validación de documentos.

Conclusiones y Trabajos Futuros I

- Este trabajo tuvo como objetivo principal demostrar que existen clases expresivas, no sólo tratables en la teoría, sino también en la práctica
- Estudiamos un fragmento significativo de claves XML
- Su algoritmo de implicación cuadrático en teoría, es rápido y escala bien en la implementación
- Estudiamos las consecuencias de decidir eficientemente la implicación de claves en la gestión de bases de datos XML
- Implementamos y evaluamos el algoritmo de implicación de claves en $\mathcal{K}(PL, PL, PL_s^+)$
- Diseñamos, implementamos y evaluamos un algoritmo para validación de documentos, que respete la igualdad valor definida por [Buneman et al., 2002, Buneman et al., 2003]

Conclusiones y Trabajos Futuros II

- Diseñamos, implementamos y evaluamos un método de optimización del proceso de validación de documentos XML, en base a conjuntos cover no redundantes
- Este método puede reducir significativamente el número de claves, traduciéndose en una enorme reducción en el tiempo de validación
- El tiempo de cálculo de un cover es una fracción del tiempo de validación contra una sola clave
- El algoritmo para calcular covers no depende del fragmento aquí estudiado
- Planeamos extender nuestros estudios para otros fragmentos expresivos de claves XML; y estudiar el uso de estructuras compactas para reducir el espacio de memoria

Agradecimientos

- A la Universidad de Santiago de Chile
- A la Fundación San Marcelino Champagnat
- Al laboratorio Yahoo! Research Latin America
- A mis profesores guías Flavio Ferrarotti y Mauricio Marín
- A todos mis amigos y compañeros

Referencias I

Arenas, M., Fan, W., and Libkin, L. (2002).
What's Hard about XML Schema Constraints?
In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, DEXA '02, pages 269–278, London, UK, UK. Springer-Verlag.

Arenas, M. and Libkin, L. (2008).
XML data exchange: Consistency and query answering.
J. ACM, 55:7:1–7:72.

Buneman, P., Davidson, S. B., Fan, W., Hara, C. S., and Tan, W. C. (2002).
Keys for XML.
Computer Networks, 39(5):473–487.

Buneman, P., Davidson, S. B., Fan, W., Hara, C. S., and Tan, W. C. (2003).
Reasoning about keys for XML.
Inf. Syst., 28(8):1037–1063.

Fan, W. (2005).
XML Constraints: Specification, Analysis, and Applications.
In *DEXA Workshops*, pages 805–809. IEEE Computer Society.

Fan, W. and Libkin, L. (2002).
On XML integrity constraints in the presence of DTDs.
J. ACM, 49(3):368–406.

Fan, W. and Siméon, J. (2003).
Integrity constraints for XML.
J. Comput. Syst. Sci., 66(1):254–291.

Referencias II

Hartmann, S. and Link, S. (2009).
Efficient Reasoning about a Robust XML Key Fragment.
ACM Trans. Database Syst., 34(2).

Suciu, D. (2001).
On database theory and XML.
SIGMOD Rec., 30(3):39–45.

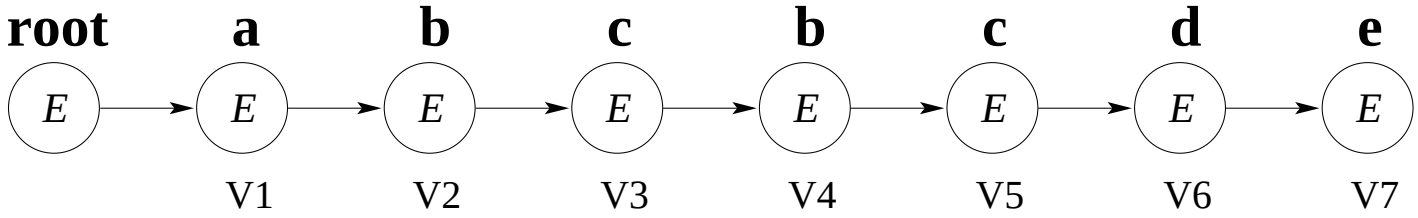
Vianu, V. (2003).
A Web odyssey: from codd to XML.
volume 32, pages 68–77.

Consultas

GRACIAS POR SU ATENCIÓN

Emir F. Muñoz Jiménez
emir.munozj@usach.cl

Regla subnodos



- Se tiene que T satisface la clave absoluta $\sigma = (\varepsilon, (a.\_^*.b.c.\_^*.d, \{e\}))$, pero no satisface la clave absoluta $\varphi = (\varepsilon, (a.\_^*.b, \{c.\_^*.d.e\}))$
- Ya que $v_2, v_4 \in [\![a.\_^*.b]\!], v_2 \neq v_4$ y $v_2[\![c.\_^*.d.e]\!] \cap_v v_4[\![c.\_^*.d.e]\!] = \{(v_7, v_7)\}$
- Esto es, φ no es implicada por σ
- Sin embargo, φ puede ser inferida a partir de σ usando la regla subnodos definida en [Buneman et al., 2003]

- Nuestra implementación del algoritmo de implicación, puede ser utilizada en el contexto de mining de claves XML
- No considerando las claves trivialmente satisfechas
- Si una clave φ es extraída desde un conjunto de claves Σ
- Y se tiene que un conjunto D de documentos, satisface todas las claves en el conjunto Σ
- Si $\Sigma \models \varphi$, entonces φ es satisfecha por todos los $d_i \in D$
- Sin necesidad de validar cada documento contra la clave φ descubierta