Declarative Inconsistency Handling in Relational and Semi-structured Databases

Ph.D. Dissertation Defense

Sławek Staworko

Department of Computer Science and Engineering
University at Buffalo, SUNY

9 May 2007

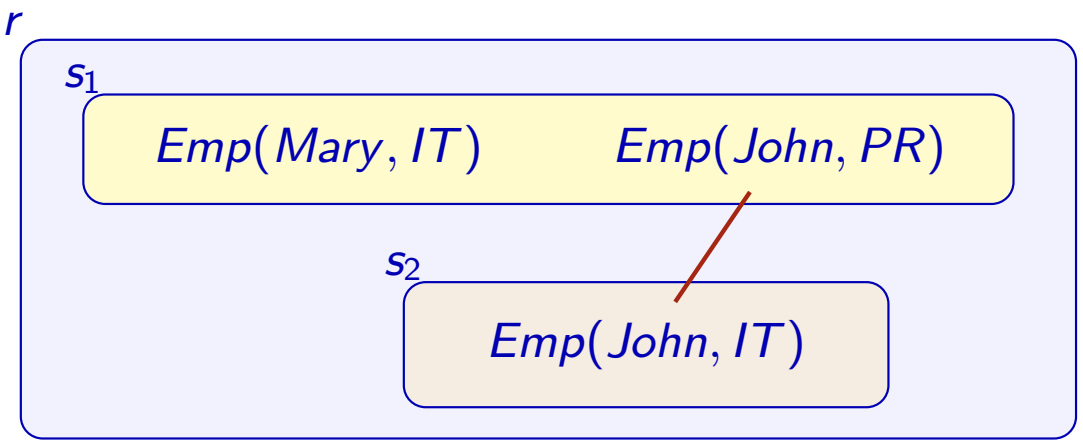
Motivation

Schema

$Emp(\underline{Name}, Dept)$

Q: who works in IT

Q(X) :- Emp(X, IT) .
Answers are: $\{Mary, John\}$
Are we sure about John ?



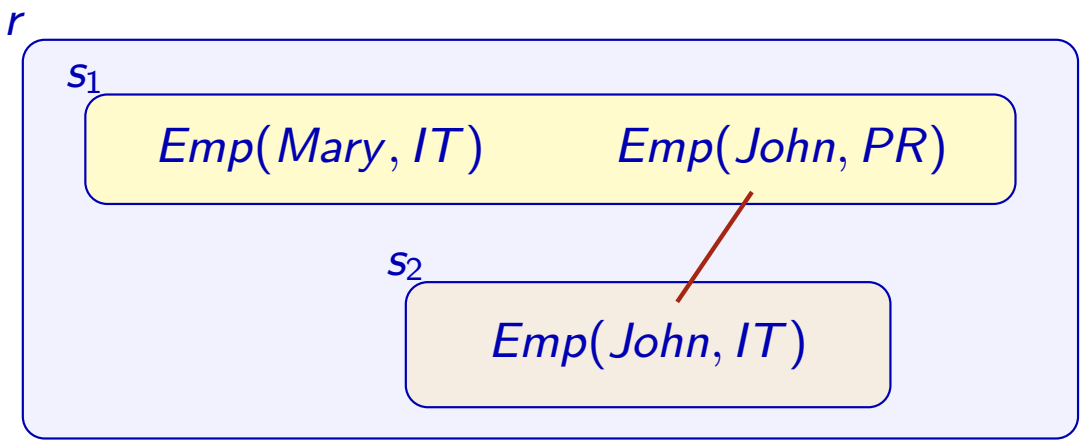
Motivation

Schema

$Emp(\underline{Name}, Dept)$

Q: who works in IT

Q(X) :- Emp(X, IT) .
Answers are: $\{Mary, John\}$
Are we sure about John ?



What about data cleaning?

- all conflicting tuples are removed
- "clean" instance $r' = \{Emp(Mary, IT)\}$
- may cause loss of information
- sources may be autonomous
- inconsistencies may be results of updates in progress, long running transactions, manual entry, exceptions from the constraints

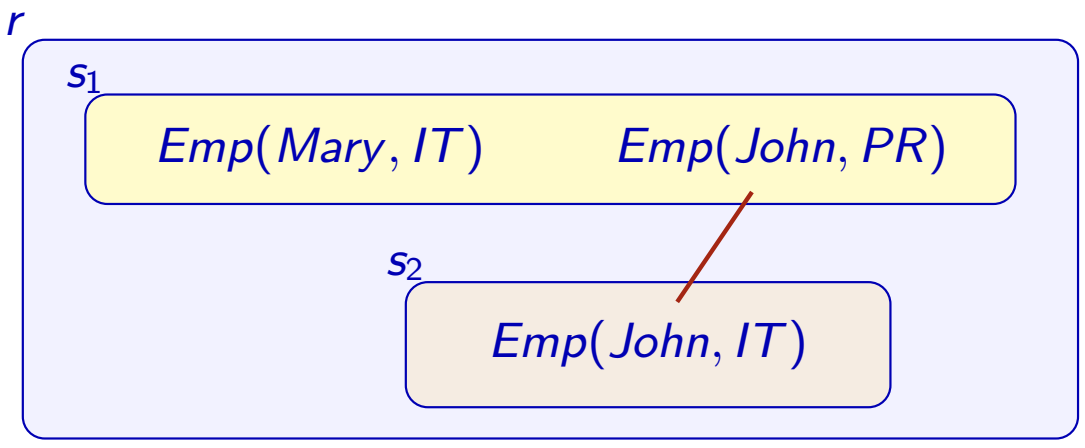
Motivation

Schema

$Emp(\underline{Name}, Dept)$

Q: who works in IT

Q(X) :- Emp(X, IT) .
Answers are: $\{Mary, John\}$
Are we sure about John ?



Repairs [Arenas et al. PODS'99]:

$r_1 = \{Emp(John, IT), Emp(Mary, IT)\}$
$r_2 = \{Emp(John, PR), Emp(Mary, IT)\}$

Consistent answers are:

$\{Mary\}$
John is not a consistent answer because of r_2

Contributions

1. Analysis of computational complexity of consistent query answers (CQA) in the presence of universal constraints.
2. Building a practical system for computing CQA.
3. Developing a general framework for preference-based conflict resolution.
4. Adapting the framework of CQA to semi-structured databases.

Integrity constraints

1. universal constraints

$$\forall^* \neg [R_1(\bar{x}_1) \wedge \ldots \wedge R_n(\bar{x}_n) \wedge \neg P_1(\bar{y}_1) \wedge \ldots \wedge \neg P_m(\bar{y}_m) \wedge \rho]$$

   where $\bar{y} \subseteq \bar{x}$ (for safety [Abiteboul et al.])

2. Horn constraints

$$R_1(\bar{x}_1) \wedge \ldots \wedge R_n(\bar{x}_n) \wedge \rho \Rightarrow P(\bar{y})$$

3. denial constraints

$$\forall^* \neg [R_1(\bar{x}_1) \wedge \ldots \wedge R_n(\bar{x}_n) \wedge \rho]$$

4. functional dependencies (FD)

$$R : X \rightarrow Y$$

Query languages for relational databases

1. relational algebra (a subclass of SQL)

$$E ::\equiv R \mid \sigma_{\varphi}(E) \mid \pi_{X}(E) \mid E_1 \times E_2 \mid E_1 \cup E_2 \mid E_1 \setminus E_2.$$

2. first-order queries

$$\varphi ::\equiv R(\bar{x}) \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \exists x.\varphi(x) \mid \forall x.\varphi(x).$$

3. conjunctive queries

$$Q(\bar{x}) = \exists \bar{y}.R_1(\bar{z}_1) \wedge \ldots \wedge R_k(\bar{z}_k).$$

Repairs and Consistent Query Answers [PODS'99]

Repair:

a consistent instance minimally different from the database

$R : A \rightarrow B$

A	B
1	0
1	1
2	0
2	1

Repairs:

$r_1 = \{R(1,0), R(2,0)\}$
$r_2 = \{R(1,0), R(2,1)\}$
$r_3 = \{R(1,1), R(2,0)\}$
$r_4 = \{R(1,1), R(2,1)\}$

There can be an exponential number of repairs

Consistent Query Answers:

answers present in every repair.

Computational complexity for universal constraints

Data complexity

- query and IC assumed to be fixed
- input size depends on the DB size only
- standard measure in relational databases [Vardi, STOC'82]

Constraints	Repair Checking	Consistent Answers to	
		$\{\forall, \exists\}$-free queries	conjunctive queries
Universal	coNP-c	Π_2^P-c	Π_2^P-c
Horn	PTIME	coNP-c	coNP-c
Acyclic Horn	PTIME	PTIME	coNP-c
Denial	PTIME	PTIME	coNP-c

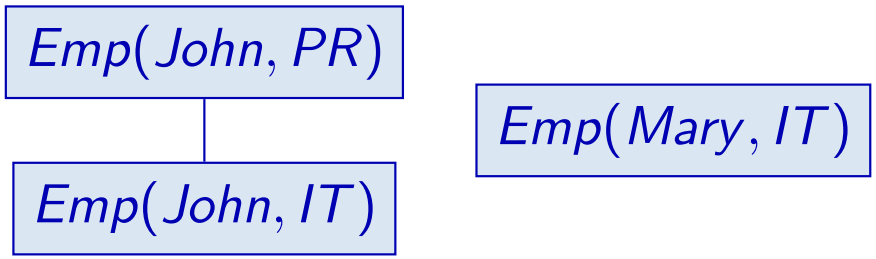
[Chomicki and Marcinkowski, IC'05]

Conflict hypergraph [Chomicki and Marcinkowski, IC'05]

Denial constraint ($Emp : Name \rightarrow Dept$)

