CASE
CASE SCHOOL OF ENGINEERING

# Rewriting XPath Queries Using Materialized Views

**Wanhong Xu and Z. Meral Özsoyoğlu**
**Case Western Reserve University**
**Cleveland, Ohio 44106**

# Big Picture

Query rewriting using materialized views in relational databases has been successfully used in

- Query Optimization
- Data Integration
- Data Warehouse Design: View Selection to Materialize
- Semantic Data Caching

# Big Picture

## Materialized XPath Views

- Server Side: Most XML indexing schemes can be modeled as materialized XPath Views [Balmin, et. al. VLDB'04]
- Client Side: Previous XML queries with their results can be seen as materialized XPath views

## Several theoretical studies on

- Containment and Equivalence of XML queries [Miklau, Suciu, PODS'02, JACM'04, Neven, Schwentick, ICDT'03, Grahne, Thomo, PODS'03, Wood, ICDE'03]
- Minimization of XML queries [Wood, WebDB'01, Amer-Yahia, et. al., SIGMOD'02, Flesca, et.al. VLDB'03]
- Complexity of XPath Query Evaluation [Gottlob, et. al. ICDE'03, PODS'04]

Not as much on XPath query rewriting

# Roadmap

Motivation

XML & XPath

Problem Definition

Rewriting Existence Problem

Finding Minimal Rewritings Problem

Conclusion & Future Work

# Motivation

Server Side: XML Document

```
<Pathway name =  "PA1">
  <Reaction name = "RE1">
    <Enzymes>
      <Protein name =
     "PR1“ EC# =" 1.0.0.1"/>
      <RNA name = "RN1"/>
    </Enzymes>
  </Reaction>
  <Reaction >
    <Enzymes>
      <RNA name = "RN2">
    </Enzymes>
  </Reaction>
</Pathway>
```

Client Side:

v: /Reaction/Enzymes

```
<Enzymes>
  <Protein name = "PR1" EC# ="1.0.0.1"/>
  <RNA name = "RN1"/>
</Enzymes>
<Enzymes>
   <RNA name = "RN2">
</Enzymes>
```

Q2:  Reaction/Enzymes[/Protein]

Q2/: Enzymes[/Protein]

Q3: Reaction/Enzymes/Protein

Q3/: Enzymes/Protein

Compensation Queries

Q4:  /Reaction[@name = “RN2”]/Enzymes ?

# Motivation

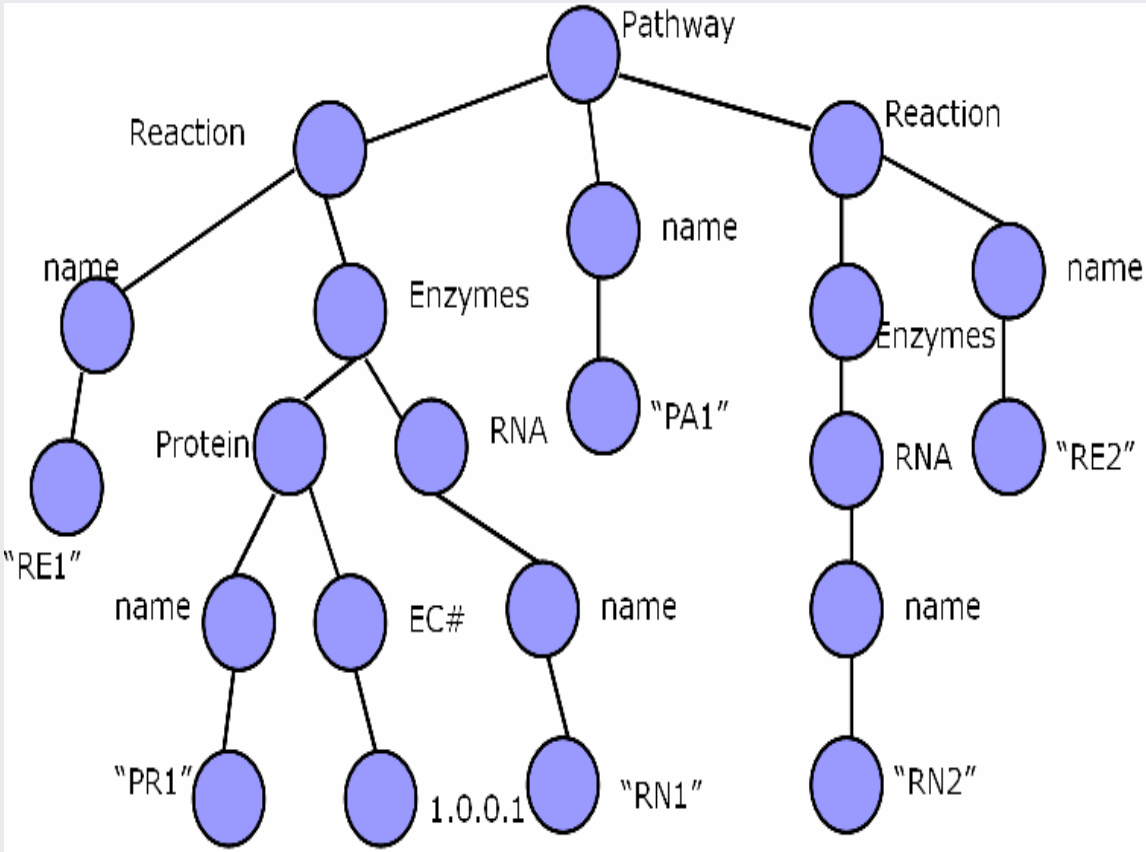
Given an XML database, and a cached query result.

Question: How to answer a new query by using the cached result?

- Is there a compensation query?
- If it exists, then what is the best compensation query?
  - Depends on many factors: # of descendant axes, wildcards, and so on.
  - Only size is considered.

