

Incremental Evaluation of Schema-Directed XML Publishing

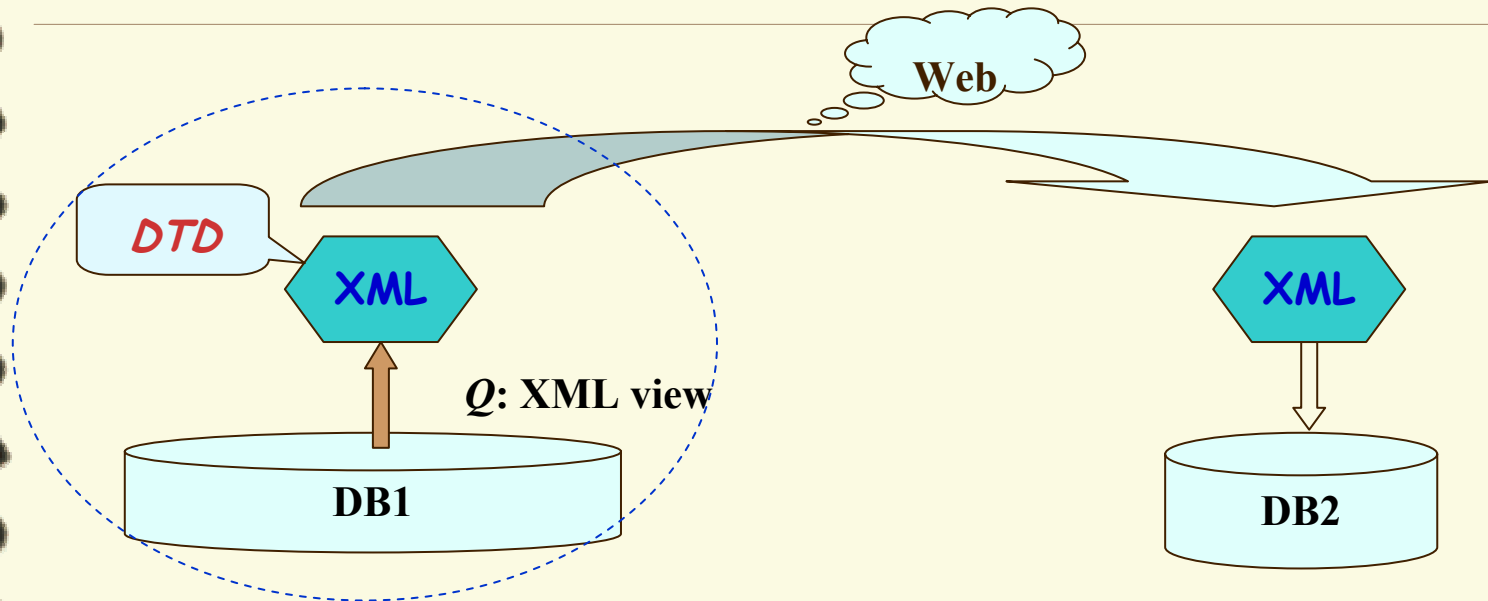
Byron Choi

Supervisor: Wenfei Fan

LFCS, University of Edinburgh



Data exchange on the Web



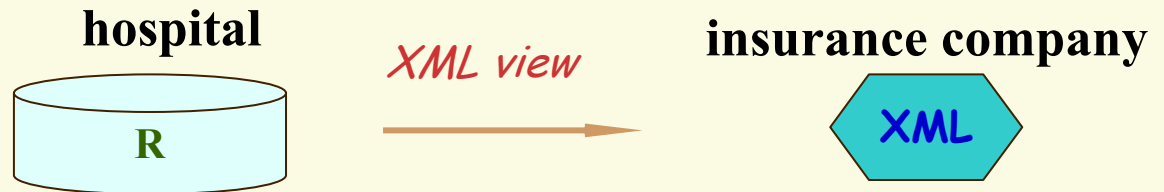
All members of a community (or industry) agree on a **DTD** and then exchange data w.r.t. the DTD: e-commerce, health-care, ...

