Node Query Preservation for Deterministic Linear Top-Down Tree Transducers

Kazuki Miyahara[1] Kenji Hashimoto[2] Hiroyuki Seki[1,2]

[1]Nara Institute of Science and Technology, Japan

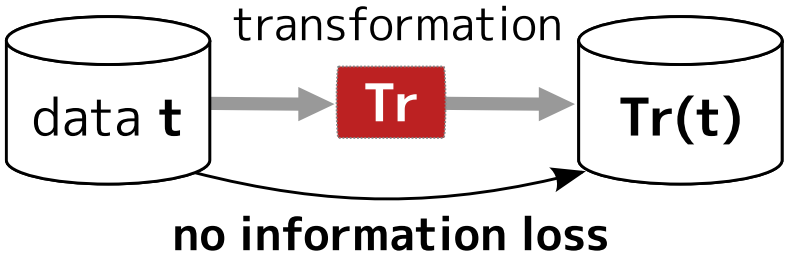
[2]Nagoya University, Japan

TTATT, Hanoi, October 19, 2013

Query Preservation

Lossless data transformation

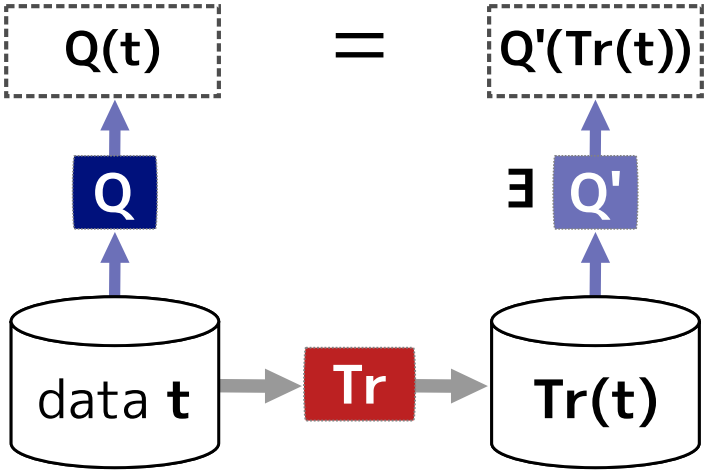
- Information in the data source is preserved through data transformation.



Query preservation

$$\exists Q'.\ \forall t.\ \ Q(t) = Q'(Tr(t))$$

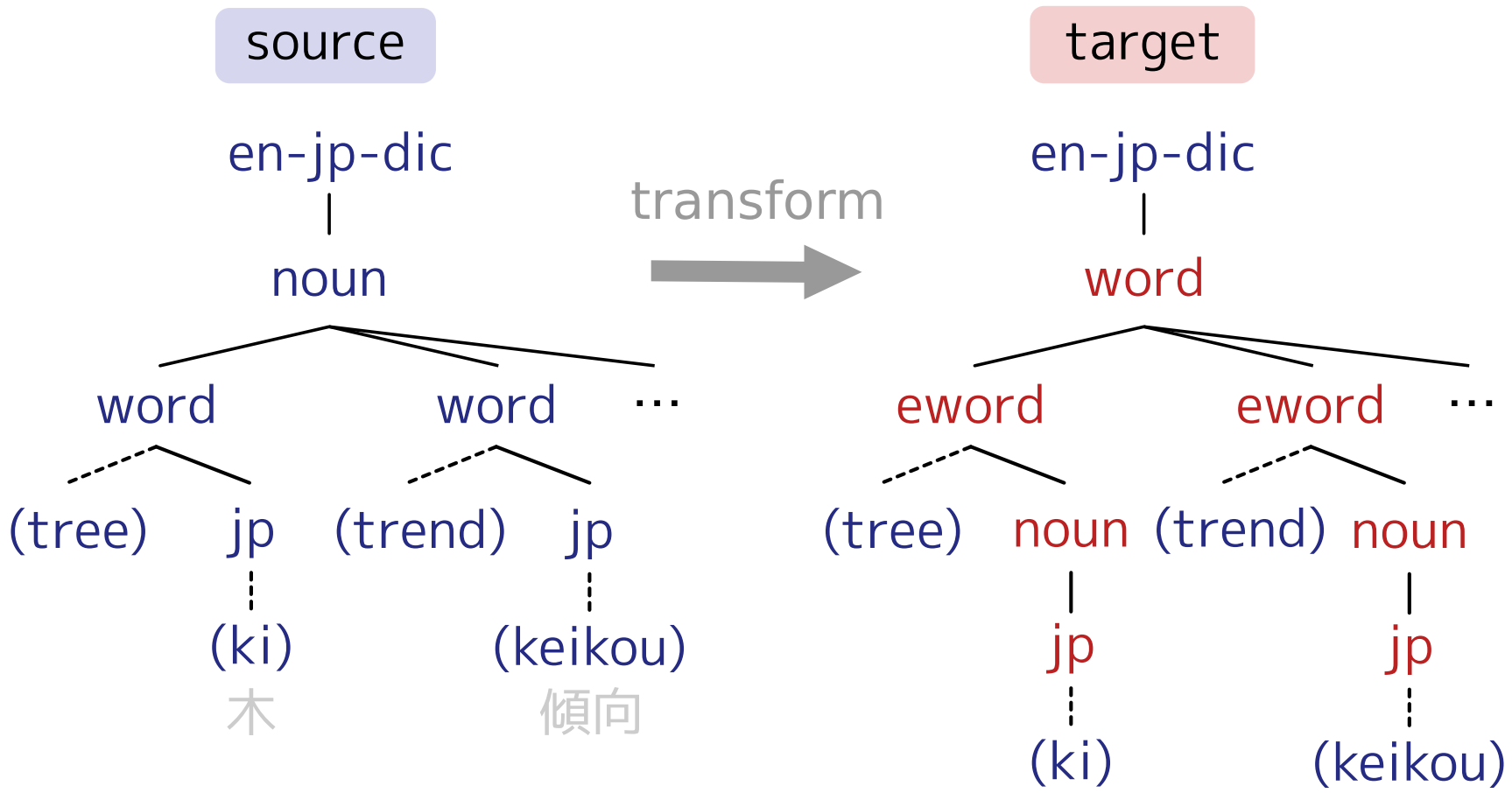
- We focus on the query preservation for data tree transformations.



Data Tree Transformation

Data tree is a data whose nodes have a data value.