# XML Documents -> Unordered rooted labeled trees

```
<Pathway name =  "PA1">
  <Reaction name = "RE1">
    <Enzymes>
      <Protein name =
    "PR1“ EC# =“1.0.0.1"/>
      <RNA name =
    "RN1"/>
    </Enzymes>
  </Reaction>
  <Reaction>
    <Enzymes>
      <RNA name = "RN2">
    </Enzymes>
  </Reaction>
</Pathway>
```



# XPath Queries -> Tree Patterns

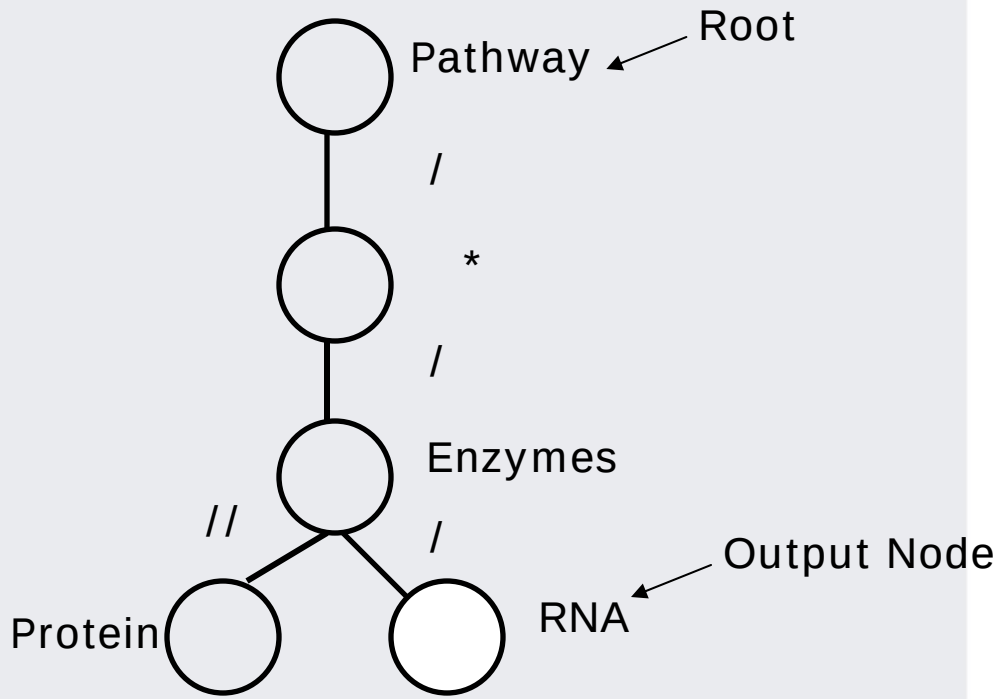
## XPath Fragment $XP^{\{/, [\,], *, //\}}$

- Node tests
- Child axes (/)
- Descendant axes(//)
- Wildcards (*)
- Predicates ([…])

**It's three subclasses**

- $XP^{\{/, [\,], //\}}$
- $XP^{\{/, *, //\}}$
- $XP^{\{/, [\,], *\}}$

Pathway/*/Enzymes[//Protein]/RNA



# Embeddings

XML tree: $t(V_t,E_t,r_t)$
Tree pattern: $p(V_p,E_p,r_p,o_p)$

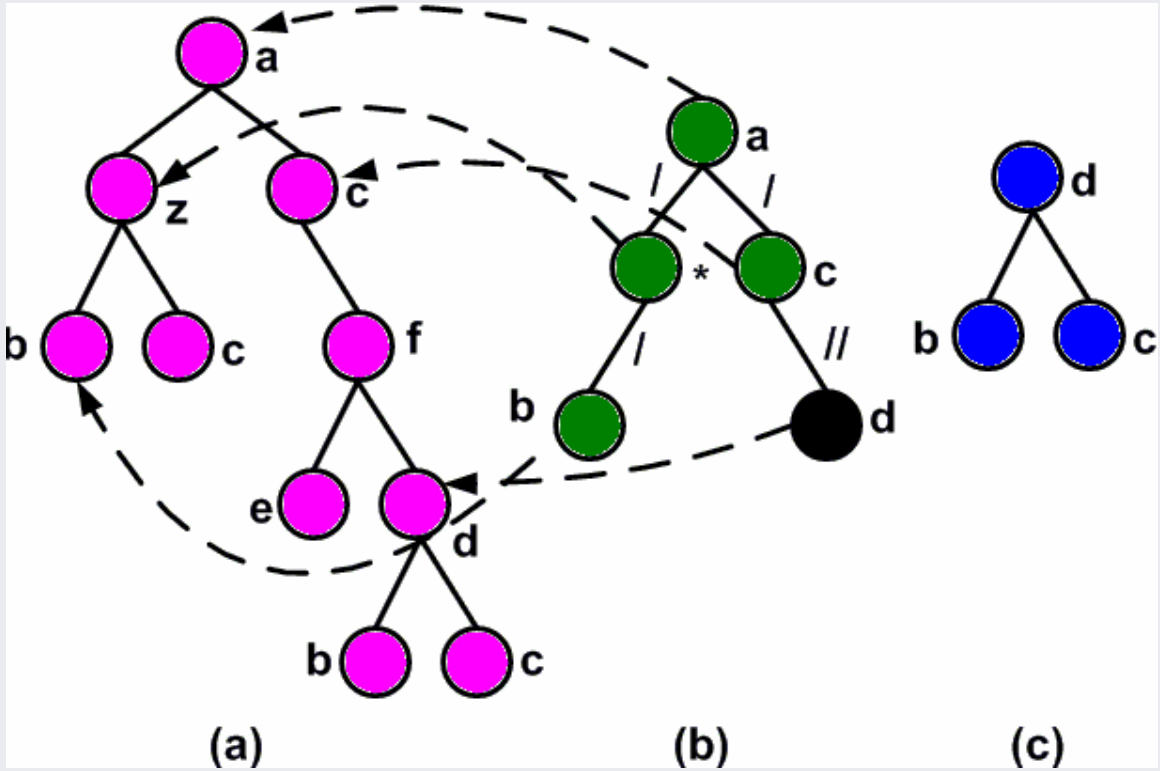
Embedding: e: $V_p \rightarrow V_t$

Root preserving
Label preserving
Structure preserving

Let $n=e(o^p)$,
then subtree with root n of t
$(t)^{e(op)}$ is the result of
embedding.

