

XML Query Optimisation: Specify your Selectivity

Sven Hartmann, Sebastian Link

This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

Approaches to XML Query Optimisation

- ▶ Numbering schemes: // -relationship, updates vs. efficient querying
- ▶ Indices: name/value index, dataguide, template, $A(k)$, for/backward
- ▶ Formal semantics: mapping into functional core enjoys benefits from functional programming and other database languages
- ▶ XQuery algebras: XAL, tree algebra for XML, tree patterns
- ▶ XPath and Core XPath: combined complexity **P**-complete
↳ for positive Core XPath **LOGCFL**-complete
- ▶ Selectivity estimation: crucial for choice of query execution plan
↳ simple navigation, structural joins, twig joins etc.

Schema-based Optimisation

- ▶ utilizing additional information in schemata to simplify queries
- ▶ query: `//Student[Birthday]/Address[phone|email]`
- ▶ DTD fragment:
 - `<!ELEMENT Students(Student*)>`
 - `<!ELEMENT Student(Name,Address,Birthday)>`
 - `<!ELEMENT Address(Street,City,Zip,(phone|email))>`
- ▶ simplified query: `//Student/Address`

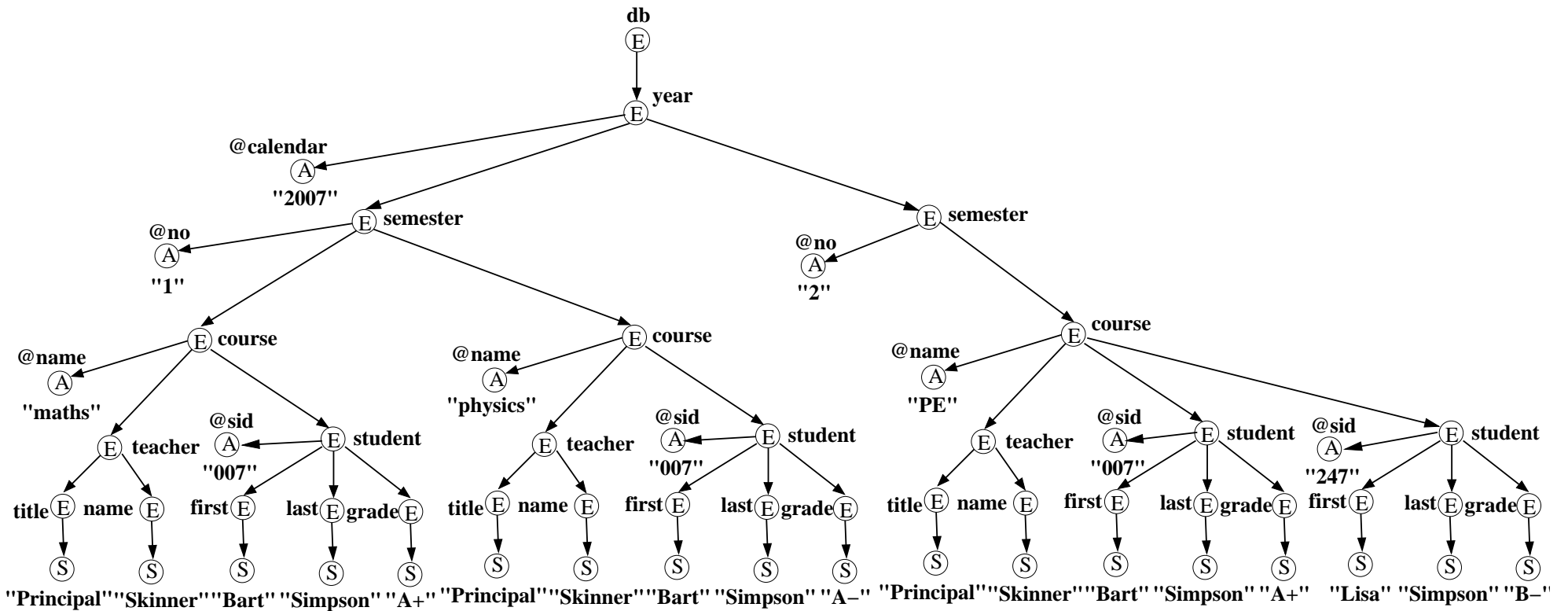
Our Idea: Constraint-based Optimisation

- ▶ we utilize information encoded in constraints
- ▶ show benefits of cardinality constraints for XML data processing
 - ↳ query optimisation by selectivity specification
 - ↳ predicting no of query answers, updates, en/decryptions
 - ↳ schema transformation for efficient querying and updating
 - ↳ advanced policies for consistent query answering
- ▶ formalize cardinality constraints
- ▶ unlock applications effectively by capturing implicitly-defined constraints efficiently
- ▶ investigate tradeoffs between expressiveness and tractability

An XML fragment

```
<db>
  <year calendar=2007>
    <semester no=1>
      <course name=maths>
        <teacher> <title>Principal</title> <name>Skinner</name> </teacher>
        <student sid=007> <first>Bart</first> <last>Simpson</last> <grade>A+</grade> </student>
      </course>
      <course name=physics>
        <teacher> <title>Principal</title> <name>Skinner</name> </teacher>
        <student sid=007> <first>Bart</first> <last>Simpson</last> <grade>A-</grade> </student>
      </course>
    </semester>
    <semester no=2>
      <course name=PE>
        <teacher> <title>Principal</title> <name>Skinner</name> </teacher>
        <student sid=007> <first>Bart</first> <last>Simpson</last> <grade>A+</grade> </student>
        <student sid=247> <first>Lisa</first> <last>Simpson</last> <grade>B-</grade> </student>
      </course>
    </semester>
  </year>
</db>
```