Schema-directed XML Publishing:

- ✓ mapping relational data to XML
- ✓ conforming to the **predefined DTD**

Real-life example: insurance company and hospital

Report:



Relational database **R** at the hospital:

Patient (SSN, name, policy#)

Visit (SSN, tname, date)

inTreatment (tname, cost)

outTreatment (tname, referral#)

Procedure (tname1, tname2) -- inTreatment hierarchy

treatment

✓ in hospital: composition hierarchy in **Procedure**

Example: insurance company and hospital

DTD D predefined by the insurance company:

report → patient*

patient → SSN, pname, treatment, policy#

treatment → tname, (inTreatment + outTreatment)

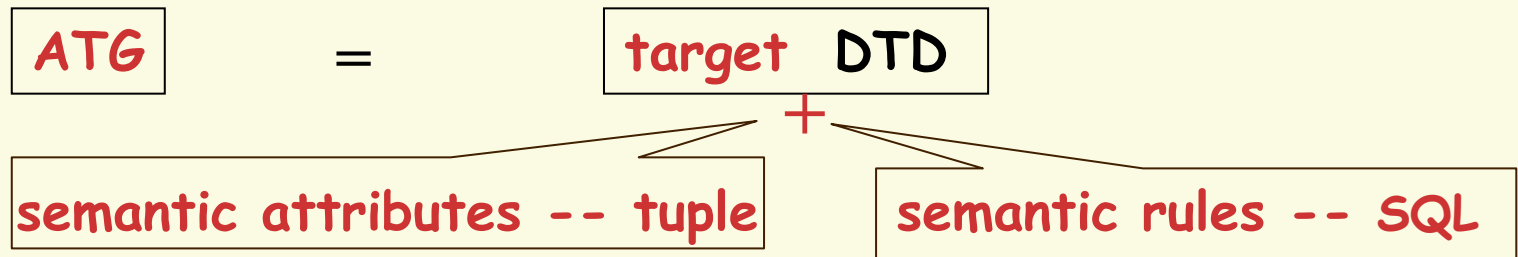
inTreatment → treatment*

outTreatment → referral#

Non-determinism

Recursion

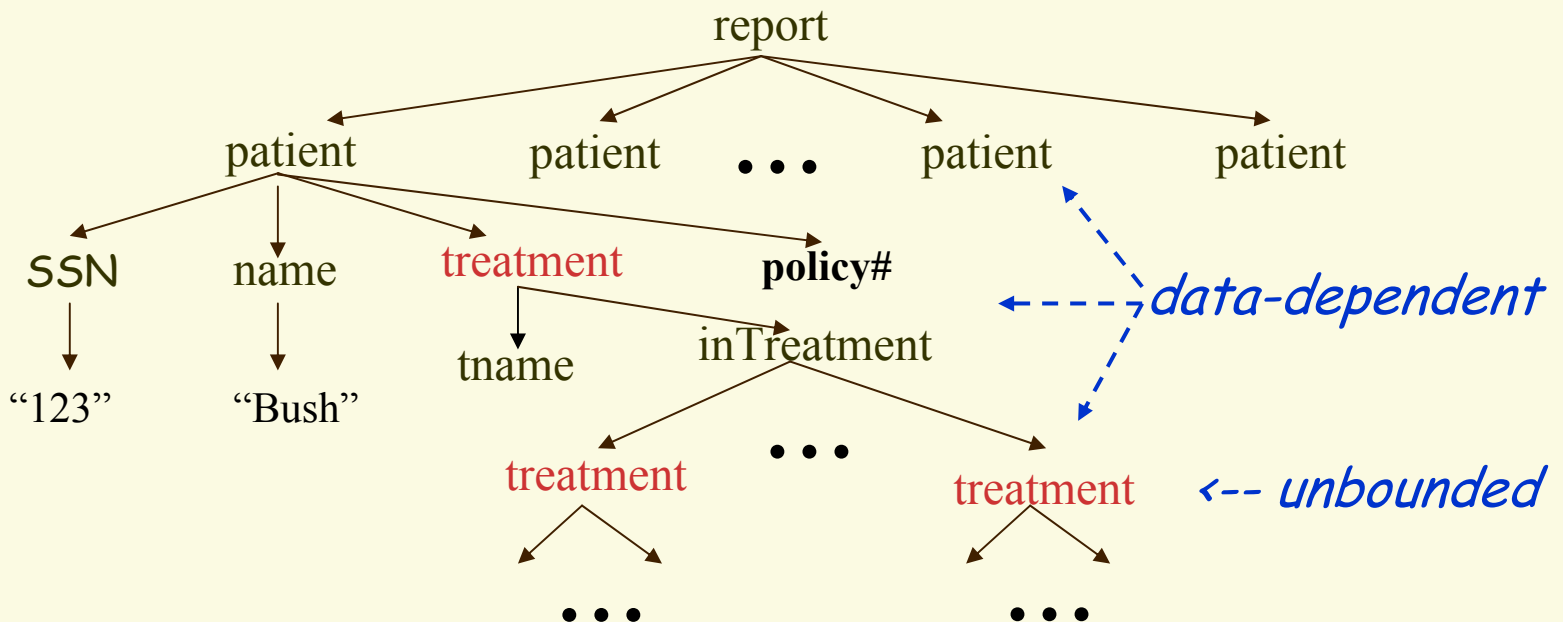
Attribute Transformation Grammar (ATG)