Result
$p(t)=$ union $\{(t)^{e(op)}\}$

for all embeddings from p to t.



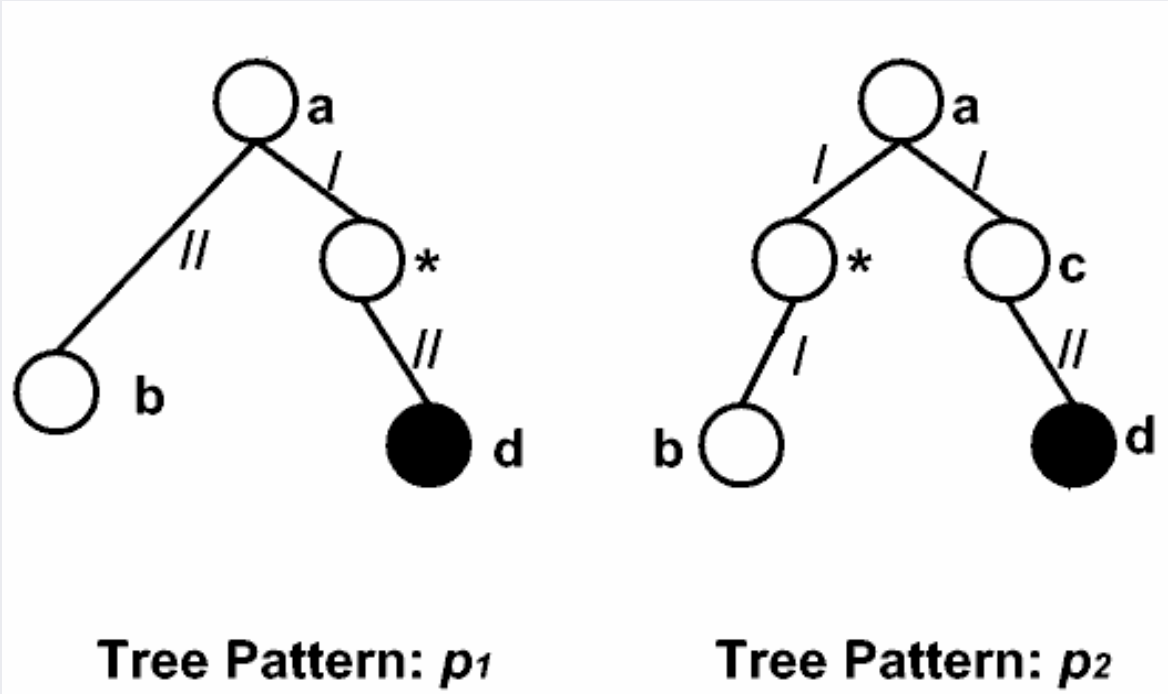

XML tree t | Pattern p | result

for all embeddings from p to t.

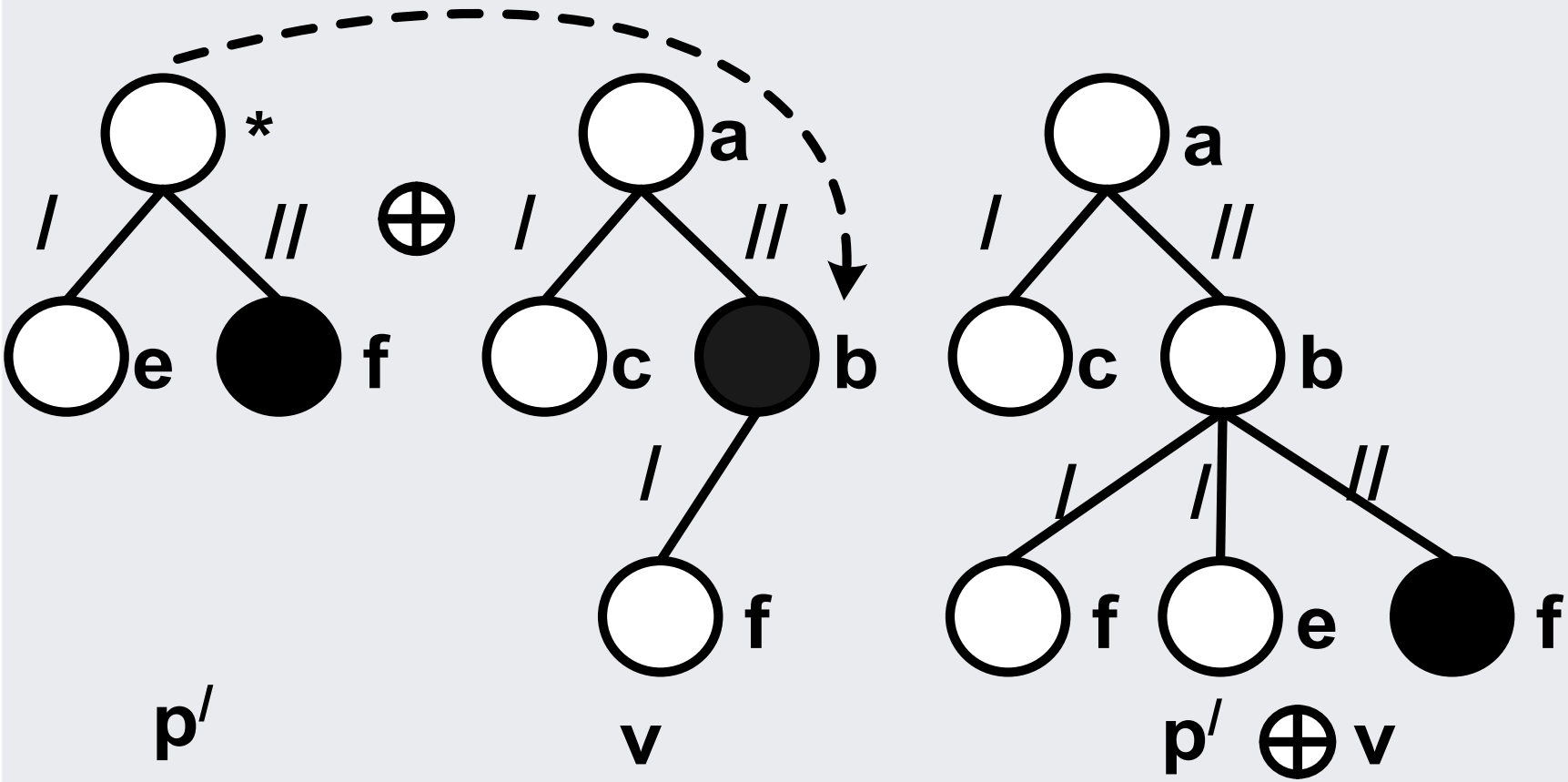
# Containment and Equivalence of Tree Patterns