$\forall n, d_1, d_2.\neg[Emp(n, d_1) \wedge Emp(n, d_2) \wedge d_1 \neq d_2]$

Conflict: $\{Emp(John, IT), Emp(John, PR)\}$



Conflict graph

- vertices = tuples from DB
- edges connect conflicting tuples
- repairs $\cong$ maximal independent sets
- used to effectively compute CQA

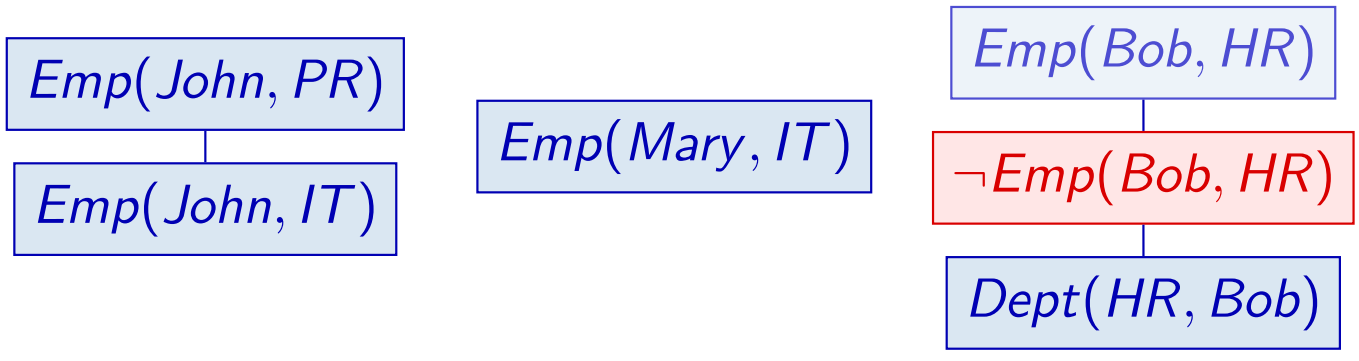
Extended conflict hypergraph

Universal constraint

$(Dept(DeptName, MgrName) \Rightarrow Emp[MgrName, DeptName])$

$$\forall n_1, n_2, d_1, d_2. \neg[Dept(d, m) \wedge \neg Emp(m, d)]$$

Conflict: $\{Dept(HR, Bob), \neg Emp(Bob, HR)\}$



Extended conflict graph:

- vertices = tuples present and missing in DB
- edges connect conflicting tuples
- repairs $\cong$ independent sets minimally different from DB

Computing consistent answers to π-free queries

System Hippo:

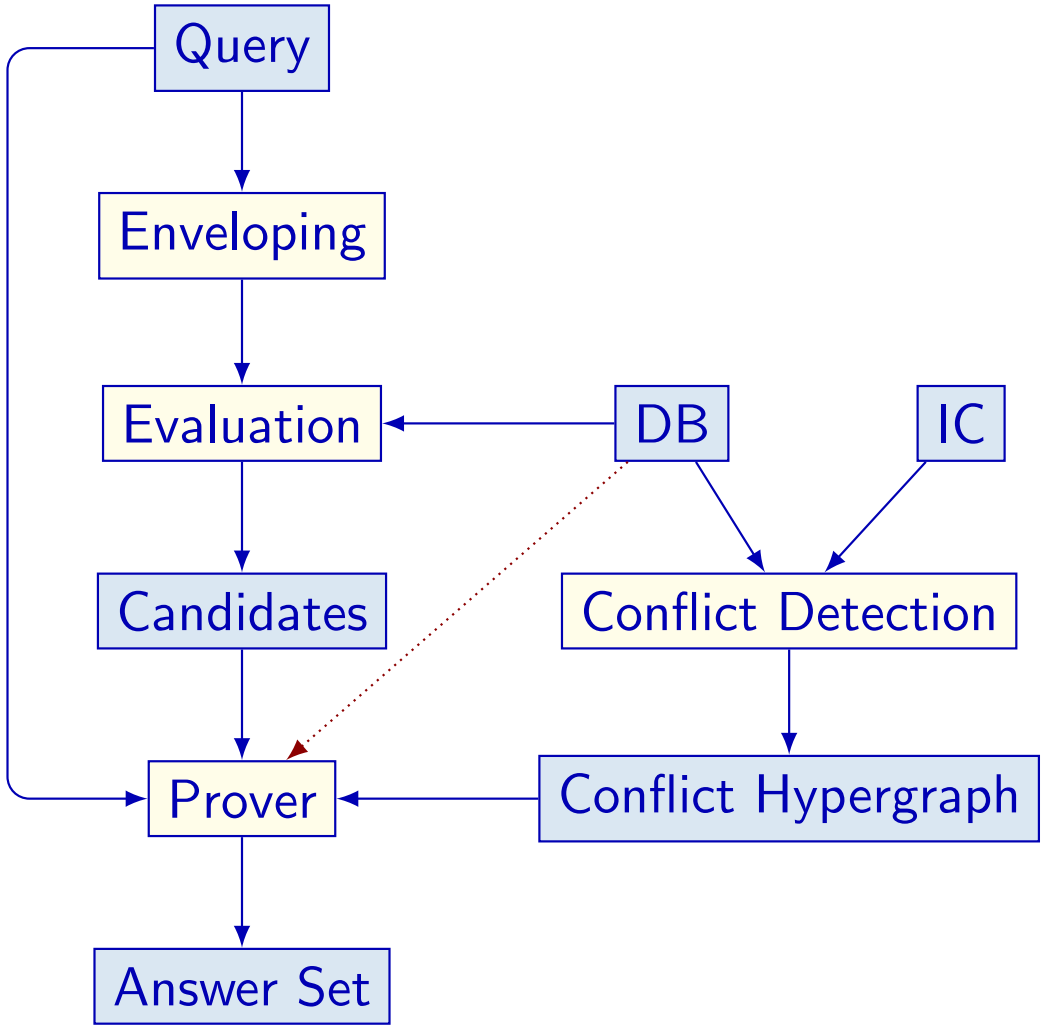
- π-free queries
- denial constraints
- implemented in Java
- JDBC front-end

Efficiency

- for $\{\sigma, \times\}$-queries no overhead
- for $\{\sigma, \times, \cup, \setminus\}$-queries twice the time of standard evaluation

Handling projection

co-NP-complete problem



Related work: Computing consistent query answers

1. Query rewriting [Arenas et al., PODS'99]
   - A query Q is rewritten to a query Q' defining consistent answers to Q.
   - easy to incorporate into already existing applications
   - relatively small overhead
   - applicable to a limited class of queries and constraints
   - ConQuer: Select-Project-Join SQL queries in the presence of key dependencies [Fuxman et al., SIGMOD'05].
2. Logic programs
   - logic programs specify all repairs
   - evaluation with existing systems like `dlv` or `smodels`
   - very general: arbitrary first order queries and universal constraints (even some referential constraints)
   - computationally expensive: Π_2^p-completeness of logic programs
   - INFOMIX: incorporates various optimizations [Eiter et al., SIGMOD'05]

Further work

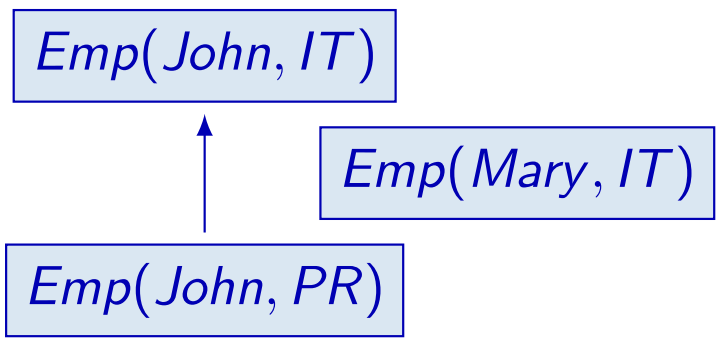
1. Adding projection (π)
2. Analysis of inconsistencies encountered in practice
3. Benchmarking CQA systems
4. Universal constraints without the safety restriction

$$\forall x.\neg[\neg P(x) \wedge x < 100] \equiv \forall x < 100.P(x)$$

Priorities, Preferences, and Cleaning

Priority $\succ$

- an acyclic orientation of the conflict graph
- $\succ$ is called total when all edges are oriented



$Emp(John, PR) \succ Emp(John, IT)$

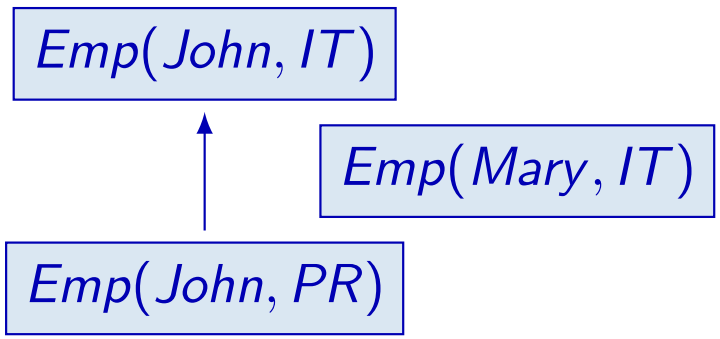
Priorities, Preferences, and Cleaning

Priority $\succ$

- an acyclic orientation of the conflict graph
- $\succ$ is called total when all edges are oriented

Preferred CQA

- use $\succ$ to define a family of preferred repairs $Rep(\succ)$
- preferred consistent answers w.r.t. $\succ$ are the answers present in every preferred repair w.r.t $\succ$