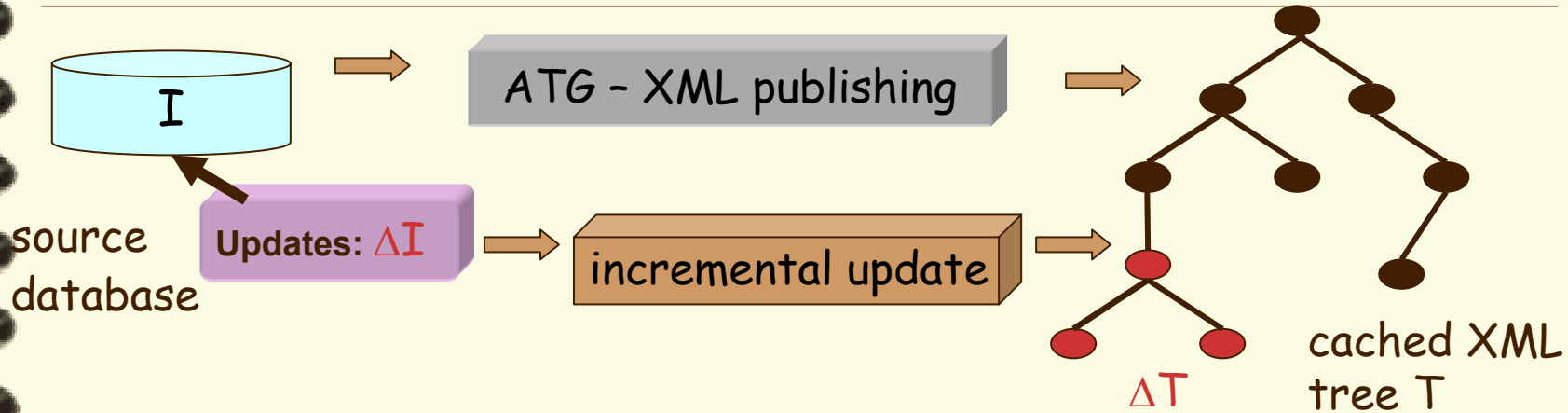
- ✓ DTD: element type definitions $e \rightarrow \alpha$
 $\alpha ::= \text{PCDATA} \mid \varepsilon \mid e_1, \dots, e_k \mid e_1 + \dots + e_k \mid e^*$
- ✓ **Attribute**: associated with each element type e
- ✓ **Rules**: with each $e \rightarrow \alpha$; for e' in α , define $e' = Q(e)$
 - SQL query Q extracts data from **DB**
 - parent attribute e as a **constant parameter** in Q
 - generate e' children of e strictly following α

DTD-directed publishing with ATG

- ✓ **DTD-directed**: the XML tree is constructed **strictly** following the productions of a DTD
- ✓ **Data-driven**: the choice of productions and expansion of the XML tree (recursion) depends on **the source data**



Coping with source updates



Problem: the underlying database may be updated constantly (ΔI),

Goal: update the published (materialized) XML tree in response to source changes ΔI