$P_2$ is contained in $P_1$ if $P_2(t) \subseteq P_1(t)$ for any XML tree $t$

$P_2 \equiv P_1$ if $P_2$ is contained in $P_1$ and $P_1$ is contained in $P_2$

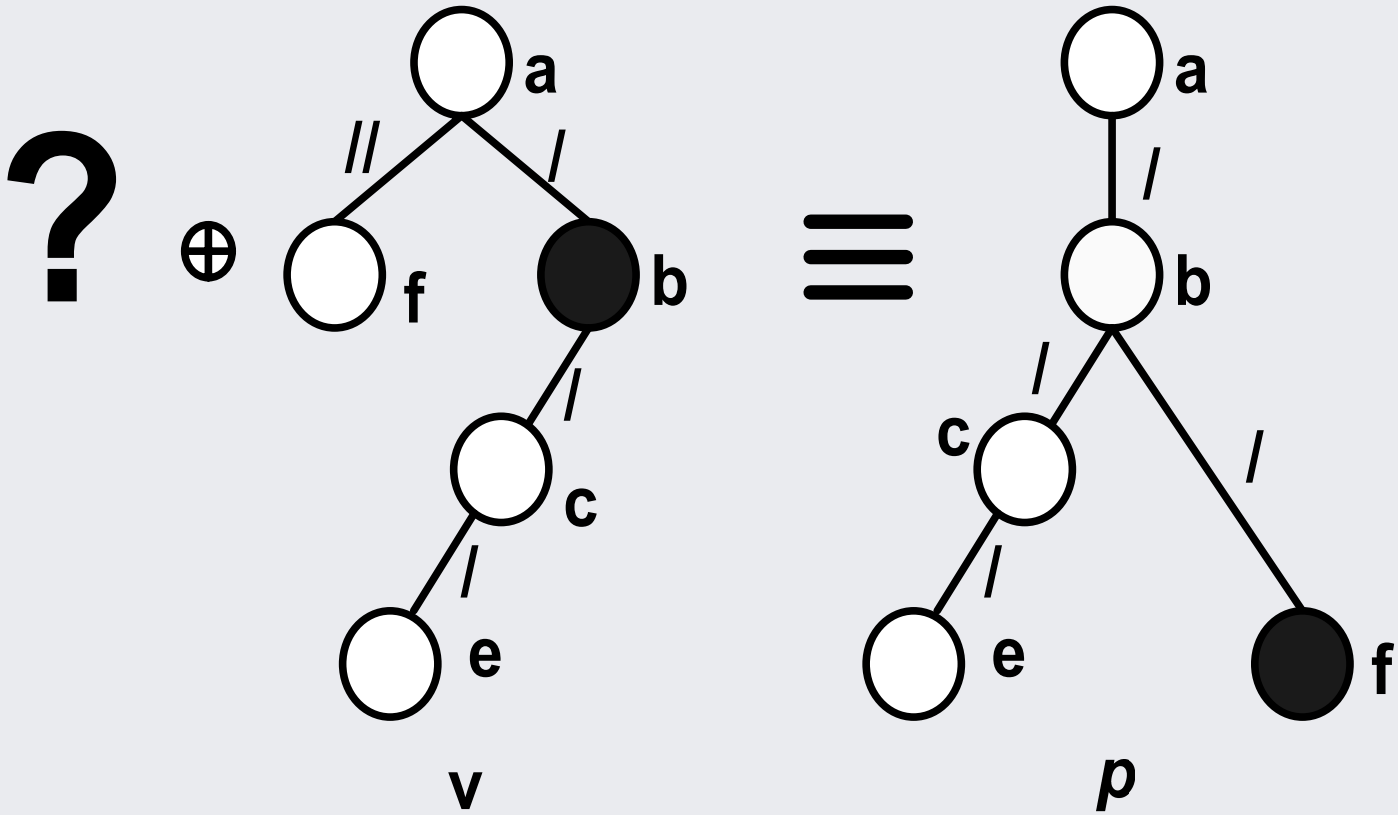


# Concatenation of Patterns




The XPath fragment and its three subclasses are close under concatenation

# Rewriting Existence Problem
# Finding Minimal Rewritings Problem



# Rewriting Existence Problem

Rewriting existence problem is equivalent to the containment problem of patterns.