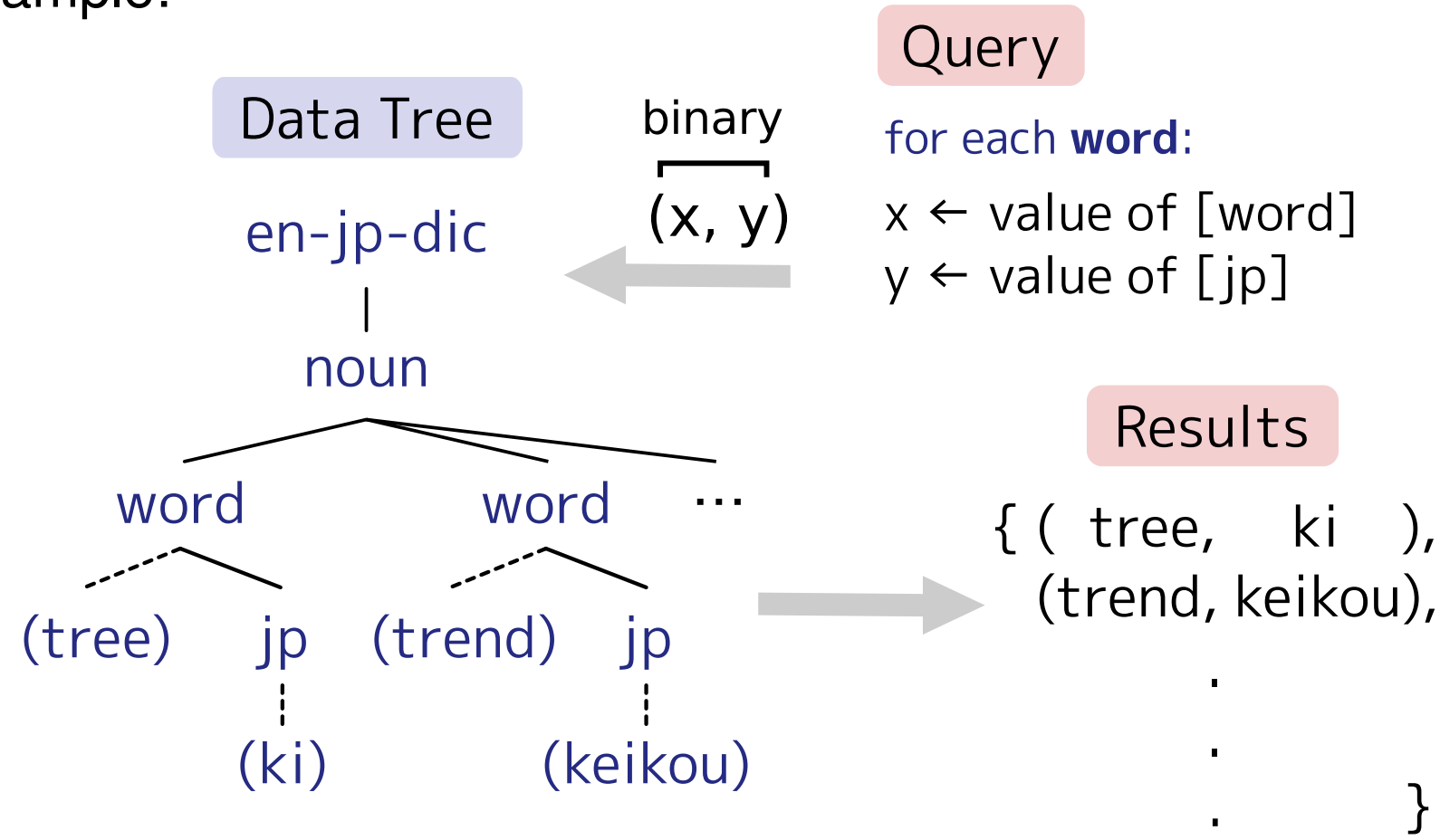
Example: Mapping noun/word/jp to word/eword/noun/jp



Node Query

Mapping data tree to tuples of data values

Example:

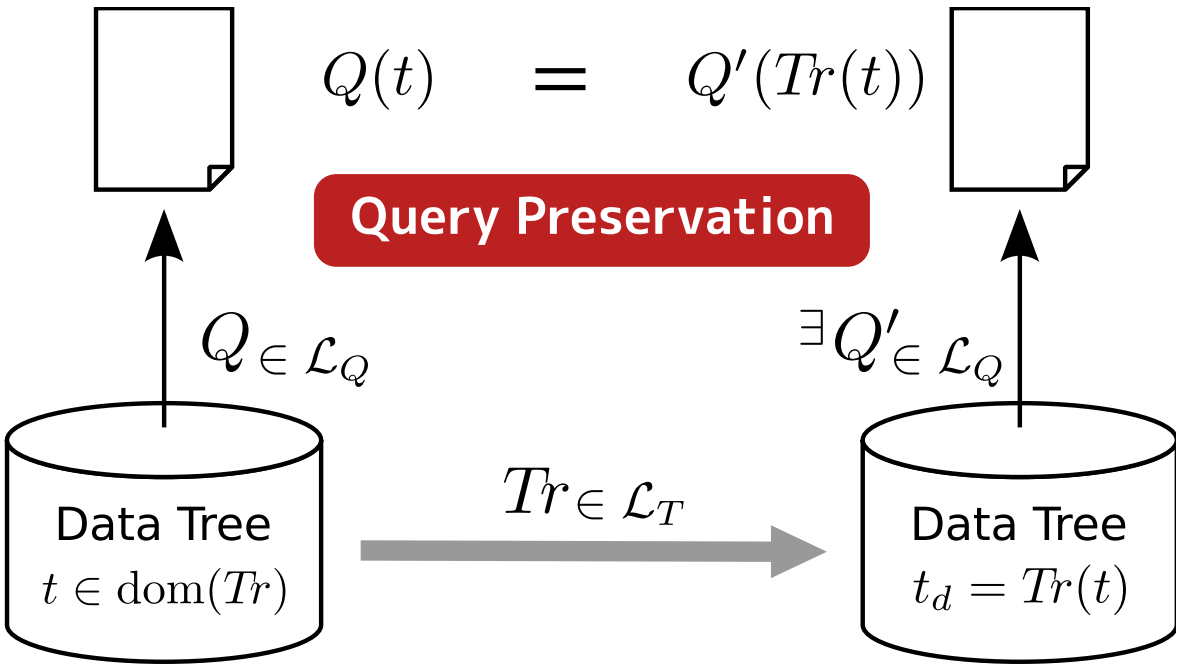


(Strong) Query Preservation

Let $\mathcal{L}_Q$ be a class of queries, $\mathcal{L}_T$ be a class of transformations.
Given a query $Q \in \mathcal{L}_Q$ and a transformation $Tr \in \mathcal{L}_T$,

We say $\boxed{Tr \text{ preserves } Q}$,

if there exists $Q' \in \mathcal{L}_Q$, such that for any tree $t \in \mathbf{dom}(Tr)$,
results of Q' coincide with results of Q.