Its corresponding XML Tree



Some Constraints

- relatively to each year: a semester is determined by its number

$$\text{card}(\text{year}, (\text{semester}, \{\text{no}\})) = (1, 1)$$

- each year node has either no semester nodes or precisely two

$$\text{card}(\text{year}, (\text{semester}, \emptyset)) = (2, 2)$$

- relatively to each course: a student is determined by its sid

$$\text{card}(.//\text{course}, (\text{student}, \{\text{sid}\})) = (1, 1)$$

- each student deciding to study in a semester enrolls into at least two and at most 4 courses in that semester

$$\text{card}(.//\text{semester}, (\text{course}, \{.//\text{sid}\})) = (2, 4)$$

- no student enrolls more than three times into the same course

$$\text{card}(. , (.//\text{course}, \{\text{name}, .//\text{sid}\})) = (1, 3)$$

Scenario 1: Approximating Query Costs

- ▶ *Bart Simpson* wishes to query XML source for results in 2007
 - ↳ using his cell-phone but only when costs reasonable
 - ↳ service provider: only retrieve data if service paid for
- ▶ consider the following XQuery query:
for \$s **in** doc(“enrol.xml”)/year[@calendar=”2007”]//course/student
where \$s/@sid=”007”
return <grade>{\$s/grade}</grade>
- ▶ XML DBMSs capable of reasoning about numerical constraints foresees maximal number of eight answers (without processing any data!)
- ▶ approximate costs returned to *Bart Simpson* who decides accordingly
- ▶ service provider minimizes costs for unpaid services

Scenario 2: Query Optimisation

- ▶ teachers teach at most 3 courses per year
- ▶ students take up to 4 courses per semester, with two semester per year
- ▶ XML query optimiser should transform the XQuery query

```
for $c in doc("enrol.xml")/year[@calendar="2007"]/semester/course
where $c/student/@sid="247" and $c/teacher="Principal Skinner"
return <course>{$c/@name}</course>
```

- ▶ into

```
for $c in doc("enrol.xml")/year[@calendar="2007"]/semester/course
where $c/teacher="Principal Skinner" and $c/student/@sid="247"
return <course>{$c/@name}</course>
```

- ▶ based on smaller selectivity

Some more Query Optimisation

- ▶ retrieve *@name*-child of those 2007-courses which only feature students that participated in at most 10 different courses in 2007:

for \$c **in**

doc(“courses.xml”)/year[@calendar=”2007”]//course

where every \$s **in** \$c/student/@sid **satisfies**

count(doc(“courses.xml”)/year[@calendar=”2007”]//
course[student/@sid=\$s]) ≤ 10

return <c_name>{\$c/@name}</c_name>

- ▶ based on the constraints above this simplifies to

for \$c **in**

doc(“courses.xml”)/year[@calendar=”2007”]//course

return <c_name>{\$c/@name}</c_name>

Scenario 3: Predicting the No of Updates

- ▶ databases often subject to updates
- ▶ following XQuery updates last name of *Lisa Simpson* to *Milhouse* in all semester 2 courses of 2007:

```
for $s in doc("enrol.xml")/year[@calendar="2007"]/  
    semester[@no="2"]//student[@sid="247"]  
return do replace value of $s/last with "Milhouse"
```

- ▶ maximal number of four updates (decide implication efficiently)

Scenario 4: An encrypted XML fragment

```
<db>
  <year calendar=2007>
    <semester no=1>
      <course name=maths>
        <teacher> <title>Principal</title> <name>Skinner</name> </teacher>
        <student sid=007> <first>Bart</first> <last>Simpson</last>
        <grade>
          <xenc:EncryptedData>
            <xenc:CipherData>
              <xenc:CipherValue> AbC234ndZ... </xenc:CipherValue>
            </xenc:CipherData>
          </xenc:EncryptedData>
        </grade>
      </student>
    </course>
  </semester>
</year>
</db>
```