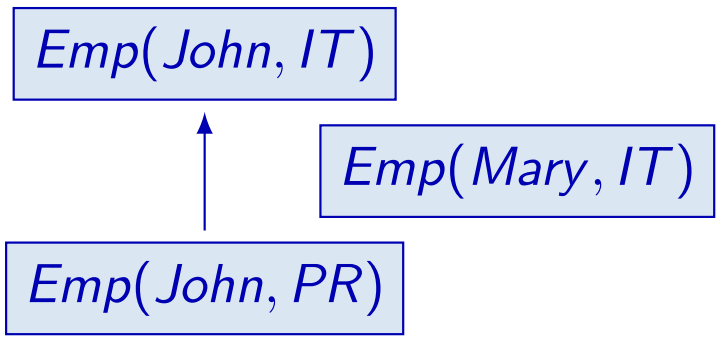


$Emp(John, PR) \succ Emp(John, IT)$

Priorities, Preferences, and Cleaning

Priority $\succ$

- an acyclic orientation of the conflict graph
- $\succ$ is called total when all edges are oriented



$Emp(John, PR) \succ Emp(John, IT)$

$$\omega_{\succ}(r) = \{t \in r | \neg \exists t' \in r.t' \succ t\}$$

Preferred CQA

- use $\succ$ to define a family of preferred repairs $Rep(\succ)$
- preferred consistent answers w.r.t. $\succ$ are the answers present in every preferred repair w.r.t $\succ$

Database cleaning with a total $\succ$

- $r' := \varnothing$
- **while** $\omega_{\succ}(r) \neq \varnothing$ **do**
  1. **choose any** $x \in \omega_{\succ}(r)$
  2. add x to r'
  3. remove x from r with neighbors
- **return** r'

Desirable Properties of Preferred Repairs

($\mathcal{P}1$) Non-emptiness

$$Rep(\succ) \neq \varnothing$$

($\mathcal{P}2$) Monotonicity

$$\succ_1 \subseteq \succ_2 \Rightarrow Rep(\succ_2) \subseteq Rep(\succ_1)$$

($\mathcal{P}3$) Non-discrimination

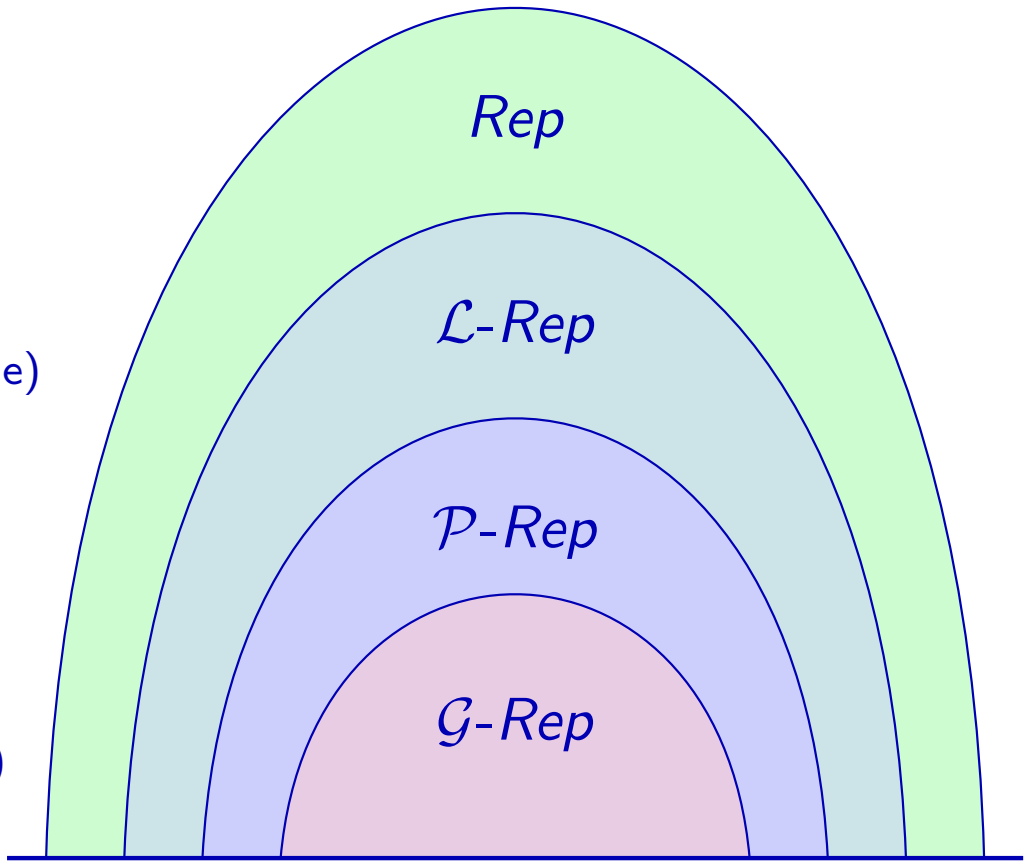
$$Rep(\varnothing) = Rep$$

($\mathcal{P}4$) Categoricity

$$\succ \text{ is total} \Rightarrow |Rep(\succ)| = 1$$

Optimal Use of Priorities

Properties:	Complexity: (of CQA)	Priority enforcement:
$\mathcal{P}1$–$\mathcal{P}3$	PTIME	None
$\mathcal{P}1$–$\mathcal{P}3$	co-NP-c	Local (single tuple)
$\mathcal{P}1$–$\mathcal{P}4$	co-NP-c	Pareto
$\mathcal{P}1$–$\mathcal{P}4$	Π_2^p-c	Global (tuple sets)



$\mathcal{C}$-*Rep*: Common optimal repairs

Common optimal repairs

$r' \in \mathcal{C}\text{-}Rep(\succ)$ iff r' is a result of cleaning the database with $\succ$.

Properties

- $\mathcal{C}$-*Rep* satisfies $\mathcal{P}1 - \mathcal{P}4$
- $\mathcal{C}\text{-}Rep \subseteq \mathcal{G}\text{-}Rep$
- $\mathcal{C}\text{-}Rep = \mathcal{G}\text{-}Rep$ for priorities that cannot be extended to a cyclic orientation
- CQA: co-NP-complete

Related work: Preferred consistent query answers

1. Conditioned active integrity constraints [Flesca, et all., PPDP'04] allow to specify how to resolve some conflicts
   - repairs may be not optimal
   - $\mathcal{P}1$ and $\mathcal{P}2$ are satisfied
   - $\mathcal{P}3$ and $\mathcal{P}4$ are violated
2. Repair constraints can be used to restrict considered repairs [Greco and Lembo, ER'04]
   - $\mathcal{P}3$ is satisfied, but $\mathcal{P}1$ is violated
   - in a weaker version: $\mathcal{P}1$ is satisfied, but $\mathcal{P}3$ is violated

Further work

1. Experimental evaluation
   - one FD $\longrightarrow$ PTIME
2. Cyclic priorities (what about monotonicity)
3. Generalization to denial and universal constraints

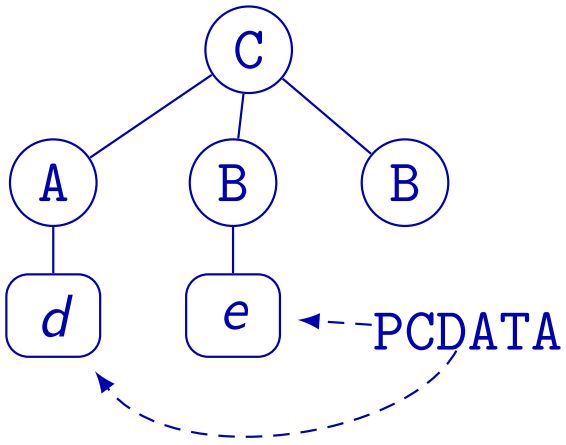
XML and DTD

XML Documents

- ordered finite trees
- labels from finite Σ
- `PCDATA` for text nodes
- text labels from Γ

DTD is a function that:

- takes the label of a node
- returns a regular expression to which the labels of the children must comply



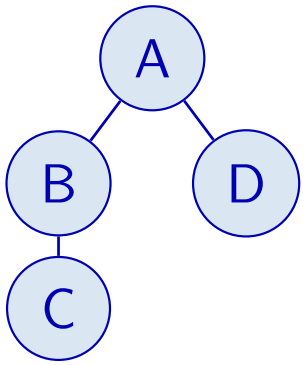
`C(A(d),B(e),B)`

Example of DTD

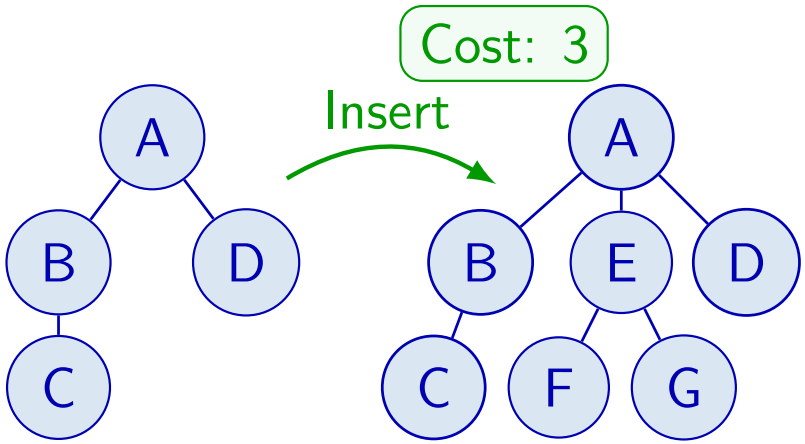
C $\longrightarrow$ (A,B)*
A $\longrightarrow$ EMPTY | PCDATA
B $\longrightarrow$ EMPTY

`C(A(d),B(e),B)` doesn't satisfy D, but `C(A(d),B())` does.