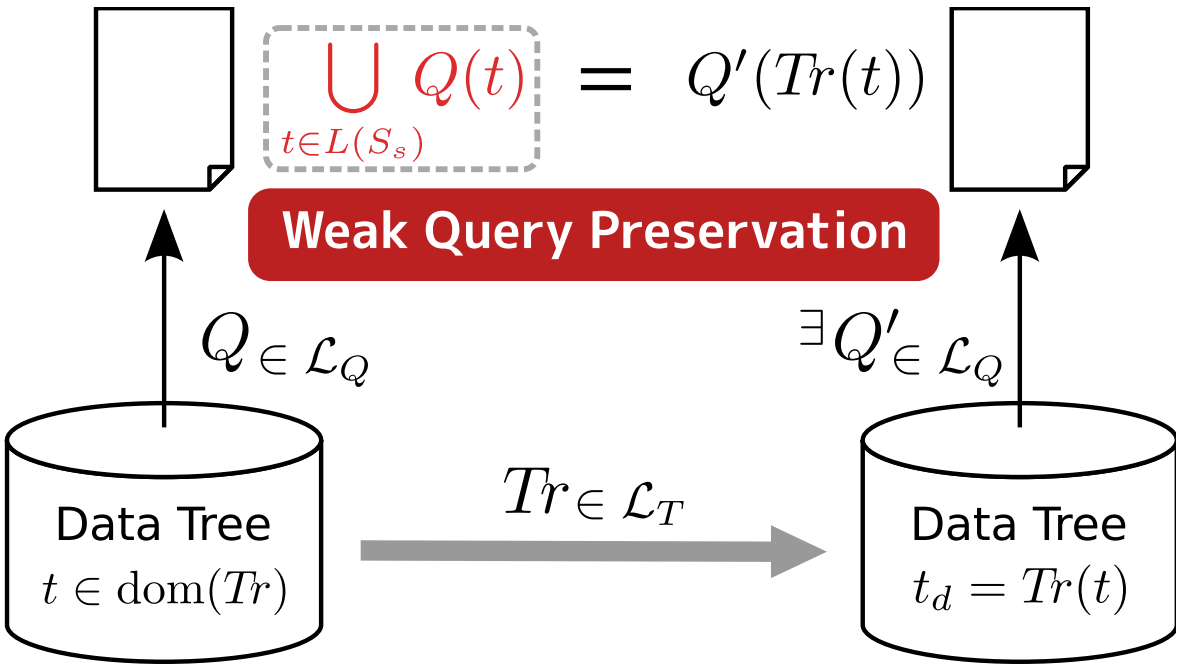


Weak Query Preservation

Let $\mathcal{L}_Q$ be a class of queries, $\mathcal{L}_T$ be a class of transformations.
Given a query $Q \in \mathcal{L}_Q$ and a transformation $Tr \in \mathcal{L}_T$,

We say $\boxed{Tr \text{ weakly preserves } Q}$,

if there exists $Q' \in \mathcal{L}_Q$, such that for any tree $t \in \mathbf{dom}(Tr)$,
results of Q' coincide with **the union of** the results of Q.