Scenario 4: Predicting the No of Encryptions

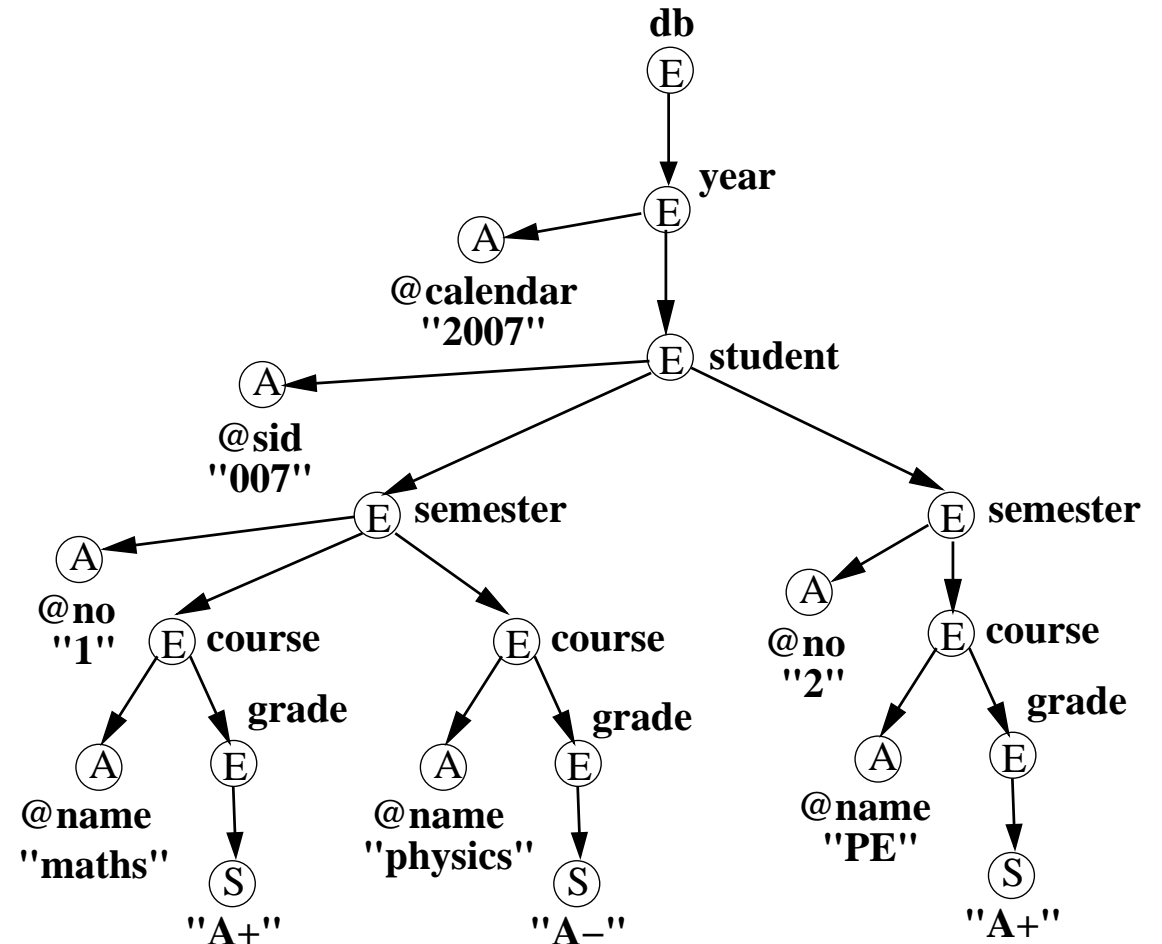
- ▶ Web data often encrypted (grades of students)
- ▶ following XQuery query retrieves grades of *Bart Simpson* in 2007:

```
for $s in doc("enrol.xml")/year[@calendar="2007"]//course/student
where $s/@sid="007"
return <grade>{$s/grade}</grade>
```

- ▶ efficient reasoning about constraints enables DBMS to infer that at most eight decryptions necessary

Scenario 5: XML Transformations

- ▶ *year*-nodes subsume between two and eight *course*-nodes containing *student/@sid*-subnodes with same value
- ▶ querying original XML tree for course info based on specific year and specific sid unnatural
- ▶ *sid*-updates difficult
- ▶ create XML view



Scenario 5 continued: Query Rewriting

▶ rewrite

```
for $s in doc("enrol.xml")/year[@calendar="2007"]//course/student
where $s/@sid="007"
return <grade>{$s/grade}</grade>
```

▶ as

```
for $c in doc("view.xml")/year[@calendar="2007"]/
    student[@sid="007"]/course
return <grade>{$c/grade}</grade>
```