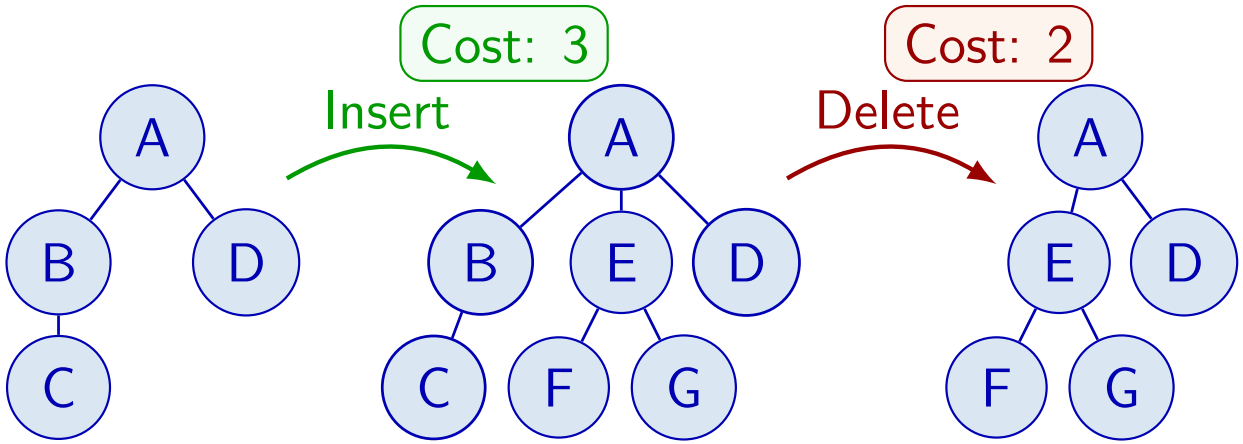
Editing operations



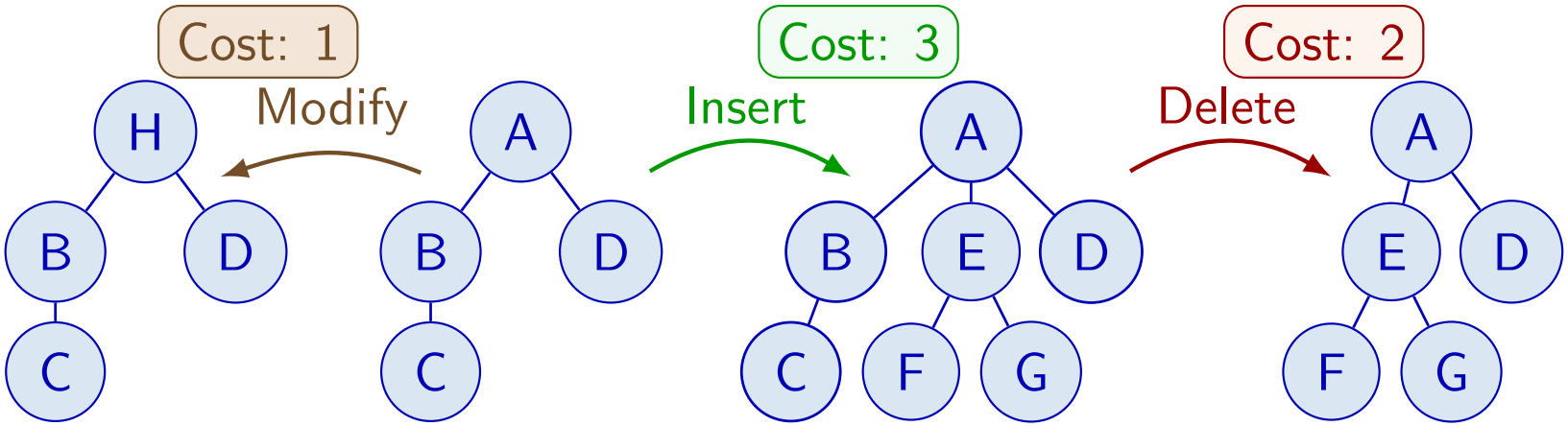
Editing operations



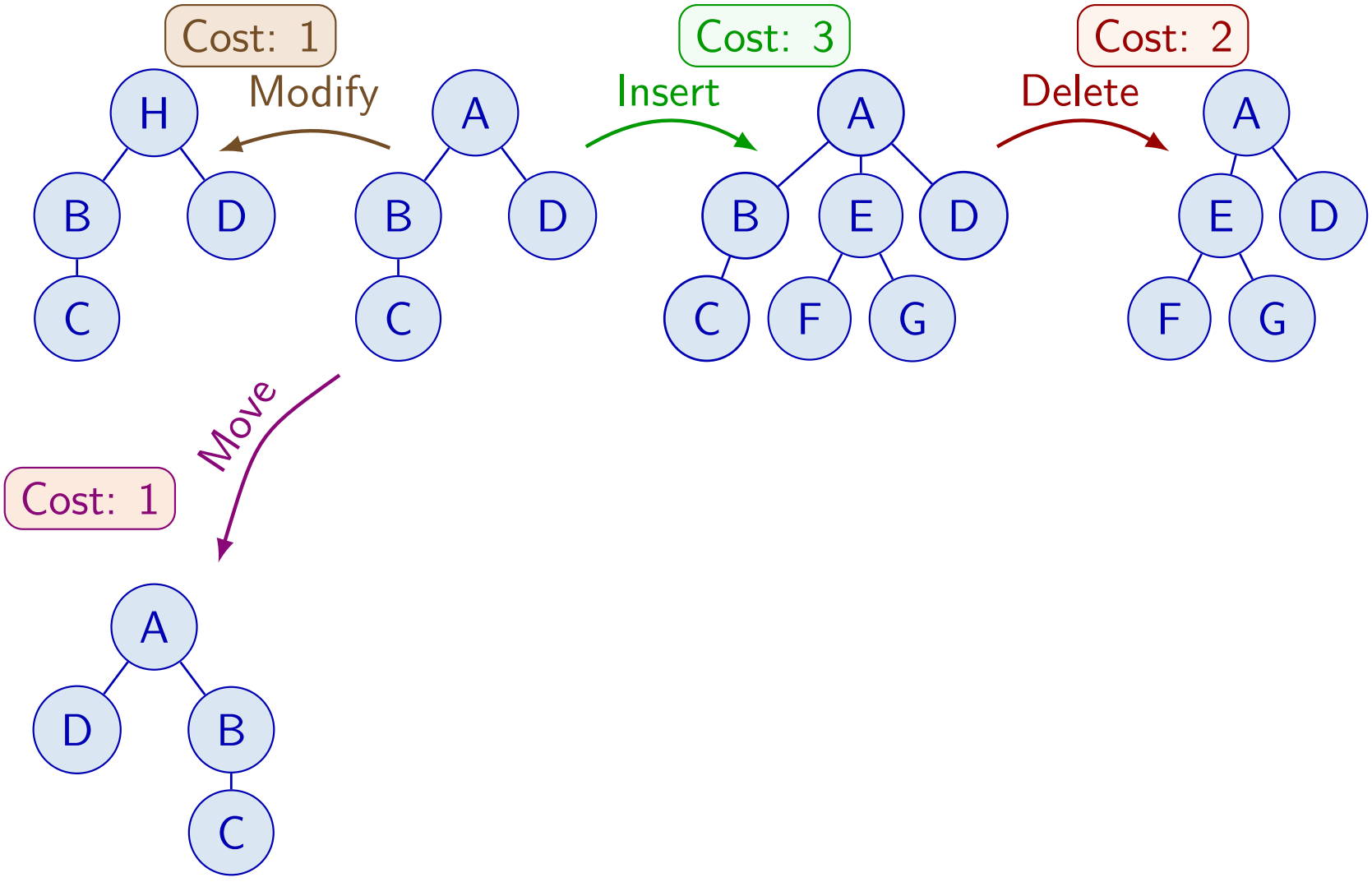
Editing operations



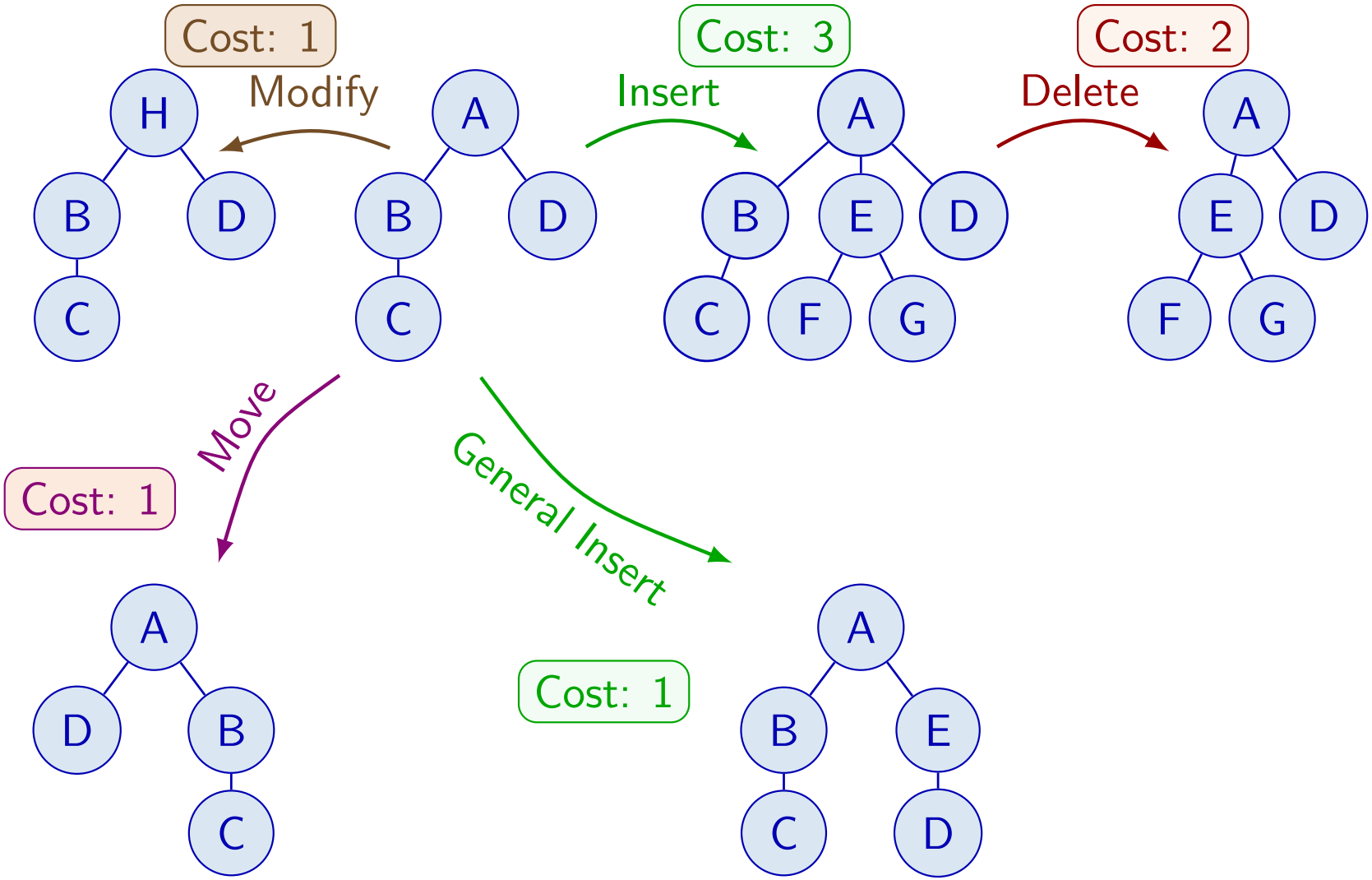
Editing operations



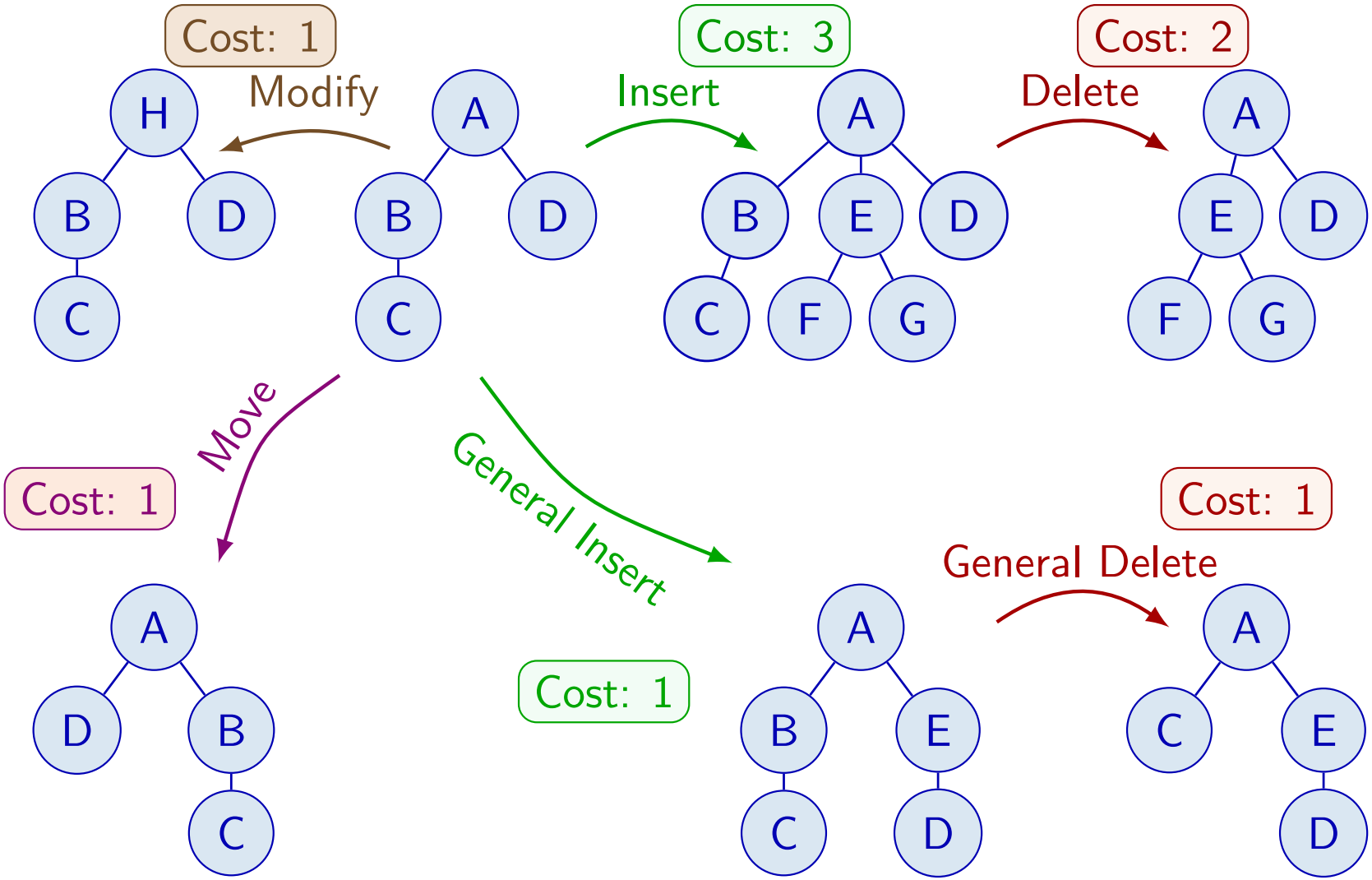
Editing operations



Editing operations



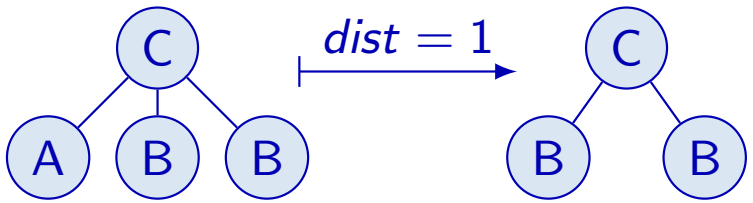
Editing operations



Edit distance, repairs, and valid query answers

Distance between documents

$dist(T, S)$ is the minimal cost of transforming T into S



Edit distance, repairs, and valid query answers

Distance between documents

$dist(T, S)$ is the minimal cost of transforming T into S