- Let *p* and *v* be two patterns with output nodes as roots. *p* is contained *v* iff there exists a rewriting of *p* using *v*

Rewriting existence problem is coNP-hard for XP{/, //,*,[]}

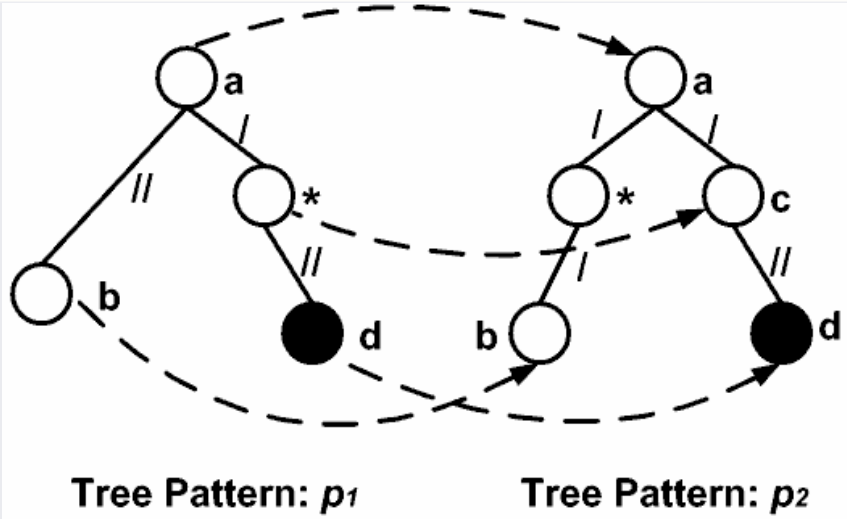
- The containment problem is coNP-complete for XP{/, //,*,[]}

Is the rewriting existence problem for three subclasses of XP{/, //,*,[]} still in P?

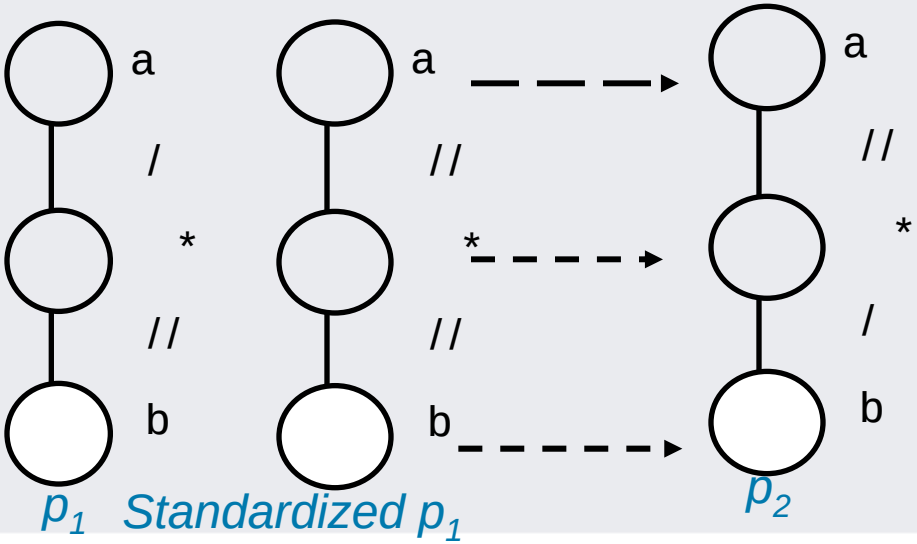
- The containment problem is P for the three subclasses.

# Homomorphisms

For XP{/,//,[]} and XP{/,*,[]},
if $p_2$ is contained in $p_1$, then
a homomorphism exists from $p_1$ to $p_2$




For XP{/,//,*},
if $p_2$ is contained in $p_1$ and
$p_1$ is standardized, then
a homomorphism exists from $p_1$ to $p_2$



# Rewriting Existence

Theorem:

For three subclasses of $XP^{\{/, [\,], *, //\}}$, the Rewriting Existence problem is in P.

Basic idea:

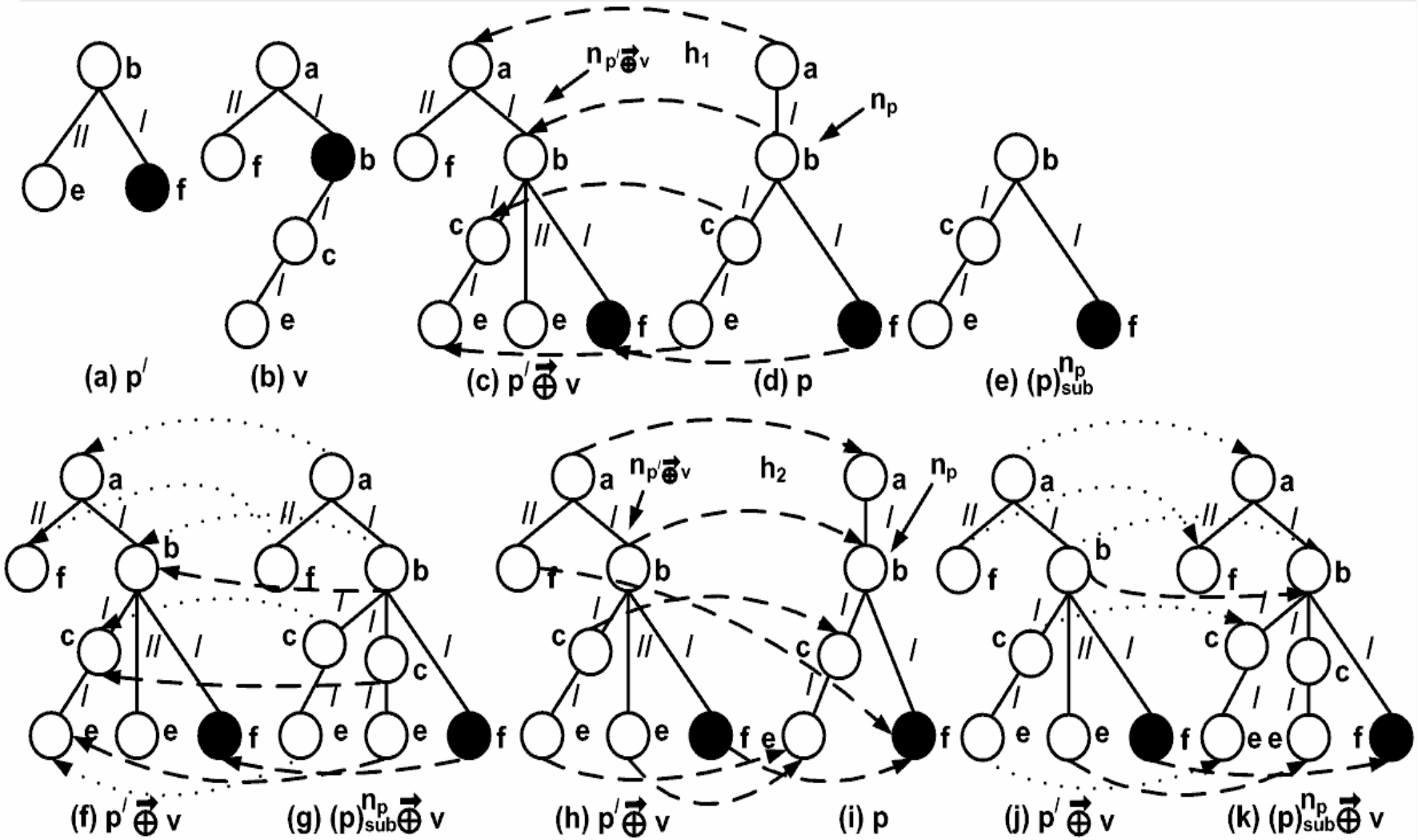
Given two patterns p and v,

If there exists a compensation pattern p' of p using v then there is a node $n_p$ in p such that

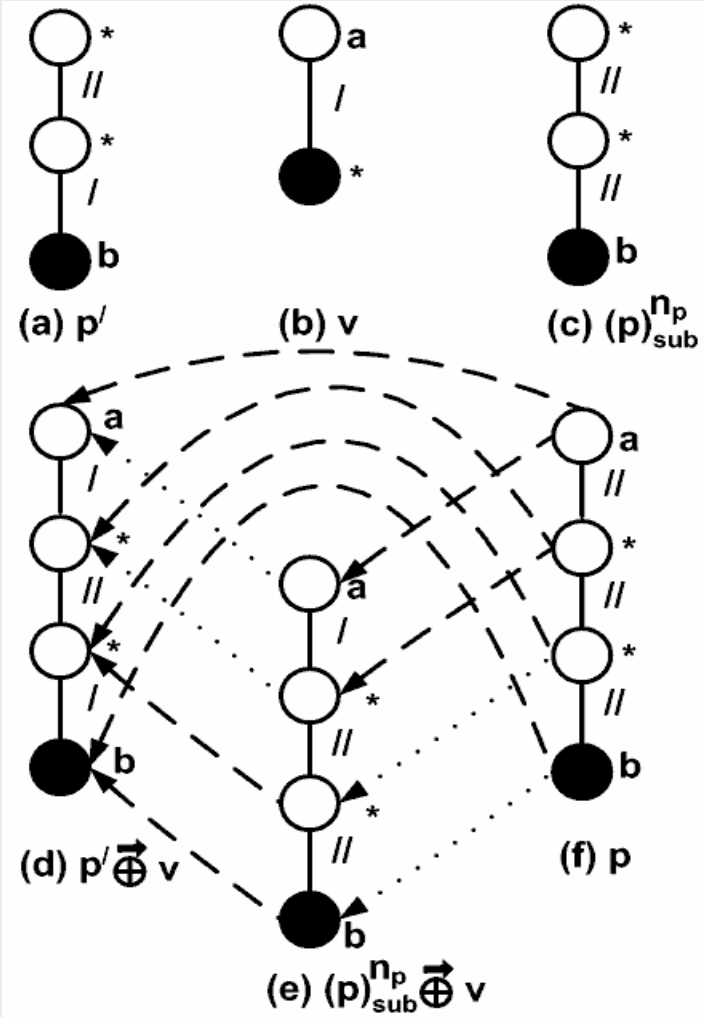
subtree of p with root $n_p$ is also a compensation pattern of p using v.

For three subclasses, only one subpattern of *p* need to be considered for the existence of rewriting of *p* using *v*.

# XP{/,//,[]} and XP{/,*,[]}


(a) p/
(b) v
(c) p/ ⊕ v
(d) p
(e) (p)$^{n_p}_{sub}$
(f) p/ ⊕ v
(g) (p)$^{n_p}_{sub}$ ⊕ v
(h) p/ ⊕ v
(i) p
(j) p/ ⊕ v
(k) (p)$^{n_p}_{sub}$ ⊕ v
$n_{p/\oplus v}$
$h_1$
$h_2$
$n_p$

# XP{/,//,*}



# Summary

| Language | Complexity |
|---|---|
| XP{/, //,*,[]} | coNP-Hard |
| XP{/, //, []} | P |
| XP{/,*,[]} | P |
| XP{/, //,*} | P |

# Finding Minimal Rewritings Problem

Minimization of Patterns

The Complexity for XP{/, //,*,[]}

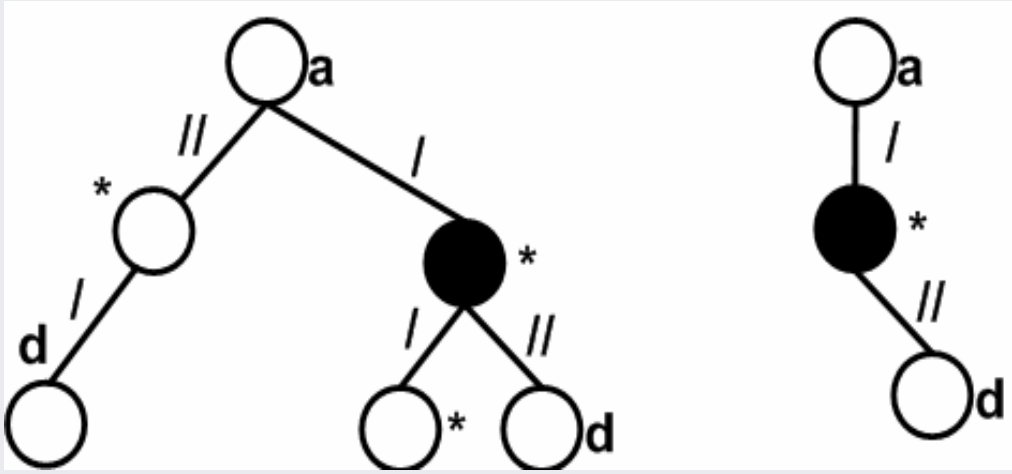
The Complexity for three subclasses of XP{/, //,*,[]}

- A special case—output node is the root
- The general case.