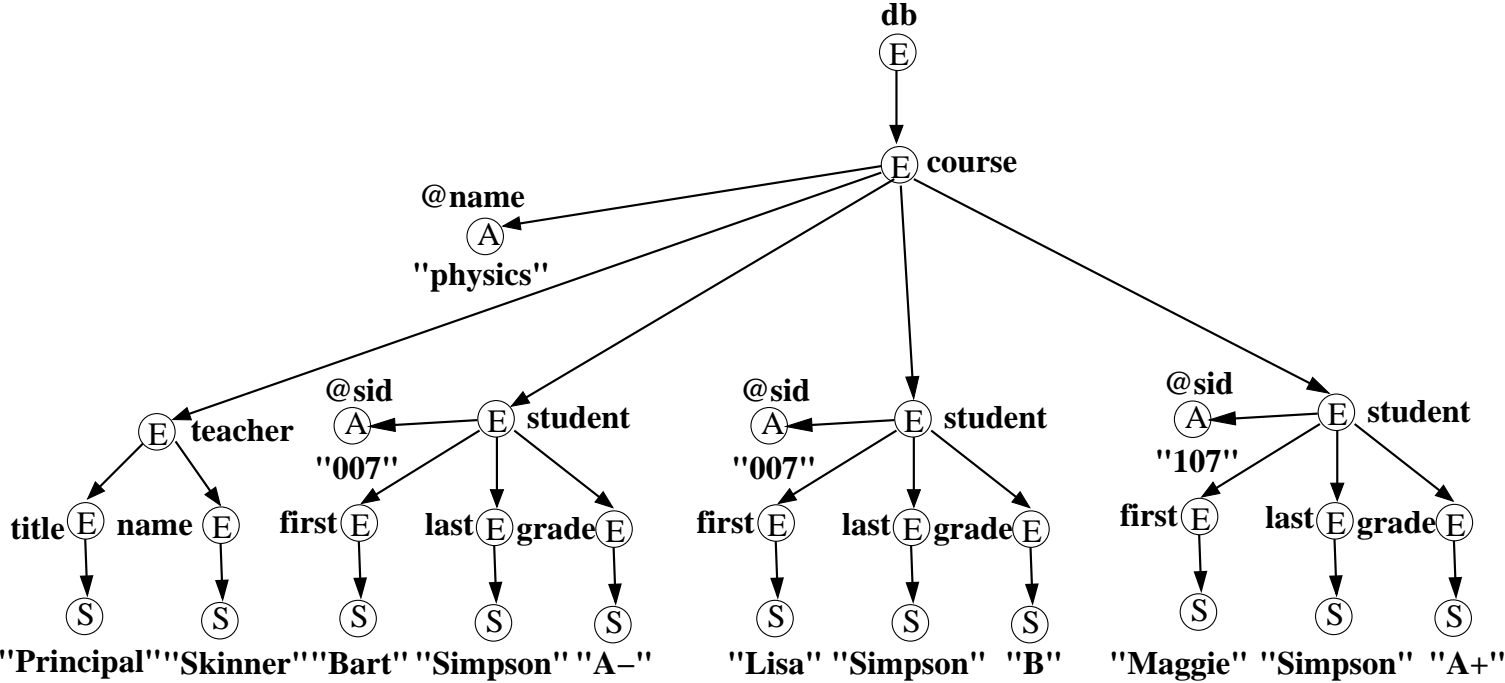
▶ early selection of *student*-elements based on their *sid*

▶ even more efficient in case @*sid*-values are encrypted

▶ easy updates of *sid*-values

Scenario 6: Consistent Query Answering

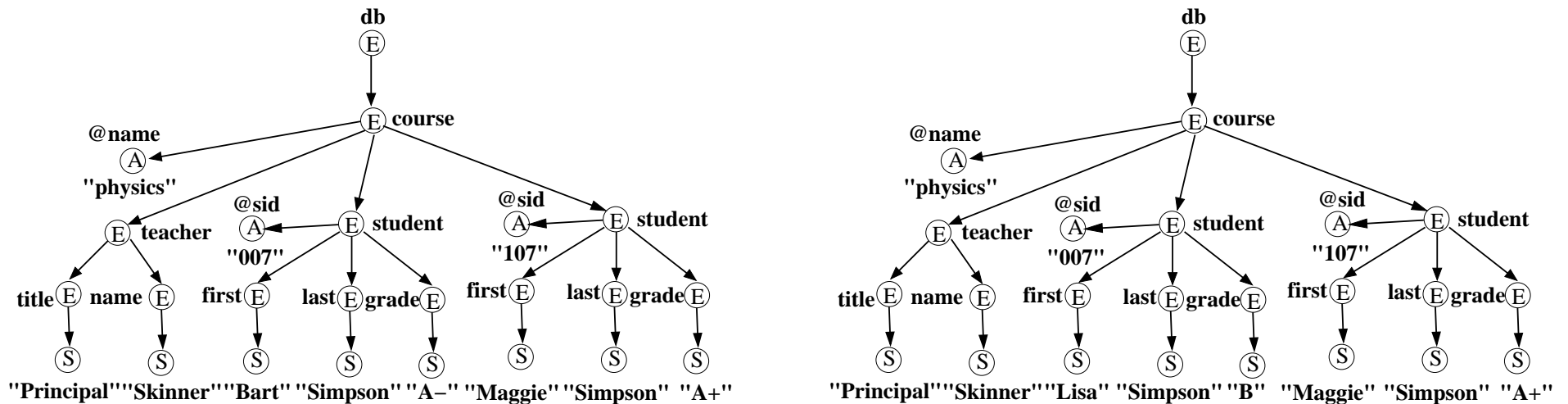
- ▶ approach to querying inconsistent databases without repairing them
- ▶ retrieves only certain answers, those present in *all* repairs



- ▶ *Bart, Lisa, Maggie*: //course[@name = 'physics']/student/first
- ▶ however:
there mustn't be different *student*-nodes with same @sid in any course

CQA: One Solution

- ▶ repair may refer to removal of any offending *student*-node:

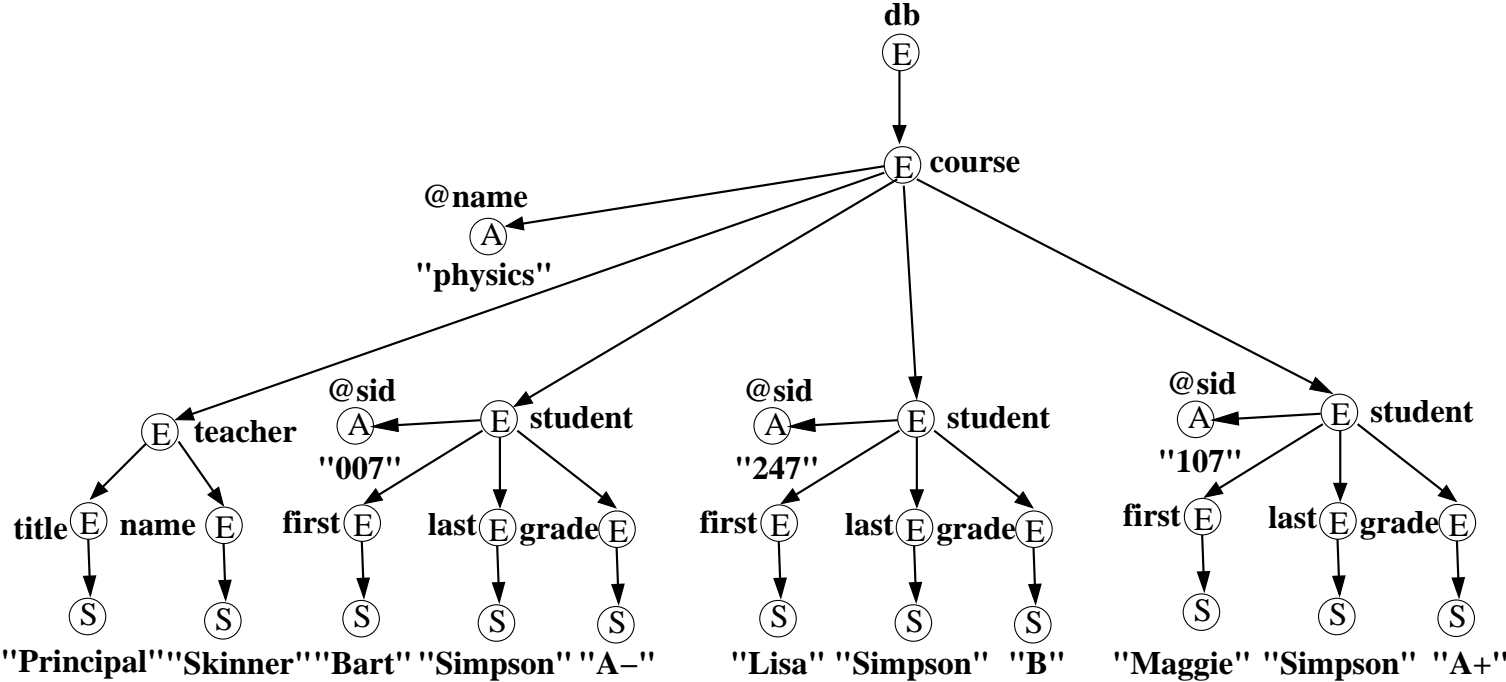


- ▶ CQA returns only *Maggie* as a certain answer:

`//course[@name = 'physics']/student/first`

CQA: A different Policy

- ▶ repair may refer to replacement of any offending @sid-value by a new @sid-value not present in the tree; infinitely many repairs, e.g.:



- ▶ *Bart*, *Lisa* and *Maggie* present in all of them:

```
//course[@name = 'physics']/student/first
```

Numerical constraints for XML

- a *numerical constraint* φ has the form

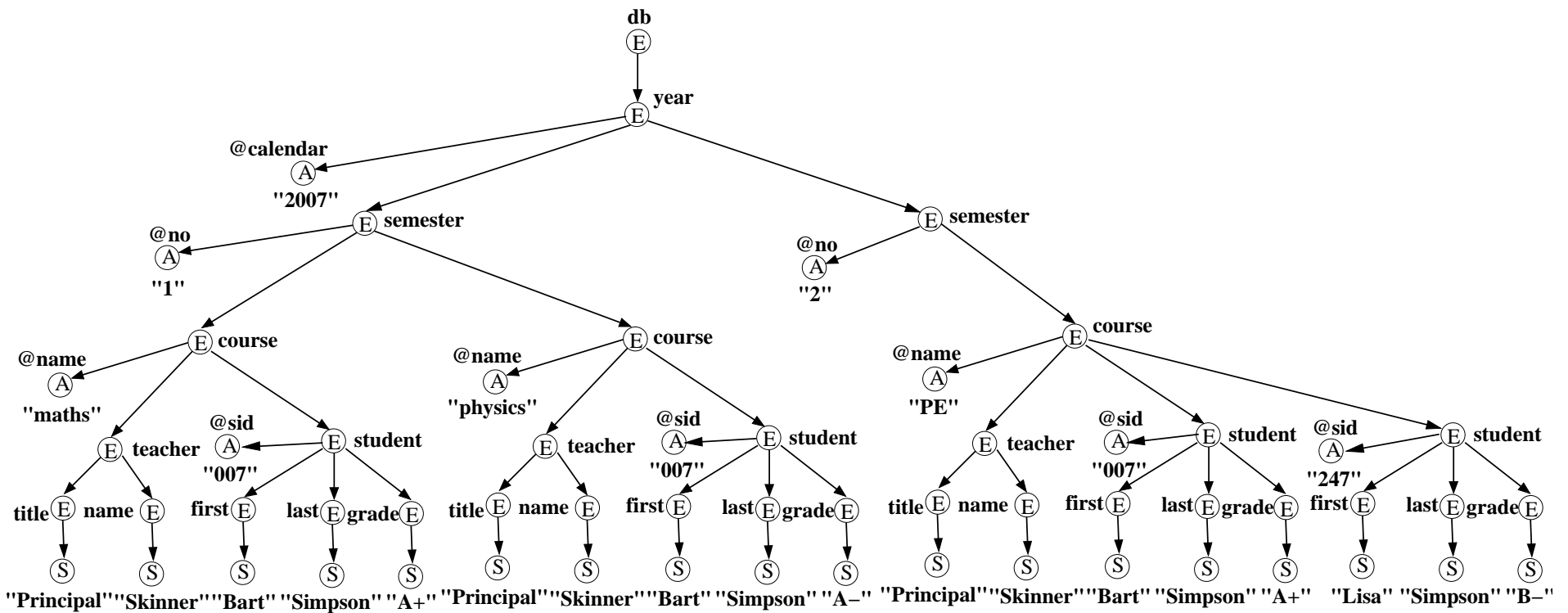
$$\text{card}(Q, (Q', \{Q_1, \dots, Q_k\})) = (\text{min}, \text{max})$$