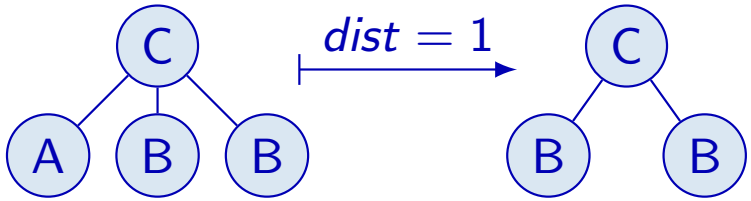
Distance to a DTD

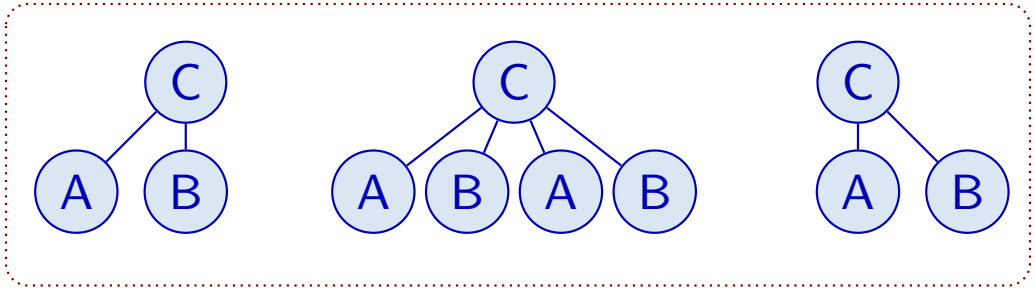
$dist(T, D)$ is the minimal cost of repairing T w.r.t D i.e.,

$\min\{dist(T, S) | S \text{ valid w.r.t } D\}$

DTD

C → (A,B)*
A → EMPTY | PCDATA
B → EMPTY





Edit distance, repairs, and valid query answers

Distance between documents

$dist(T, S)$ is the minimal cost of transforming T into S

Distance to a DTD

$dist(T, D)$ is the minimal cost of repairing T w.r.t D i.e.,

$\min\{dist(T, S) | S \text{ valid w.r.t } D\}$

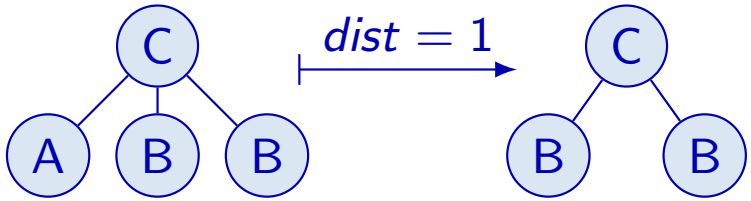
DTD

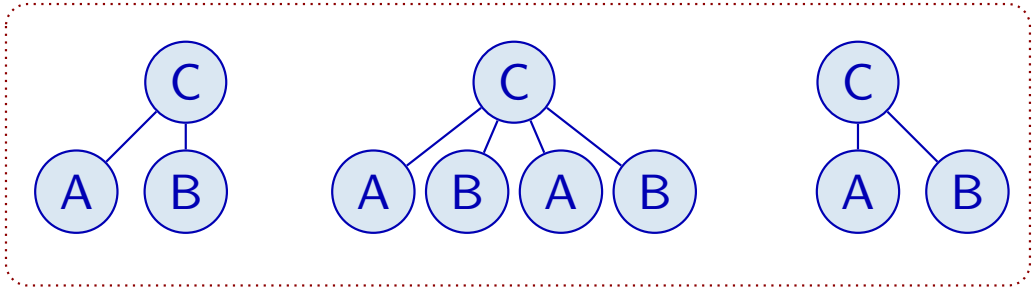
C $\rightarrow$ (A,B)*
A $\rightarrow$ EMPTY | PCDATA
B $\rightarrow$ EMPTY