# Minimization of Tree Patterns

Minimizing a pattern *p*: Find an equivalent pattern with minimum size.

- Pruning all redundant subpatterns.




$p_1$ $p_2$

# Minimal Rewriting for XP{*/*, *//*,*,[]}

Let *p* and *v* be two patterns, $p'$ is the minimal compensation pattern of *p* using *v*, i.e., $p' \oplus v \equiv p$

- Observation 1: $p'$ doesn't introduce new label.
- Observation 2: $p'$ doesn't increase the size, i.e., the size of $p'$ is less than the size of *p*.

The problem of whether there exists a compensation pattern $p'$ of *p* using *v* such that $p'$ has size less than k is $\Sigma_3^p$, where $k < \|p\|$.

- Guess in polynomial time a pattern $p'$, which doesn't introduce new label, and has size less than k.
- Check whether $p' \oplus v \equiv p$ is in coNP.

# The three subclasses of XP{/, //,*,[]}

The minimization for three subclasses of XP{/, //,*,[]} is in P.[Wood, WebDB'01, Amer-Yahia, et. al., SIGMOD'02 ]

Is the finding minimal rewritings problem for three subclasses of XP{/, //,*,[]} still in P?

- Any rewriting is minimal for the XP{/, //,*} subclass.
- Yes, for XP{/, //, []} and XP{/,*,[]} subclasses, our result is based on pruning rewriting-redundant nodes.

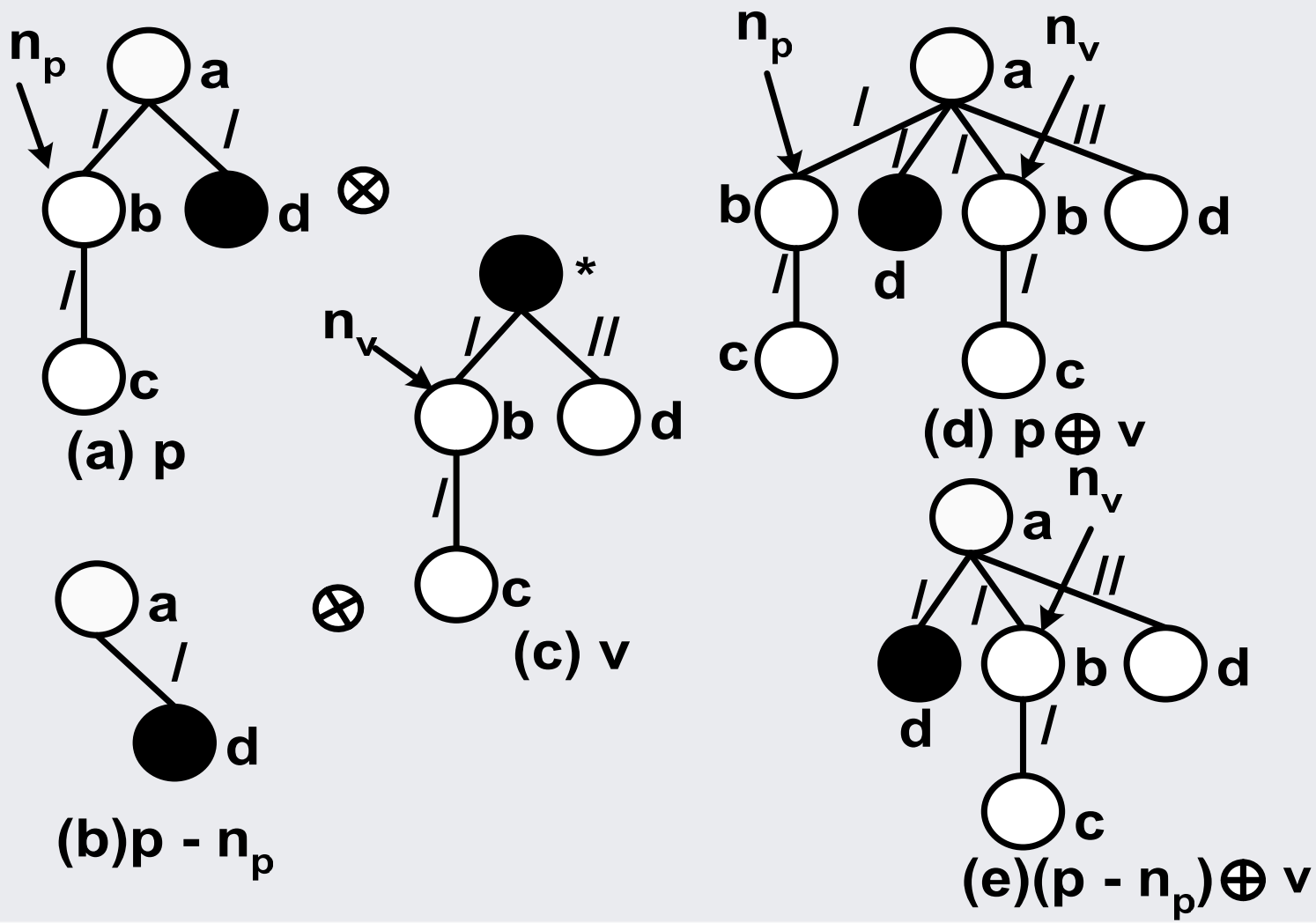
# A Special Case

Let *p* be a pattern, and

the view v be a special pattern whose output node is its root.