- $Q \in XP(/, //)$ is the *context path*
- $Q' \in XP(/, //)$ is the *target path*
- $Q_i \in XP(/, //)$ is a *key path* (for $i = 1, \dots, k$)
- $Q.Q'$ valid ($k = 0$) or $Q.Q'.Q_i$ valid path (for $i = 1, \dots, k$)
- two nodes v_1, v_2 are $\{Q_1, \dots, Q_k\}$ -*confusable* iff for all $i = 1, \dots, k$ there are value-equal nodes reachable from v_1 and v_2 via Q_i
- the XML tree T *satisfies* φ iff for all nodes $q \in r[[Q]]$ and all nodes $q' \in q[[Q']]$ there are no less than *min* and no more than *max* nodes in $q[[Q']]$ that are $\{Q_1, \dots, Q_k\}$ -confusable with q'

Example 1

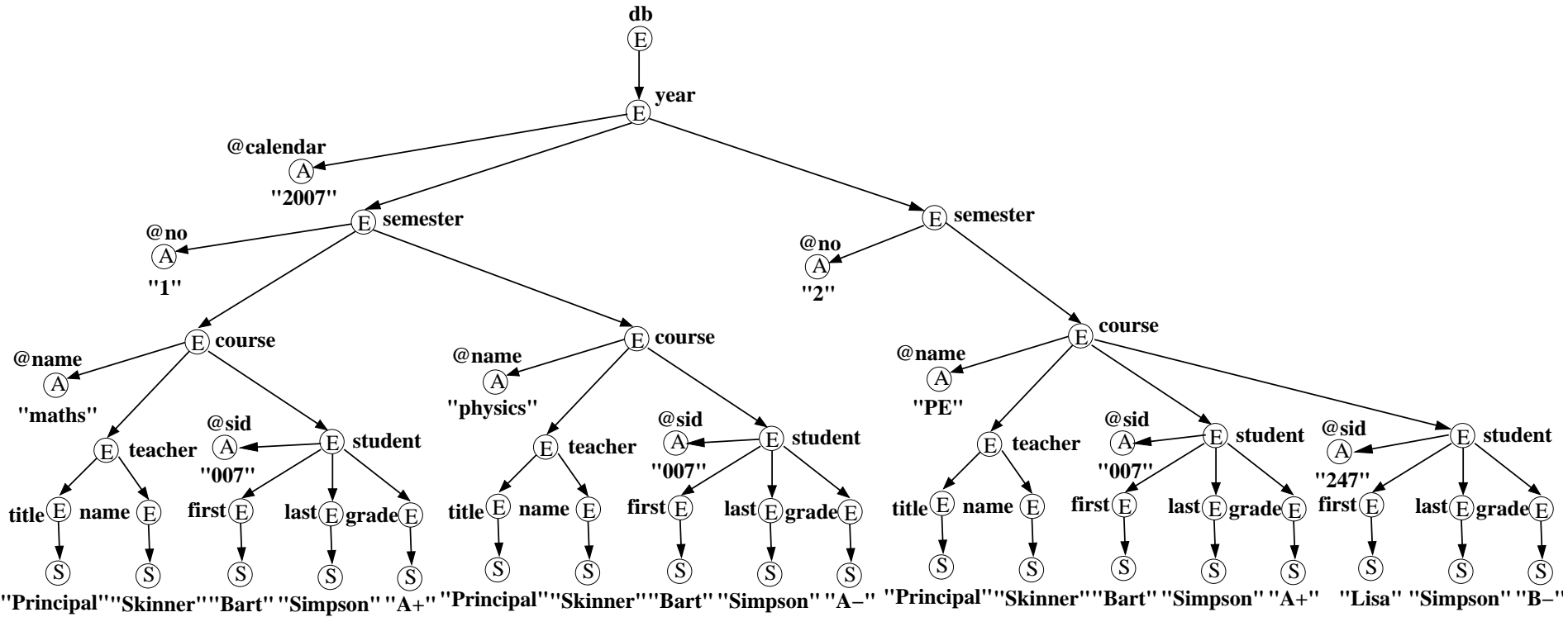
- ▶ in every semester every student who studies must enrol in 2 to 4 courses