Repair

T' is a repair of T w.r.t D iff

$$dist(T', T) = dist(T, D)$$

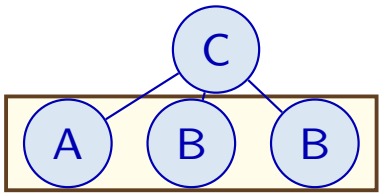




Valid Query Answers

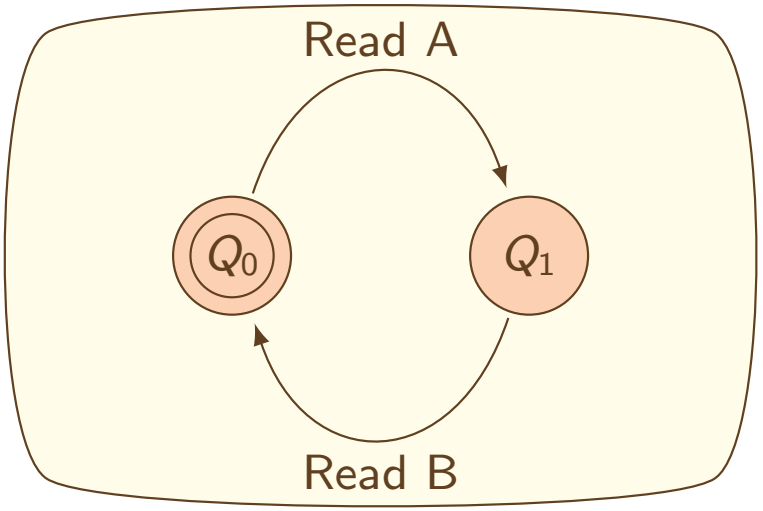
x is a valid answer to query Q in T w.r.t. D iff
x is an answer to Q in every repair of T w.r.t. D.

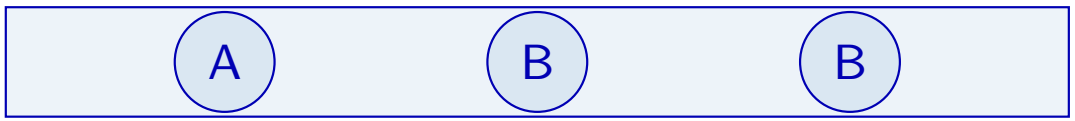
Trace graph



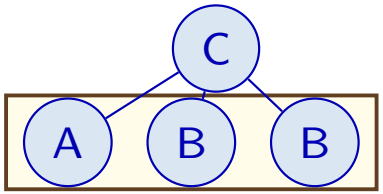
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



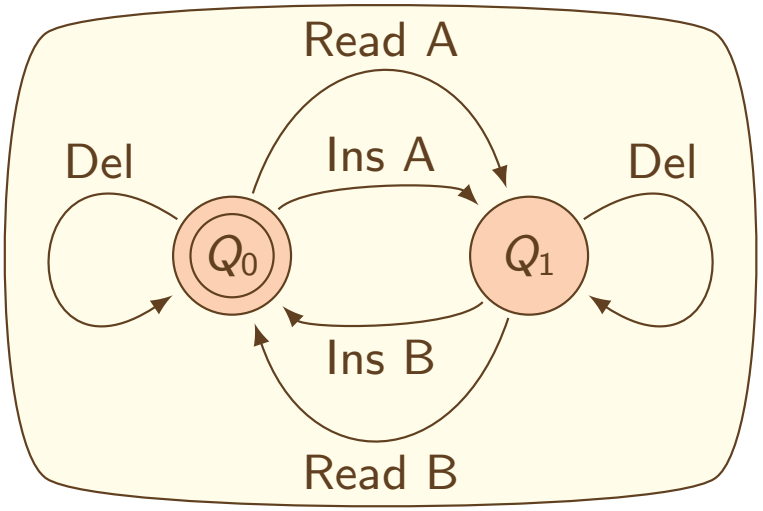


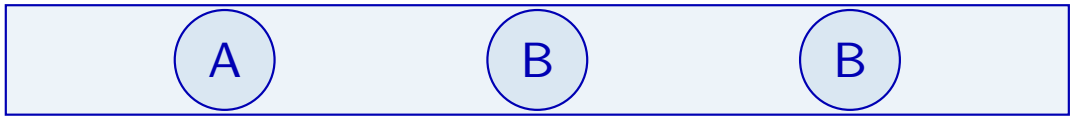
Trace graph



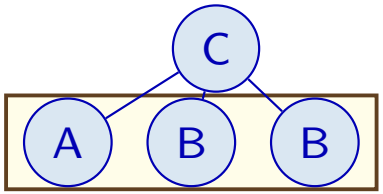
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



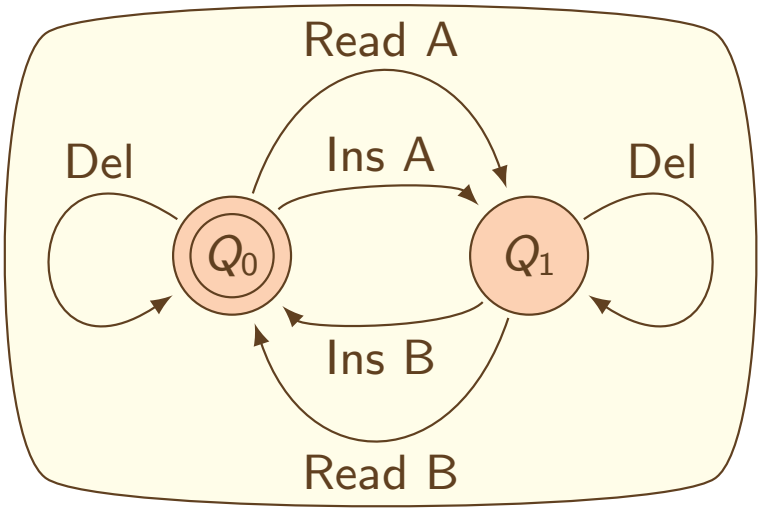


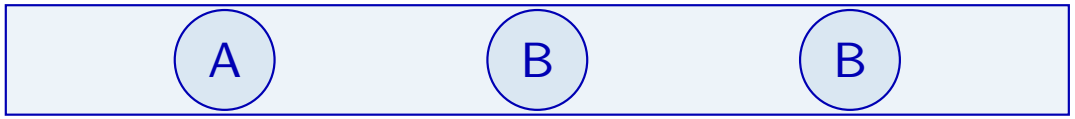
Trace graph



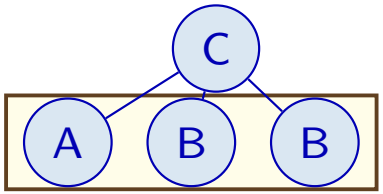
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



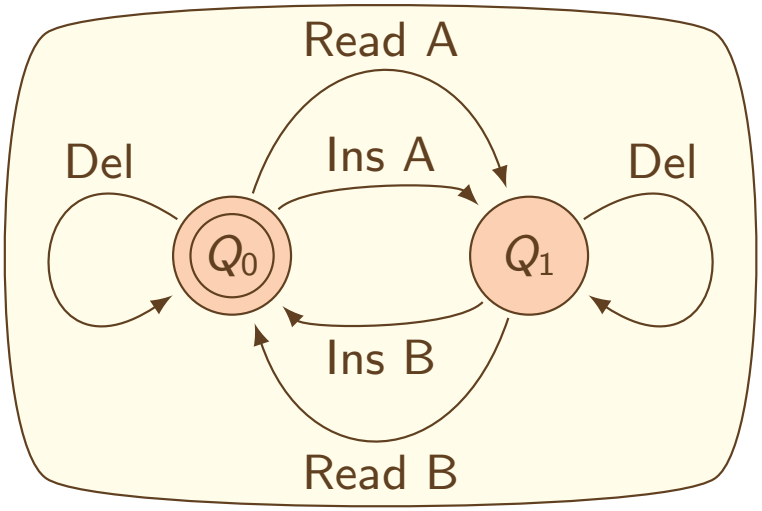


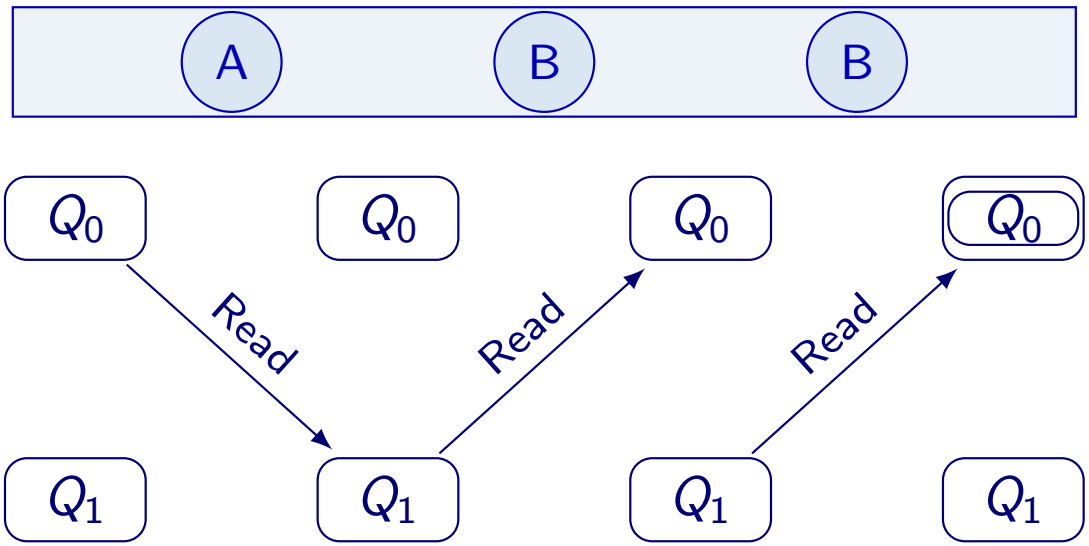
Trace graph



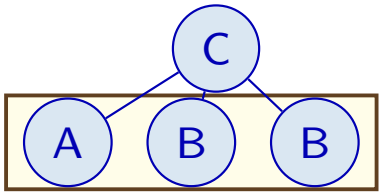
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