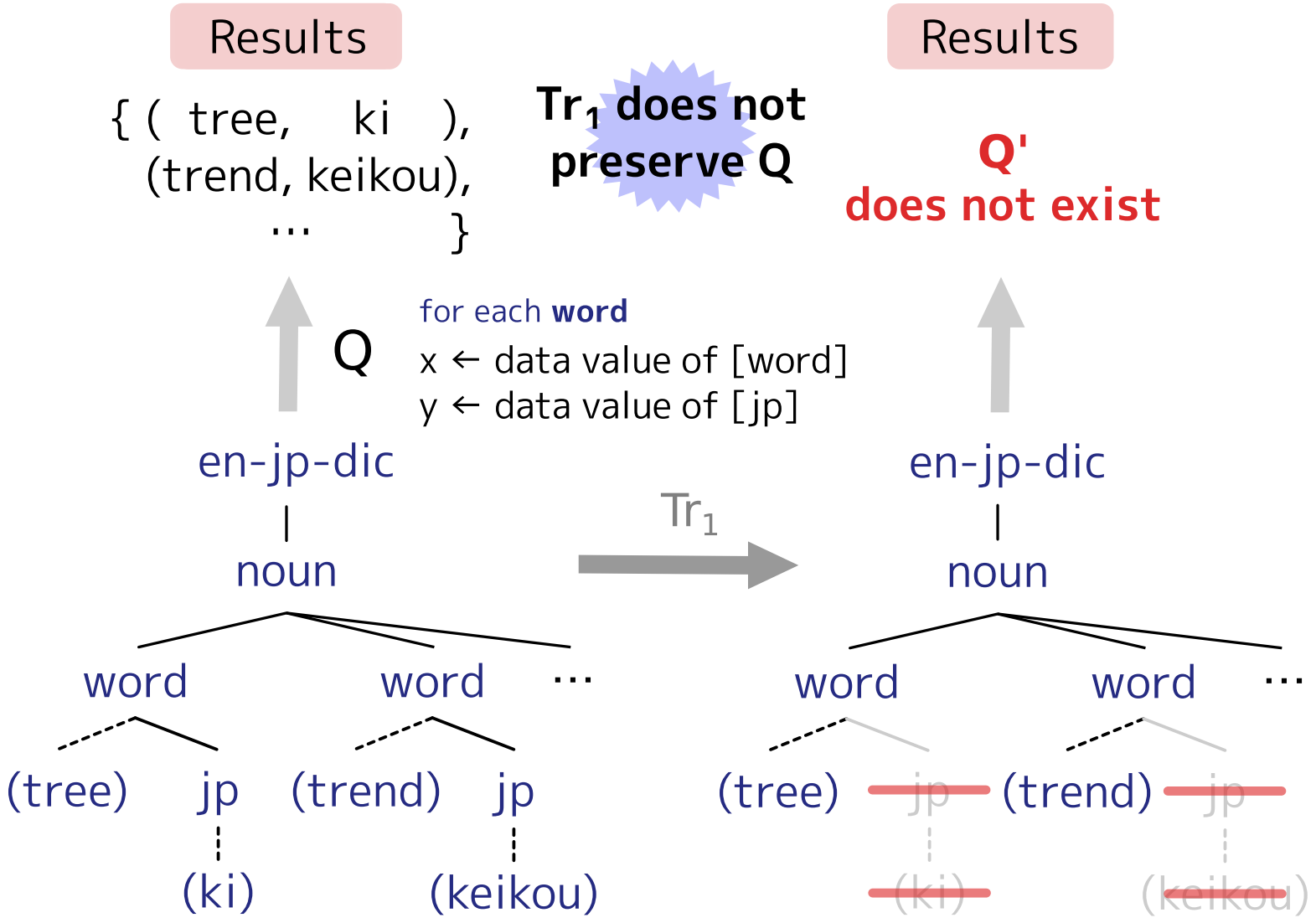

Results
{ (tree, ki),
(trend, keikou),
… }
Tr1 does not preserve Q
Results
Q'
does not exist
Q
for each word
x ← data value of [word]
y ← data value of [jp]
en-jp-dic
noun
word
word
…
(tree)
jp
(trend)
jp
(ki)
(keikou)
Tr1
en-jp-dic
noun
word
word
…
(tree)
jp
(trend)
jp
(ki)
(keikou)

Table of Contents

Our Results

We obtain the results 1. and 2. for the following class:

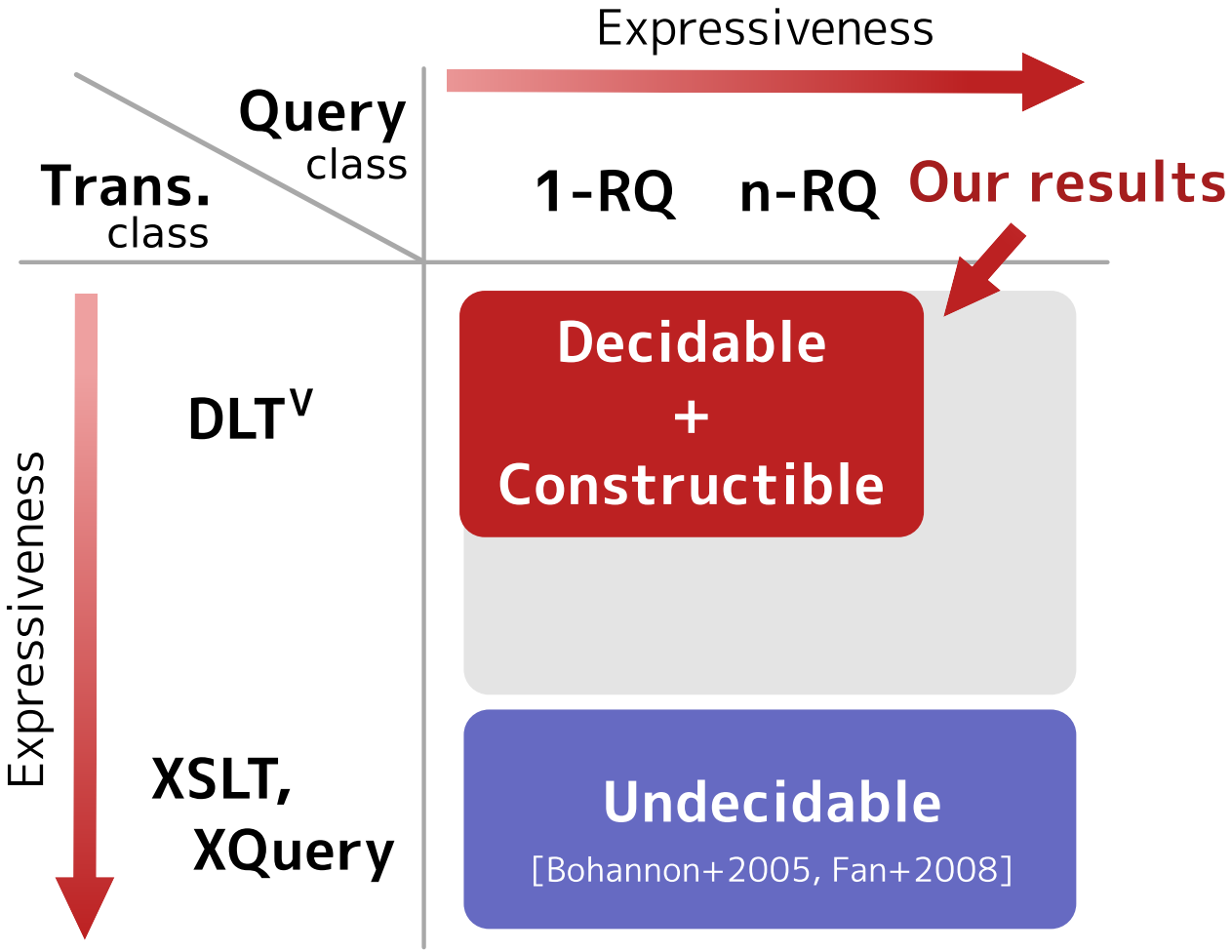
Trans. Deterministic Linear Top-down Data Tree Transducer ($\mathbf{DLT}^V$)

Query Run-based n-ary Query (n-$\mathbf{RQ}$)

1. Query preservation problem is decidable
2. Query on transformed tree can be constructed

Our Results and Related Work

Decidability of Query Preservation



Our Results and Related Work

[Our results]

- DLTT^V as transformations
- Run-based ***n*-ary** (= MSO n-ary) queries (node queries)

[Benedikt+13]

- (fu-ELT^R)* as transformations
- DMSOTT (tree queries)