T does not satisfy $card(.//semester,(course,\{.//sid\}))=(2,4)$



Example 2

- ▶ T satisfies $card(year, (.//course, \{teacher\})) = (3, 6)$
- ▶ T violates $card(.//semester, (course, \{teacher\})) = (2, 2)$



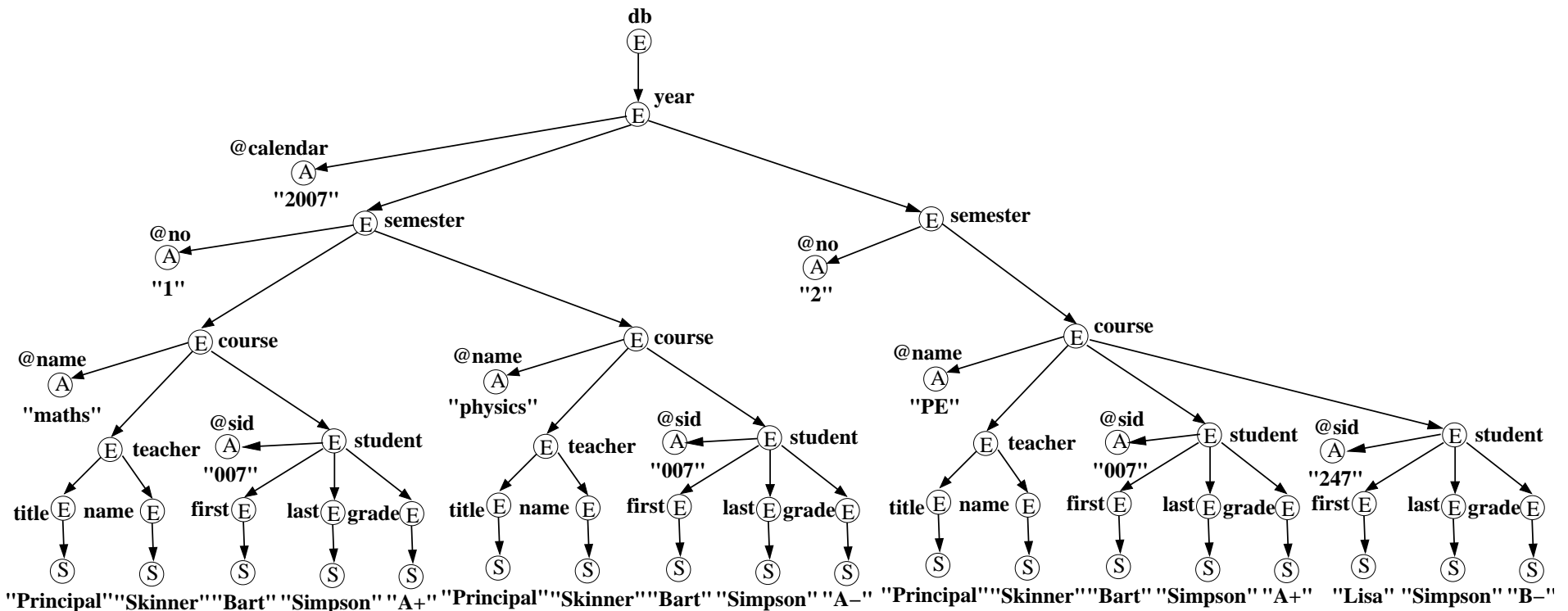
Example 3

► XML keys covered by numerical constraints: $min = max = 1$

↳ P. Buneman et al. Reasoning about keys for XML. Inf. Syst.28(8):1037–1063, 2003

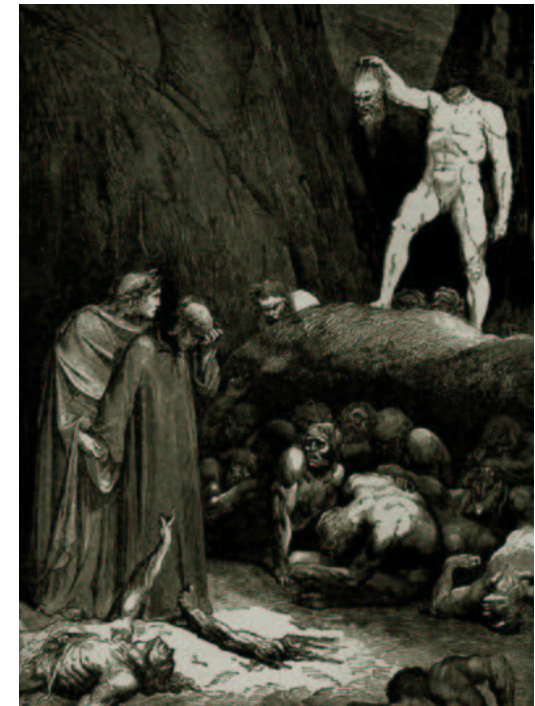
↳ S. Hartmann, S. Link. Unlocking keys for XML trees. ICDT, LNCS 4353, pp. 104-118, 2007

► T satisfies $card(.//course,(student,\{sid\}))=(1,1)$



A Computer Scientist's Pilgrimage

- ▶ Mark Musa, in his translation of Dante's *Divine Comedy*:
Dante the Pilgrim, in order to arrive at the Divine Light, will come to an understanding of sin on his journey through Hell and will see the penance imposed on repentant sinners on the Mount of Purgatory.
- ▶ our journey through Hell (it's not that bad!):
 - ↳ our sin: too much expressiveness
 - ↳ understanding: identifying intractabilities
- ▶ our Mount Purgatory:
 - ↳ restriction to tractable cases
 - ↳ do penance by studying properties
- ▶ our "Divine Light":
 - ↳ finite axiomatization and efficient algorithms