✓ **Incremental approach**: compute XML change ΔT such that

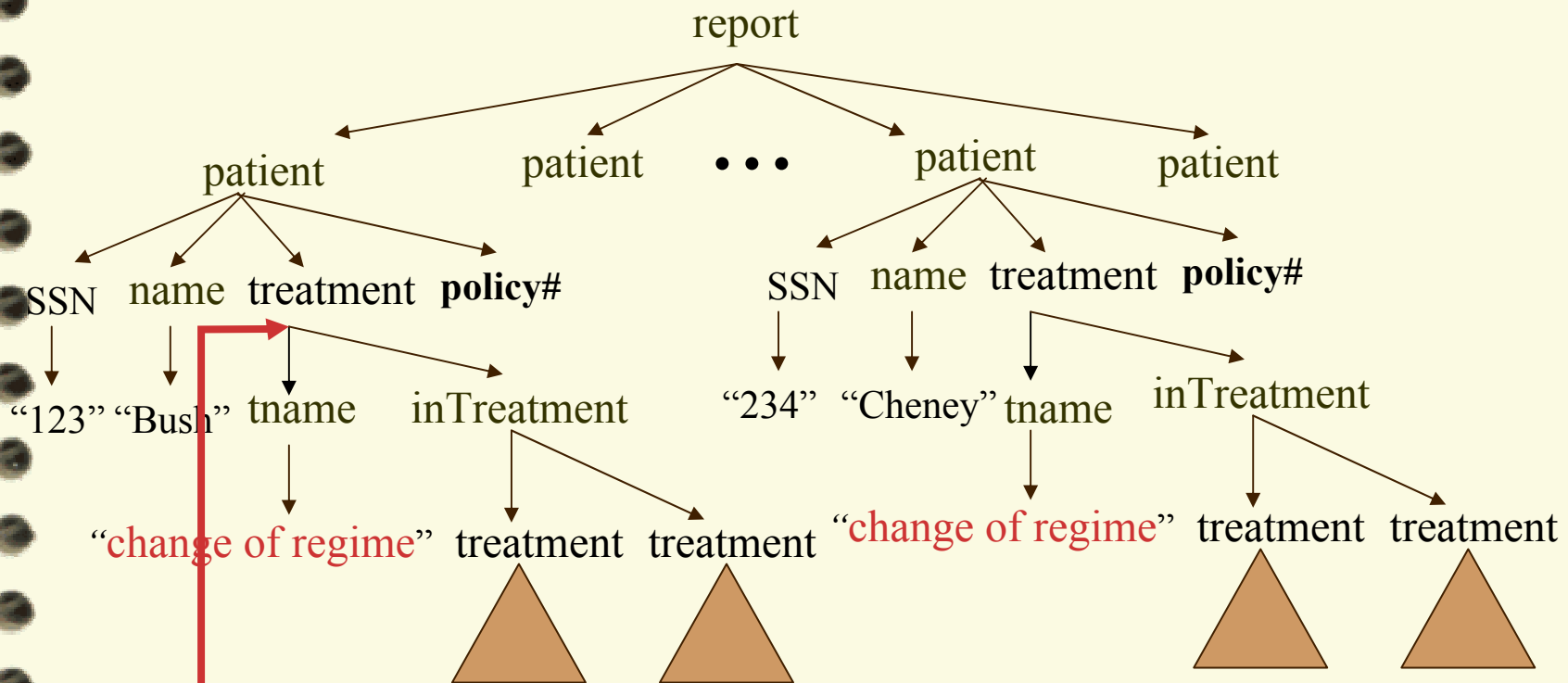
the **new** view T' = the **old** view T + ΔT

Δ ATG - the Bud-Cut Approach

- ✓ Computing XML changes ΔT from database changes ΔI by **incrementalizing** SQL queries in an ATG, e.g., delta edges:

```
select  IP, P.tname2
from     $\Delta$ Procedure P, inTreatment IP
where   P.tname1 = IP
```