- If a compensation pattern of *p* using *v* exists, then *p* is a compensation pattern.
- The minimal compensation pattern can be found among subpatterns of *p*.

# An Example



# The General Case

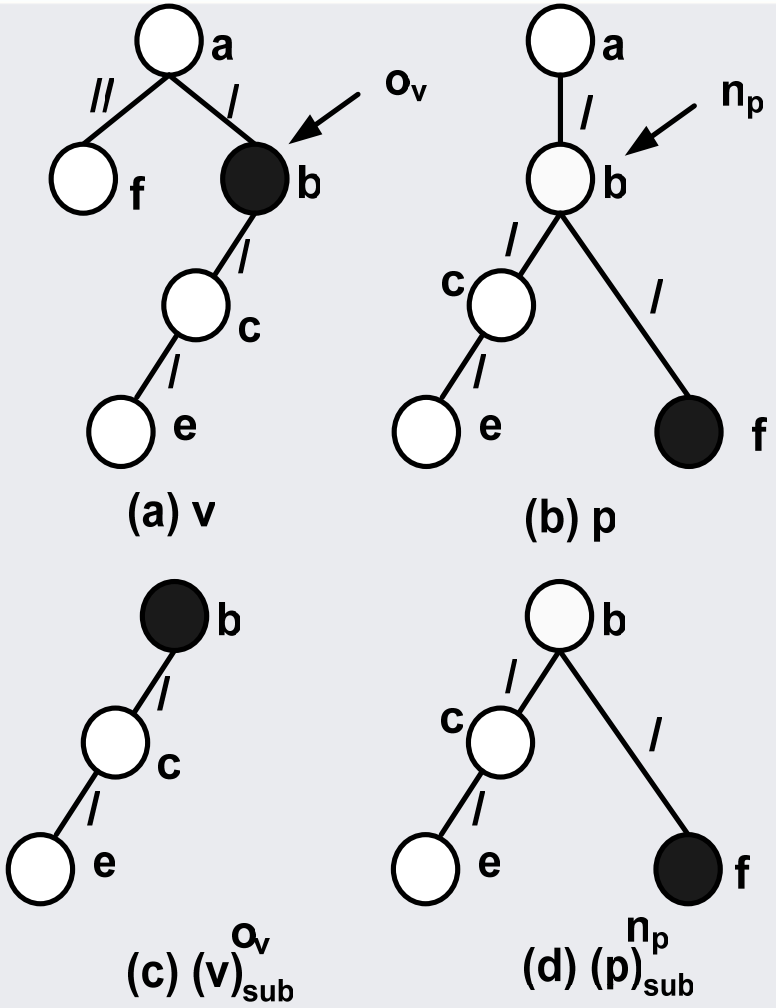
For XP{/, //, []} and XP{/,*,[]} subclasses, the general case of the problem of finding minimal rewritings can be reduced to the special case of the finding minimal rewritings problem.

# An Example

If $(p)_{sub}^{n_p}$ is a compensation pattern of *p* using *v*, then it is a also compensation pattern of itself using $(v)_{sub}^{o_v}$.

Then, the minimal compensation pattern of $(p)_{sub}^{n_p}$ using $(v)_{sub}^{o_v}$ is a also minimal compensation pattern of *p* using *v*.

The general case of finding minimal rewritings problems can be reduced to its special case



**Algorithm 1: Finding minimal rewritings**

**Input:** $p$ and $v$(two patterns)

**Output:** a minimal compensation pattern of $p$ using $v$ if exists; otherwise *null*.

$p \leftarrow min(p)$;
$v \leftarrow min(v)$;
Let $o_v$ be the output node of $v$;
Let $n^p$ be a node in $p$ has the same position to $o_v$;
**if** $(p)_{sub}^{n_p} \overrightarrow{\oplus} v \not\equiv p$ **then**
  return *null*;
**end if**
$R_v^p \leftarrow \Phi$;
$p' \leftarrow (p)_{sub}^{n_p}$;
$v' \leftarrow (v)_{sub}^{o_v}$;
**for** Each $n_{p'} \in C_{p'}$ **do**
  **for** Each $n_{v'} \in C_{v'}$ **do**
    **if** $(p' \overrightarrow{\oplus} v')^{n_{p'}} \equiv (p' \overrightarrow{\oplus} v')^{n_{v'}}$ **then**
      $R_{v'}^{p'} = R_{v'}^{p'} \cup \{n_{p'}\}$;
    **end if**
  **end for**
**end for**
return $p' - R_{v'}^{p'}$;

# The general case

The reduction doesn't work for the whole fragment XP{/, //,*,[]}, since the fact, $(p)_{sub}^{n_p}$ is not a compensation pattern of *p* using *v*, doesn't imply no compensation pattern of *p* using *v* exists.

The above algorithm is still sound but potentially runs in exponential time for XP{/, //,*,[]}.

# Conclusions

## Two problems

- The rewriting existence problem.
- The finding minimal rewritings problem.

## The rewriting existence problem

- coNP-hard for XP{/, //,*,[]}.
- P for the three subclasses of XP{/, //,*,[]}.

## The finding minimal rewritings problem

- $\Sigma_3^p$ for XP{/, //,*,[]}.
- P for the three subclasses of XP{/, //,*,[]}.

# Future Work

Investigating those two rewriting problems in presence of DTDs, or other constraints.

Developing an index for materialized patterns: by using this index, for a new pattern *p*, a materialized pattern *v* can be efficiently found such that there exists a compensation pattern of *p* using *v*.

Developing algorithms on selecting a set of patterns to materialize for obtaining optimal performance with a given query workload.

# Thank you