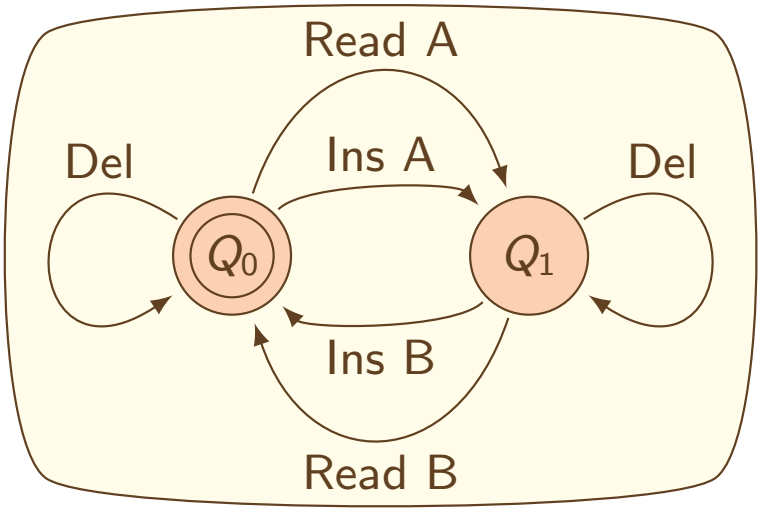


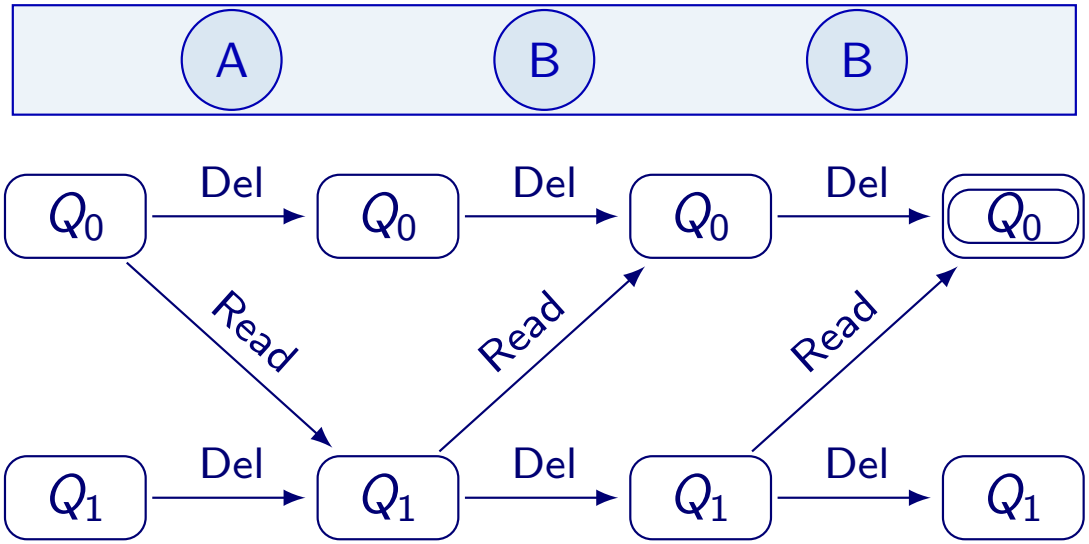
Trace graph



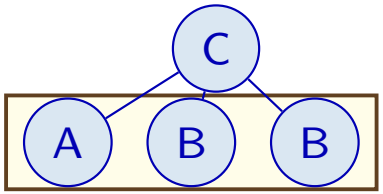
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



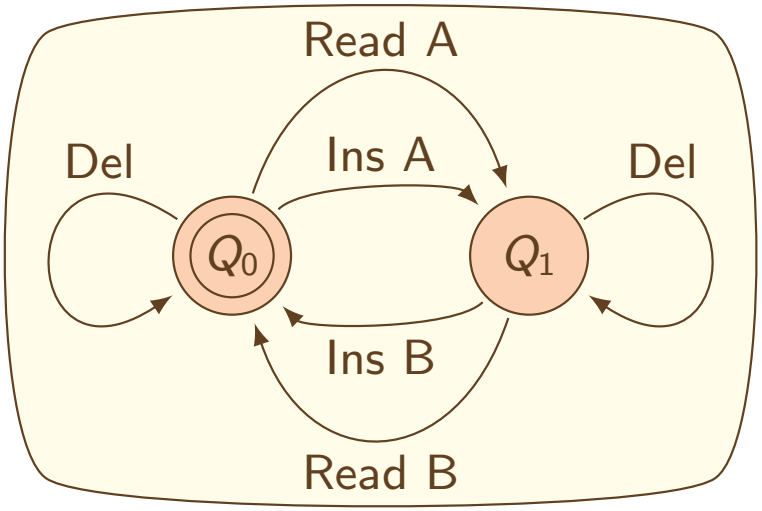


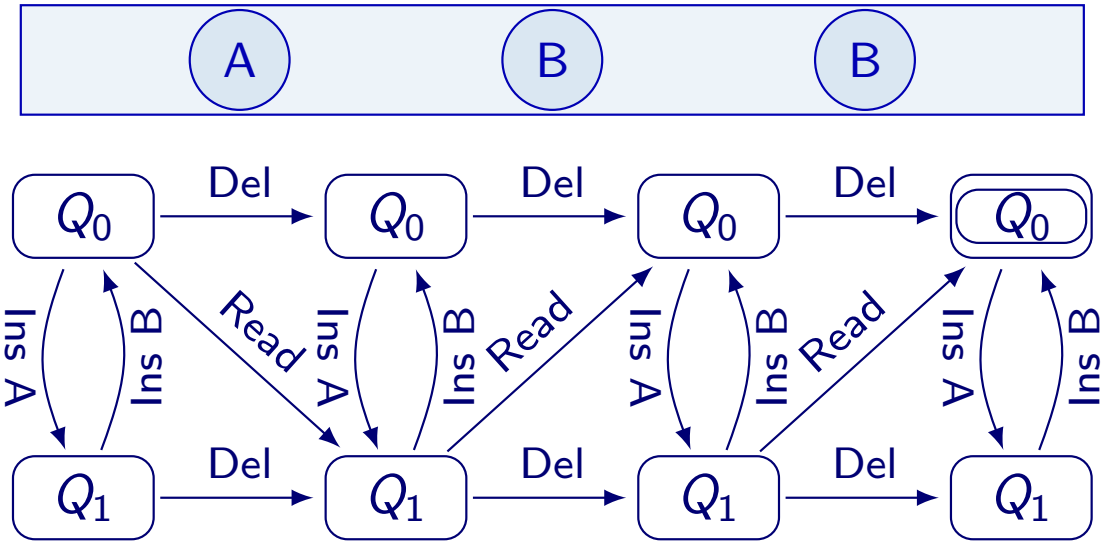
Trace graph



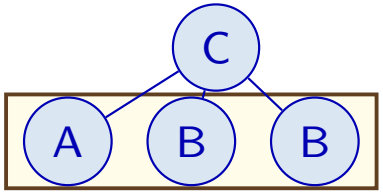
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



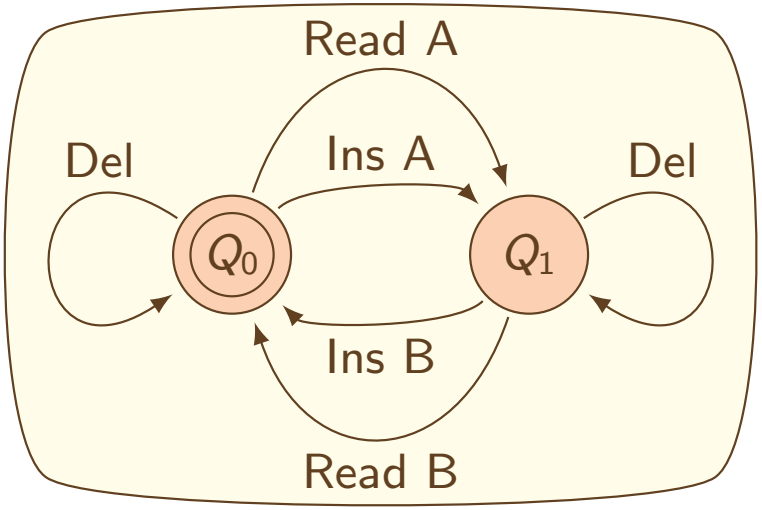


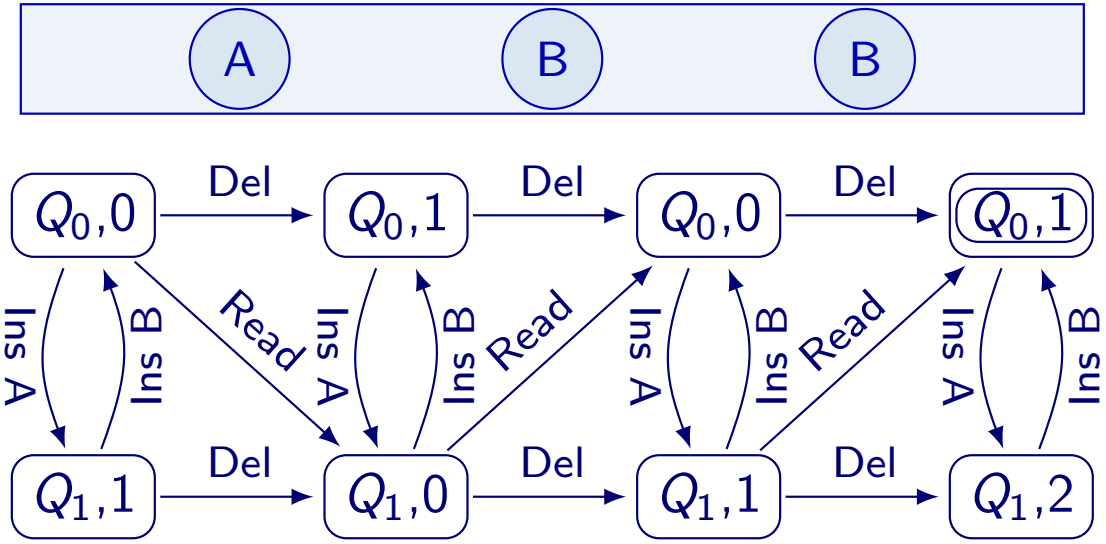
Trace graph



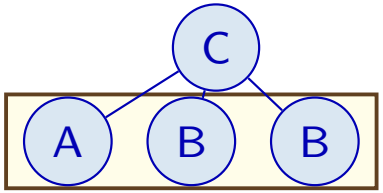
DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