Sub-Tree Property



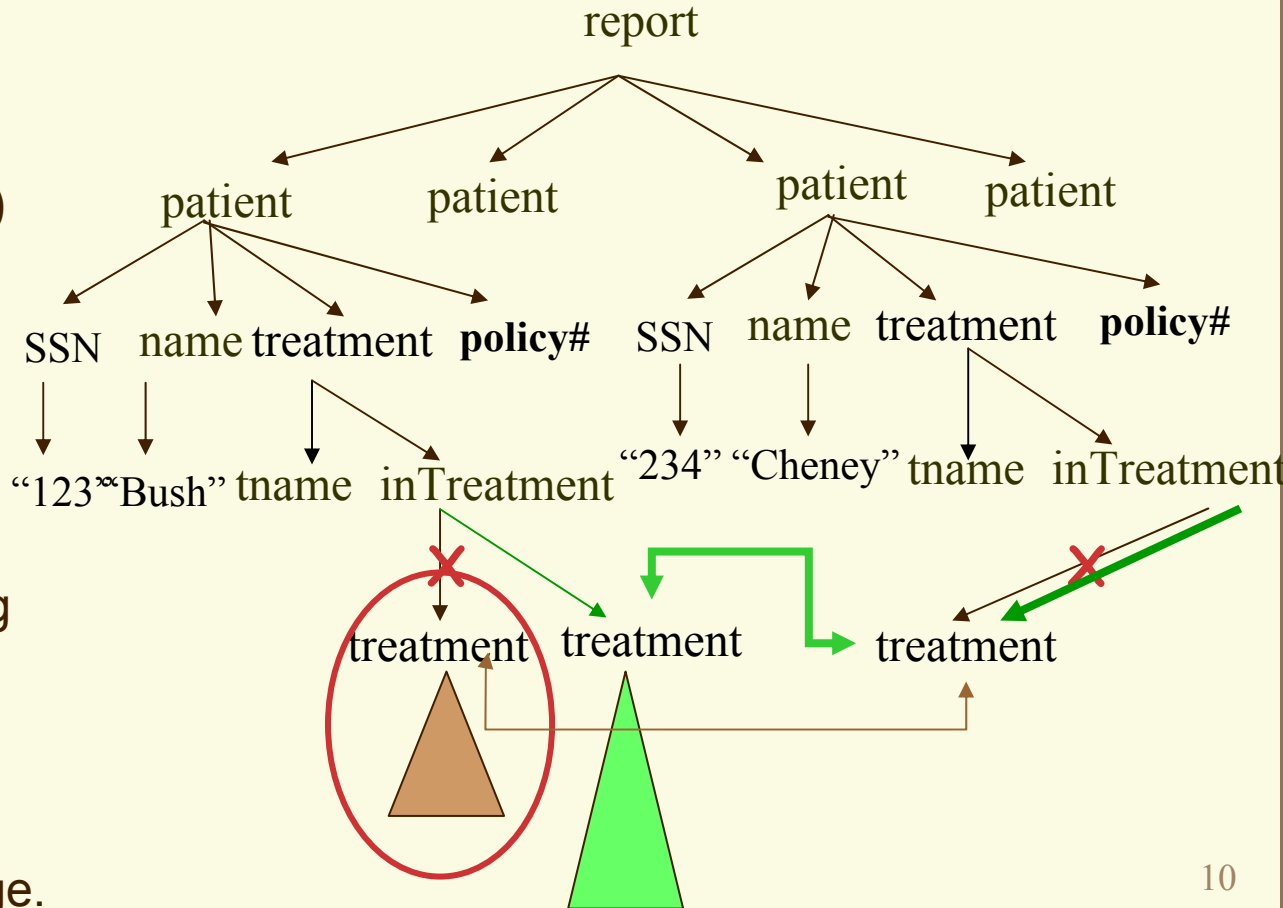
Sub-tree Property: Given a fixed ATG and DB, each sub-tree is uniquely determined by its root attribute

ΔATG - the Bud-Cut (Step by Step)

1. For a set of database changes, ΔI , execute a fixed number of non-recursive queries which determine direct edge changes, E-, E+

E- are **cuts**

E+ are **buds**
(or cross edges)



2. Generate the sub-trees under the buds, re-using as much existing and deleted sub-trees as possible

3. Collect Garbage.

Incremental ATG Evaluation

- ✓ Advanced features:
 - Incorporating **ATG optimizations** (partitioning, materialization), and **overlapping** the cut- and bud- phases. Supporting **lazy evaluation** and **parallel processing**.