Sources of Intractability

▶ 3 fragments:

$$\hookrightarrow \mathcal{F}_1 = \{\text{card}(\cdot, (P', \{P_1, \dots, P_k\})) = (\min, \max) \mid k \geq 1\}$$

$$\hookrightarrow \mathcal{F}_2 = \{\text{card}(\cdot, (P', \{P_1, \dots, P_k\})) = (1, \max) \mid k \geq 0\}$$

$$\hookrightarrow \mathcal{F}_3 = \{\text{card}(\cdot, (Q', \{Q_1, \dots, Q_k\})) = (1, \max) \mid k \geq 1\}$$

▶ Theorem:

The finite implication problems for \mathcal{F}_1 , \mathcal{F}_2 and \mathcal{F}_3 are all *coNP*-hard.

▶ suggests that computational intractability may result from:

\hookrightarrow the specification of both lower and upper bounds

\hookrightarrow empty sets of key paths

\hookrightarrow arbitrary path expressions in both target- and key paths

Numerical keys for XML

- ▶ a numerical keys φ has the form

$$\text{card}(Q, (Q', \{P_1, \dots, P_k\})) \leq \text{max}$$

- $Q \in XP(/, //)$ is the *context path*
 - $Q' \in XP(/, //)$ is the *target path*
 - $P_i \in XP(/)$ is a *key path* (for $i = 1, \dots, k$ and $k \geq 1$)
 - $Q.Q'.P_i$ valid path (for $i = 1, \dots, k$)
- ▶ XML tree T satisfies $\text{card}(Q, (Q', \{P_1, \dots, P_k\})) \leq \text{max}$ iff T satisfies $\text{card}(Q, (Q', \{P_1, \dots, P_k\})) = (1, \text{max})$
- ▶ Theorem:
Every finite set of numerical keys is finitely satisfiable.
- ▶ Theorem:
Implication and finite implication coincide for numerical keys.

Reasoning about numerical keys

$$\frac{\text{card}(Q, (Q', S)) \leq \infty}{\text{(infinity)}}$$

$$\frac{\text{card}(Q, (\epsilon, S)) \leq 1}{\text{(epsilon)}}$$

$$\frac{\text{card}(Q, (Q', S)) \leq \max}{\text{card}(Q, (Q', S)) \leq \max + 1} \text{(weakening)}$$

$$\frac{\text{card}(Q, (Q', S)) \leq \max}{\text{card}(Q, (Q', S \cup \{P\})) \leq \max} \text{(superkey)}$$

$$\frac{\text{card}(Q, (Q'.P, \{P'\})) \leq \max}{\text{card}(Q, (Q', \{P.P'\})) \leq \max} \text{(subnodes)}$$

$$\frac{\text{card}(Q, (Q'.Q'', S)) \leq \max}{\text{card}(Q.Q', (Q'', S)) \leq \max} \text{(context target)}$$

$$\frac{\text{card}(Q, (Q', S)) \leq \max}{\text{card}(Q'', (Q', S)) \leq \max}^{Q'' \subseteq Q} \text{(context-path-containment)}$$

$$\frac{\text{card}(Q, (Q'.P, \{\epsilon, P'\})) \leq \max}{\text{card}(Q, (Q', \{\epsilon, P.P'\})) \leq \max} \text{(subnodes-epsilon)}$$

$$\frac{\text{card}(Q, (Q', S)) \leq \max}{\text{card}(Q, (Q'', S)) \leq \max}^{Q'' \subseteq Q'} \text{(target-path-containment)}$$

$$\frac{\text{card}(Q, (Q', S \cup \{\epsilon, P\})) \leq \max}{\text{card}(Q, (Q', S \cup \{\epsilon, P.P'\})) \leq \max} \text{(prefix-epsilon)}$$

$$\frac{\text{card}(Q, (Q', \{P.P_1, \dots, P.P_k\})) \leq \max, \text{card}(Q.Q', (P, \{P_1, \dots, P_k\})) \leq \max'}{\text{card}(Q, (Q'.P, \{P_1, \dots, P_k\})) \leq \max \cdot \max'} \text{(multiplication)}$$

► numerical key implication decidable in time $\mathcal{O}((\|\Sigma\| + |\varphi|) \cdot |\varphi|)$

Findings

- ▶ identified natural class of XML constraints that generalise XML keys
- ▶ useful for many XML applications:
 - ↳ XQuery, XPath, XML Encryption, XQuery update facility
 - ↳ cost estimation, query optimisation, query rewriting
- ▶ reasoning about cardinality constraints intractable in general
 - ↳ specification of both lower and upper bounds
 - ↳ empty set of key paths
 - ↳ general key path expressions
- ▶ identified numerical keys as large tractable subclass
 - ↳ always satisfiable
 - ↳ implication and finite implication coincide
 - ↳ established finite axiomatisation
 - ↳ characterised implication in terms of shortest paths
 - ↳ efficient solutions to decision problem unlocks applications