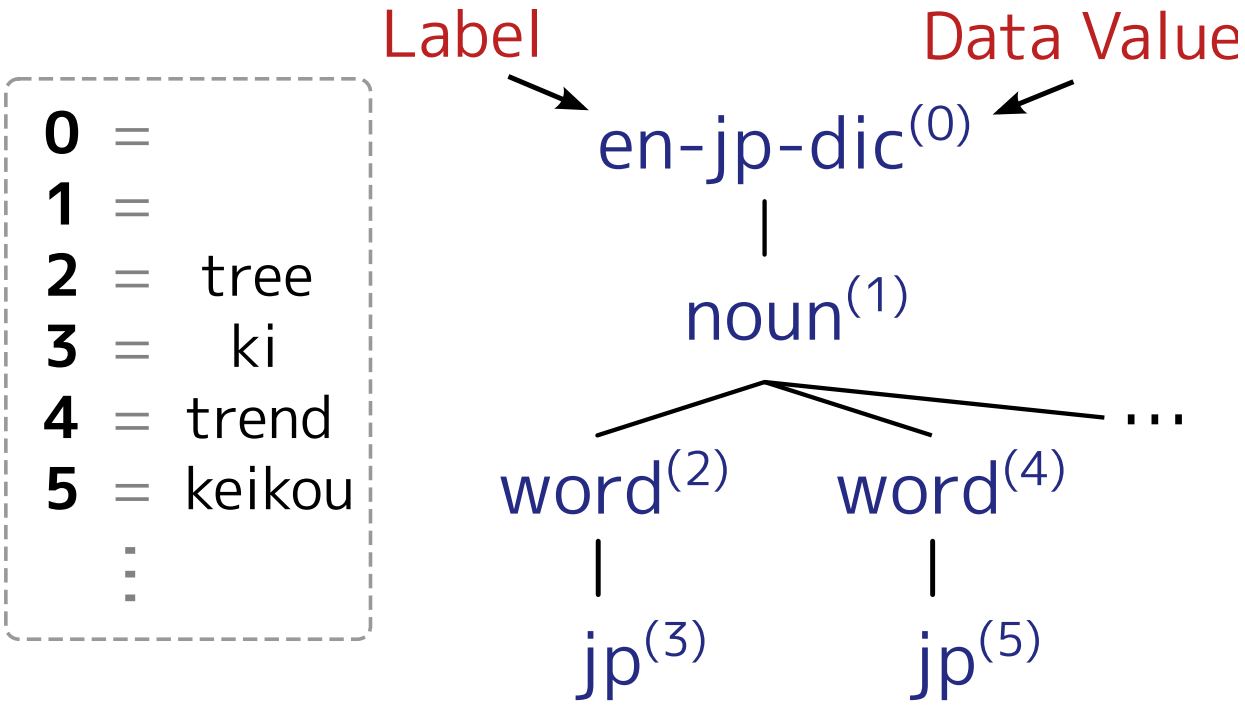
[Groz+13]

- Selection of subtrees using Regular XPath filterings as transformations and queries on **unranked** trees

Data Tree

Data Tree: Labeled ordered ranked tree such that each node can have a data value

Data Value: Nonnegative integer



Deterministic Linear Top-down **Data** Tree Transducer

Definition: $\mathbf{DLT}^V\ Tr = (P, \Sigma, \Delta, P_I, \delta)$

P: Finite set of states, Σ: Input alphabet,
Δ: Output alphabet, $P_I \subseteq P$: Finite set of initial states,
δ: Finite set of transduction rules.

Transduction rules have the form:

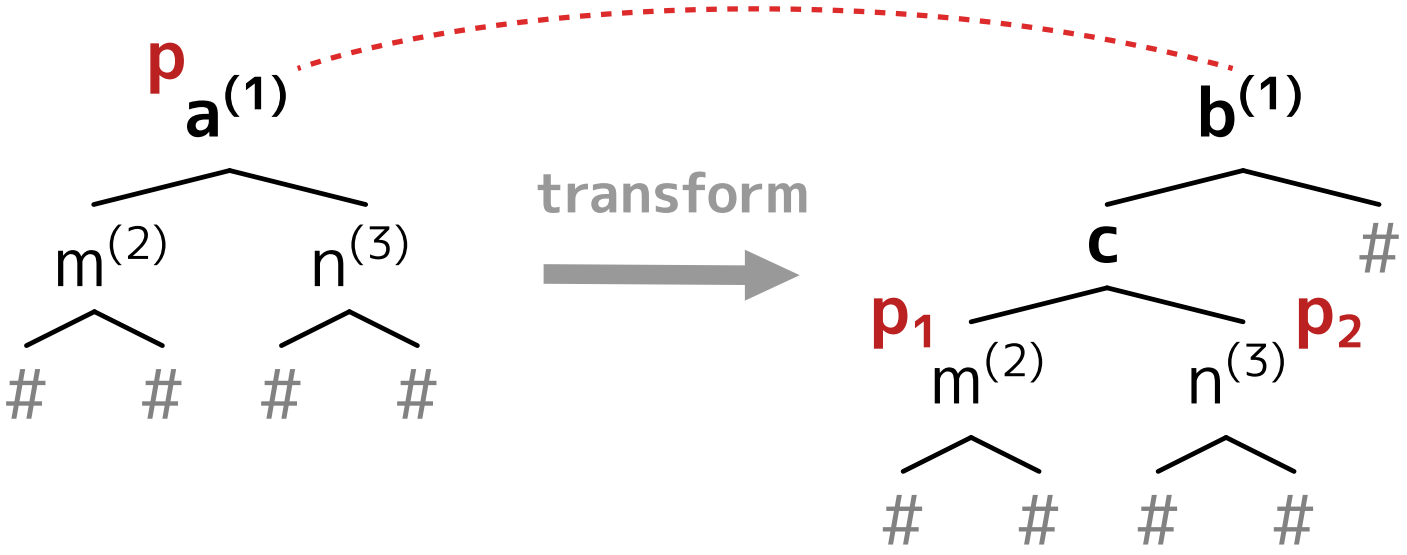
- $p(\sigma^{(z)}(x_1, \ldots, x_n)) \rightarrow C^{(j \leftarrow z)}[p_1(x_1), \ldots, p_n(x_n)]$

where $\sigma \in \Sigma,\ p, p_1, \ldots, p_n \in P, C \in \mathsf{C}(\Delta, X_n),\ t' \in \mathcal{T}_\Delta$,
z is a variable for data value,
j is a position of the context C to which the value z is transferred, and $(j \leftarrow z)$ stands for copying the data value of σ to a node on position j of the context C.