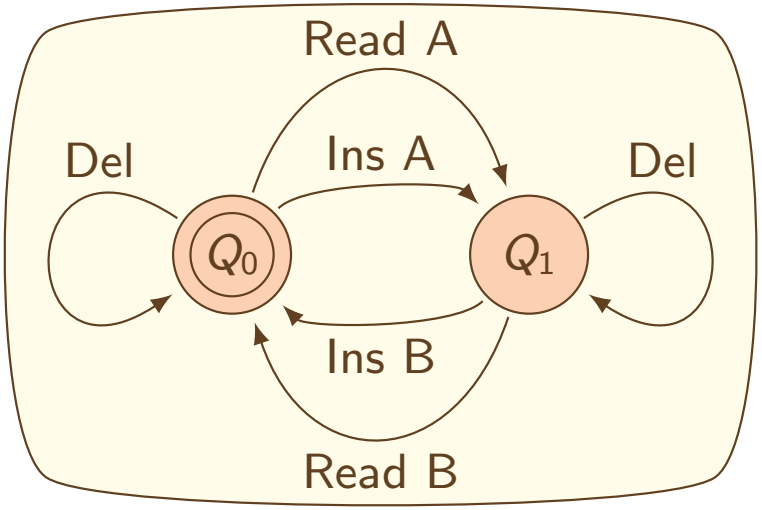


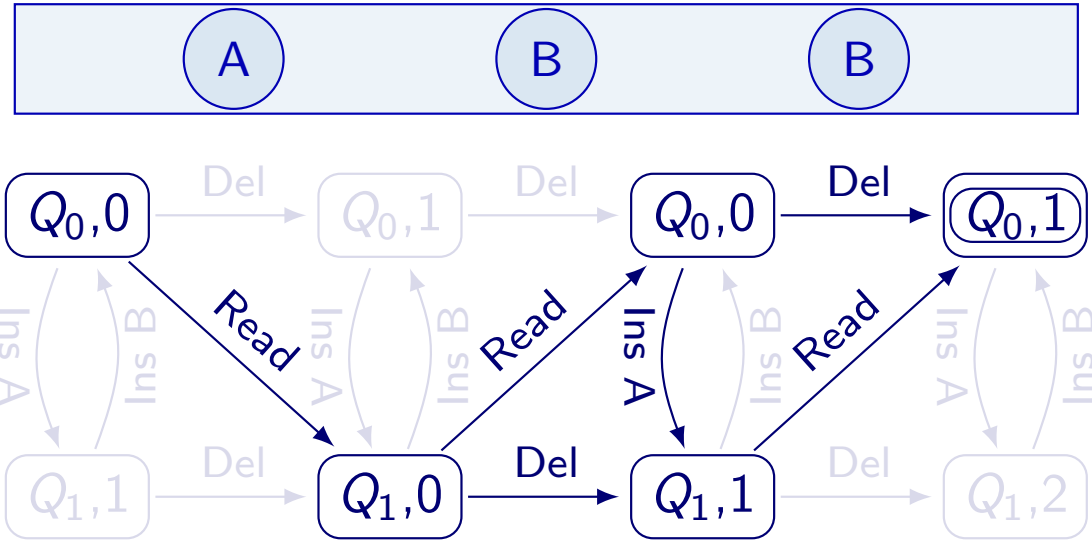
Trace graph



DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY

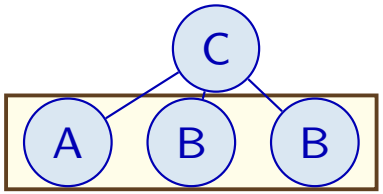




Repairing Paths:

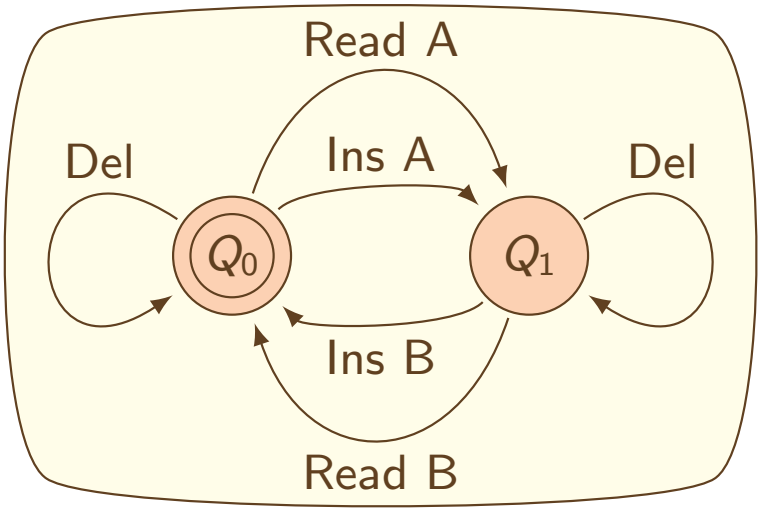
- (Read, Read, Del)
- (Read, Read, Ins A, Read)
- (Read, Del, Read)

Trace graph

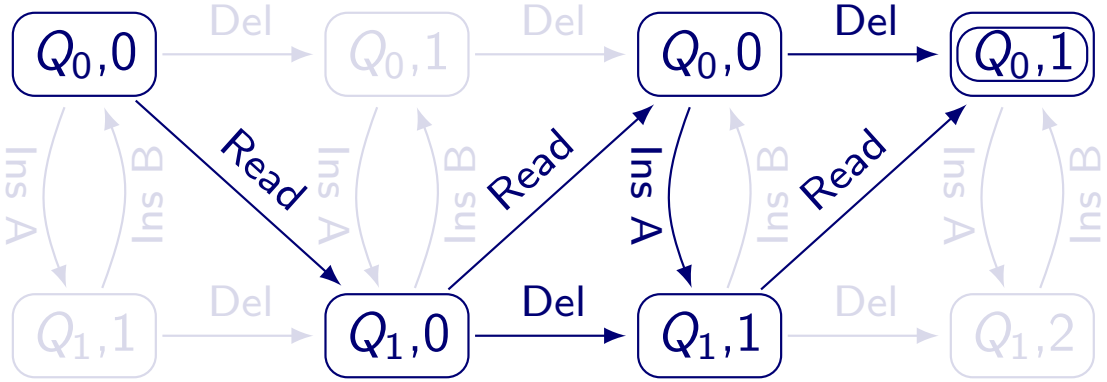


DTD

C → (A,B)*
A → EMPTY
| PCDATA
B → EMPTY



Compact representation of all repairs



Repairing Paths:

- (Read, Read, Del)
- (Read, Read, Ins A, Read)
- (Read, Del, Read)

Complexity of VQA

XPath queries

sequences of steps (with filter conditions) to navigate through the tree.

Combined complexity

- both the document and the query are parts of the input
- standard evaluation in PTIME [Gottlob et al., PODS'03]

Positive results

- VQA in PTIME for simple XPath queries (only descending axes, no negation, no union/disjunction, and no join conditions)
- Rhino: uses trace graphs to effectively compute VQA

Negative results

VQA is coNP-complete for other XPath queries.

Related work: Consistent query answers for XML

1. [Flesca et. all, Xsym'03] queries valid documents that violate FDs
   - two operations: marking nodes as unreliable and setting null value
   - polynomial time algorithm for queries of form $A_1/\ldots/A_n$
   - no experimental evaluation
2. [Flesca et. all, WISE'05] deals with invalid documents and FDs
   - edit operations similar to ours, but
   - a different (set-theoretic) definition of repairs: $\mathtt{C}(\mathtt{A}(\mathtt{d}),\mathtt{B}(\mathtt{e}),\mathtt{B})$ has only one repair $\mathtt{C}(\mathtt{A}(\mathtt{d}),\mathtt{B},\mathtt{A},\mathtt{B})$.
   - polynomial algorithm for a restricted class of queries (no union) and DTDs
   - no experimental evaluation

Further work: Valid query answers

1. Other editing operations
   - General Deletion/Insertion: $dist(T, D)$ computed in $O(|T|^5)$ time [Suzuki, SAC'05].
   - Moving nodes: intractable for string edit distance [Comrode et al., SODA'02] and edit distance on unordered trees [de Rougement, ISIP'03]
2. Incorporation of semantic constraints (e.g., key dependencies)
3. Data complexity

Conclusions

- CQA: foundation of enlivened research in the ares of inconsistent databases (more than 100 papers over the last 10 years)
- practical systems have been constructed
- tractability has been studied
- framework has been adapted to XML databases