Example of $\mathbf{DLT}^V$

Let $p, p_1, p_2 \in P,\ p \in P_I,\ a, m, n, \# \in \Sigma,\ b, c, \# \in \Delta$,
Context $C^{(\varepsilon \leftarrow z)} = b^{(z)}(c(y_1, y_2), \#)$.

$$p(a^{(z)}(x_1, x_2)) \to b^{(z)}(c(p_1(x_1), p_2(x_2)), \#) \in \delta$$



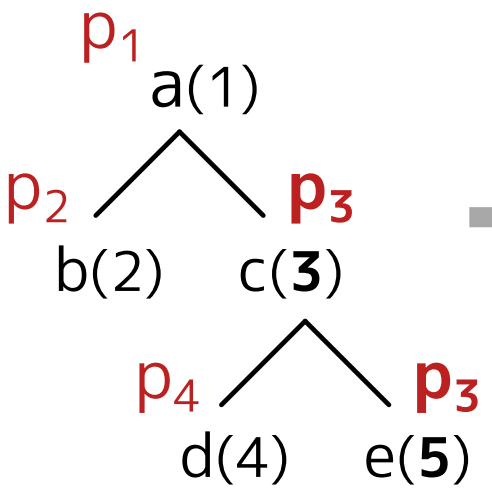
Run-based n-ary Query

n-RQ : Run-based n-ary query [Niehren+05]

Definition: **n-RQ $Q = (A, S)$**

$A = (P, \Sigma, P_I, \delta)$ is a TA, S is a set of n-tuple of states ($S \subseteq P^n$).

Result of
accepting run by A



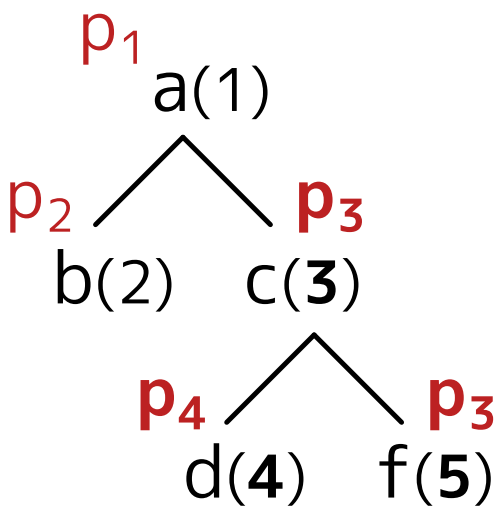
Unary

S = { p_3 }

Result

{ **3**, **5** }

Examples of Run-based Query



Unary

$S = \{ p_3, p_4 \}$

Results

$\{ \mathbf{3, 4, 5} \}$

$\underbrace{\{3,5\}}_{p_3} \cup \underbrace{\{4\}}_{p_4}$

Binary

$S = \{ (p_3, p_4) \}$

$\{ \mathbf{(3, 4), (5, 4)} \}$

$\underbrace{\{3,5\}}_{p_3} \times \underbrace{\{4\}}_{p_4}$

Strong and Weak Query Preservation

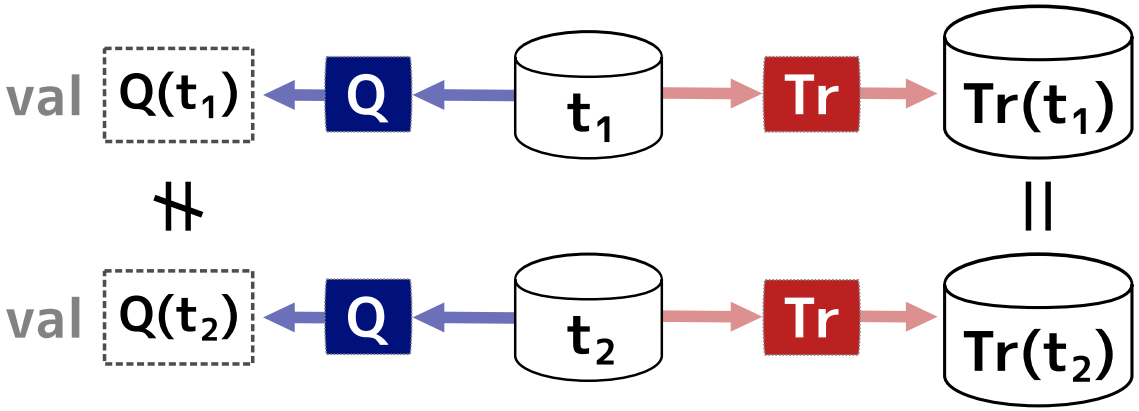
Tr preserves Q

$\Updownarrow$

Tr weakly preserves Q $\wedge$ "**Join**" does not exist

Join:

$\exists t_1, t_2 \in \mathbf{dom}(Tr)$ s.t. $Tr(t_1) = Tr(t_2) \wedge \mathbf{val}(Q(t_1)) \neq \mathbf{val}(Q(t_2))$



Example of "Join"

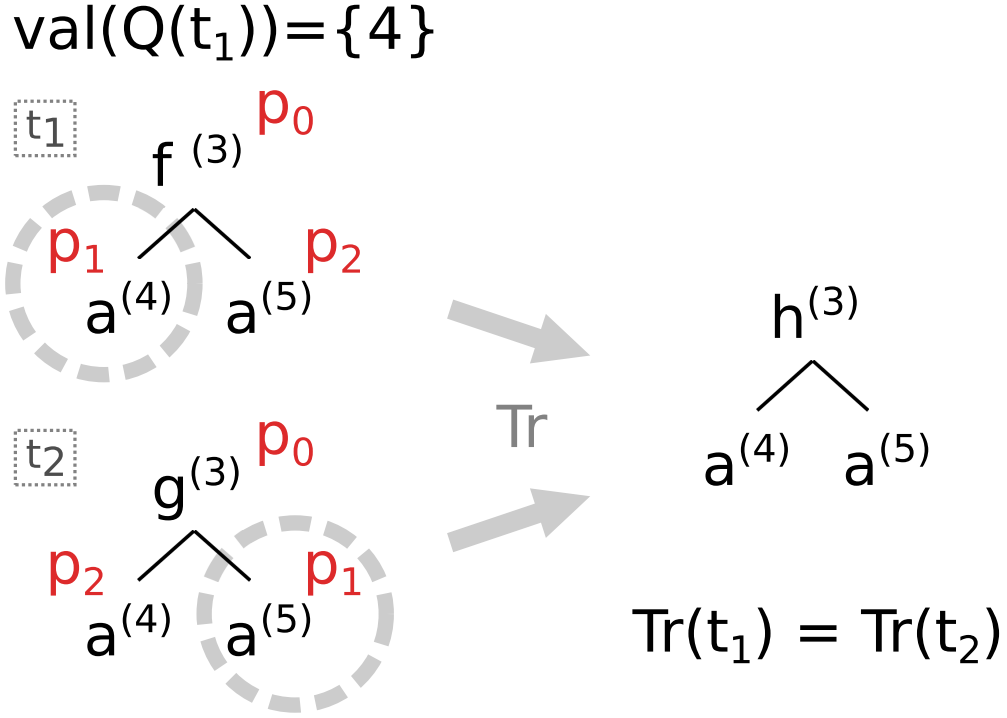
Join:

$\exists t_1, t_2 \in \mathbf{dom}(Tr)$ s.t. $Tr(t_1) = Tr(t_2) \wedge \mathbf{val}(Q(t_1)) \neq \mathbf{val}(Q(t_2))$

Consider the relabeling Tr: $f, g, a \rightarrow h, h, a$

$Q = (A, \{p_1\})$ where $A = (\{p_0, p_1, p_2\}, \{f, g, a\}, \{p_0\}, \delta)$.

$\sigma = \{$
$p_0 \rightarrow f(p_1, p_2),$
$p_0 \rightarrow g(p_2, p_1),$
$p_1 \rightarrow a,$
$p_2 \rightarrow a$
$\}$



Example of "Join"

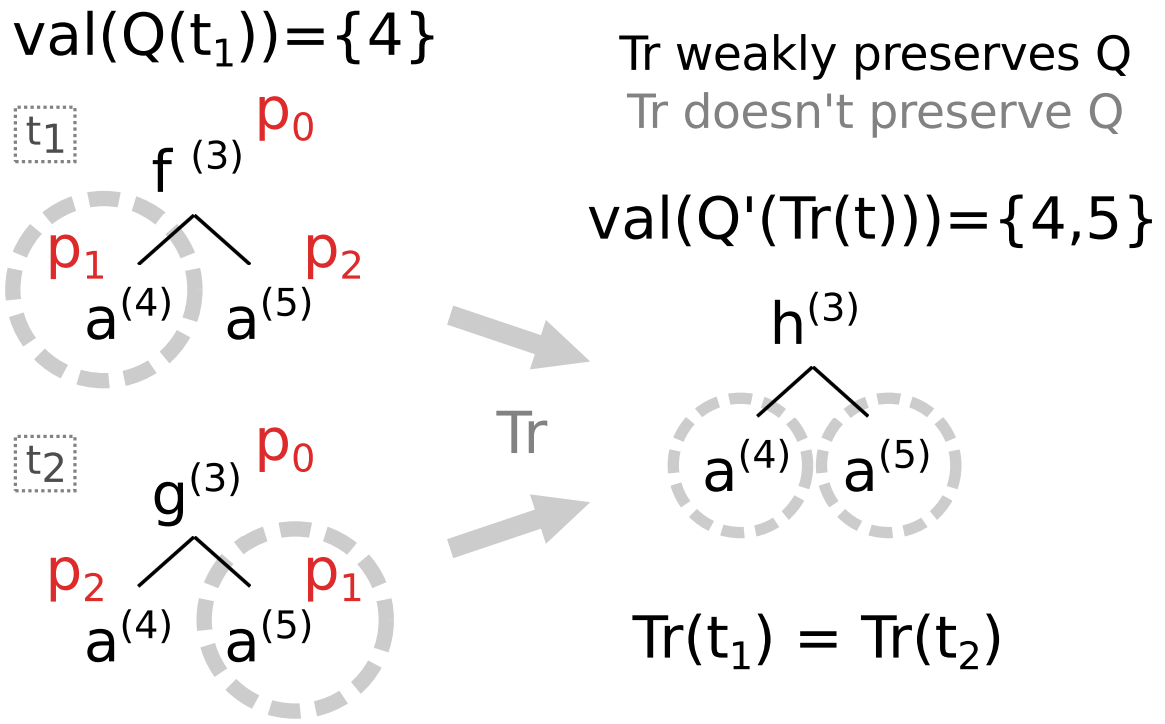
Join:

$\exists t_1, t_2 \in \mathbf{dom}(Tr)$ s.t. $Tr(t_1) = Tr(t_2) \wedge \mathbf{val}(Q(t_1)) \neq \mathbf{val}(Q(t_2))$

Consider the relabeling Tr: $f, g, a \rightarrow h, h, a$

$Q = (A, \{p_1\})$ where $A = (\{p_0, p_1, p_2\}, \{f, g, a\}, \{p_0\}, \delta)$.

$\sigma = \{$
$p_0 \rightarrow h(p_1, p_2)$,
$p_0 \rightarrow h(p_2, p_1)$,
$p_1 \rightarrow a$,
$p_2 \rightarrow a$
$\}$



Our Algorithm for Query Preservation

Input: $\mathbf{DLT}^V$ $\boldsymbol{Tr}$ and $\boldsymbol{n}$-**RQ** $\boldsymbol{Q}$

Output: If $\boldsymbol{Tr}$ preserves $\boldsymbol{Q}$, output "Yes," otherwise "No."

1. Decide whether $\boldsymbol{Tr}$ weakly preserves $\boldsymbol{Q}$
   - **Yes** — Go to 2.
   - **No** — Output "No"
2. Does "Join" exist?
   - **Yes** — Ourput "No"
   - **No** — Output "Yes" ($\boldsymbol{Tr}$ preserves $\boldsymbol{Q}$)

Our Algorithm for Query Preservation

Input: $\mathbf{DLT}^V$ Tr and n-**RQ** Q

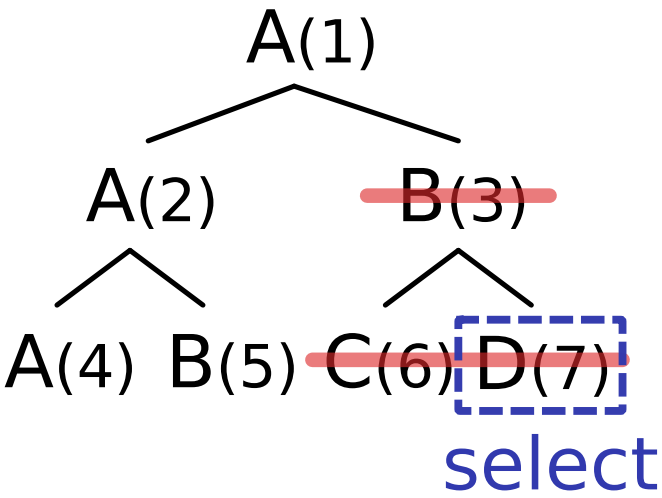
Output: If Tr preserves Q, output "Yes," otherwise "No."

1. Decide whether Tr weakly preserves Q
   - **Yes** — Go to 2.
   - **No** — Output "No"
2. Does "Join" exist?
   - **Yes** — Ourput "No"
   - **No** — Output "Yes" (Tr preserves Q)

Weak Query Preservation: Unary Case (1-RQ)

deleted by Tr

Node to which 'select' is assigned



$Q = (A, S)$

$S = \{$ select, $\ldots \}$

Tr doesn't weakly preserve Q

$\Updownarrow$

There exists a tree such that a node of the tree is deleted by Tr and is selected by Q

Weak Query Preservation: Unary Case (1-RQ)

Input

Query **1-RQ** $Q = (A, S)$

Trans. $\mathbf{DLT}^V$ Tr

1. Construct **TA** A_T, that assigns a "delete" state to nodes that will be deleted by Tr.

2. Construct **TA** A' such that $L(A') = L(A) \cap L(A_T)$.

States of A' are the pairs such that

(state of A, state of A_T).

3. Check whether the set $\{ (s, \text{"delete"}) \mid s \in S \}$ is empty.

Weak Query Preservation: n-ary Case (n-RQ)

Input: n-RQ $Q = (A, S)$ and $\mathbf{DLT}^V$ Tr

1. Divide given query Q into $|S|$ queries.
   Example:
   $$Q = (A, \{(p_1, p_2), (p_1, p_3)\}) \rightarrow Q_1 = (A, \{(p_1, p_2)\}),$$
   $$Q_2 = (A, \{(p_1, p_3)\})$$
2. Flatten queries: n-ary $\rightarrow$ unary queries.
   Flattening: $\{ (p_1, p_2, \ldots, p_n) \} \rightarrow \{ p_1, p_2, \ldots, p_n \}$
3. Input each flattened query to the algorithm for unary case.

Our Algorithm for Query Preservation

Input: $\mathbf{DLT}^V$ Tr and n-**RQ** Q

Output: If Tr preserves Q, output "Yes," otherwise "No."

1. Decide whether Tr weakly preserves Q
   - **Yes** — Go to 2.
   - **No** — Output "No"
2. Does "Join" exist?
   - **Yes** — Ourput "No"
   - **No** — Output "Yes" (Tr preserves Q)

How to Detect "Join"

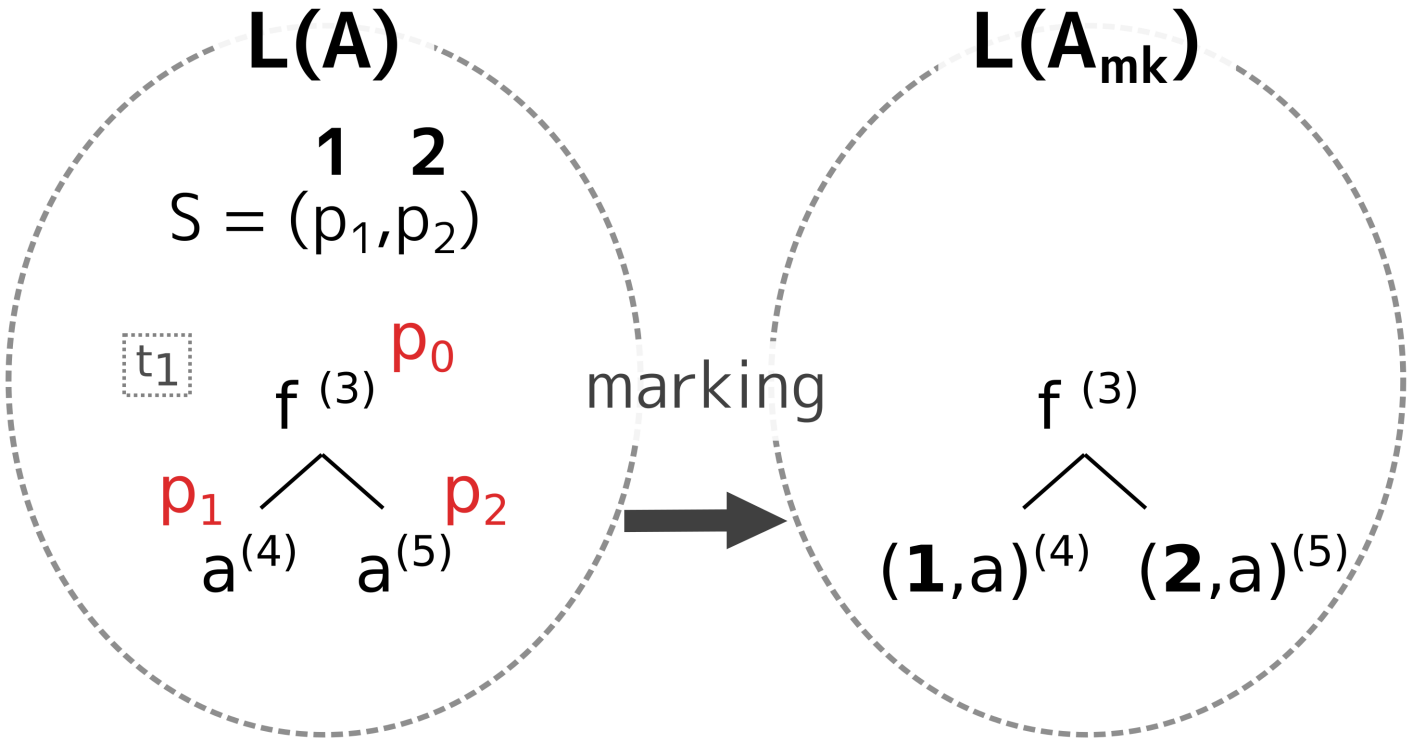
1. Construct **TA** $A_{\mathbf{mk}}$ from A and **DLT** $T_{\mathbf{mk}}$ from Tr:
   - $A_{\mathbf{mk}}$ accepts "marked" trees.
   - $T_{\mathbf{mk}}$ transforms "marked" trees.
2. Construct **TA** $A''_{\mathbf{mk}}$ such that

$$L(A''_{\mathbf{mk}}) = T^{-1}_{\mathbf{mk}}(T_{\mathbf{mk}}(L(A_{\mathbf{mk}}))).$$

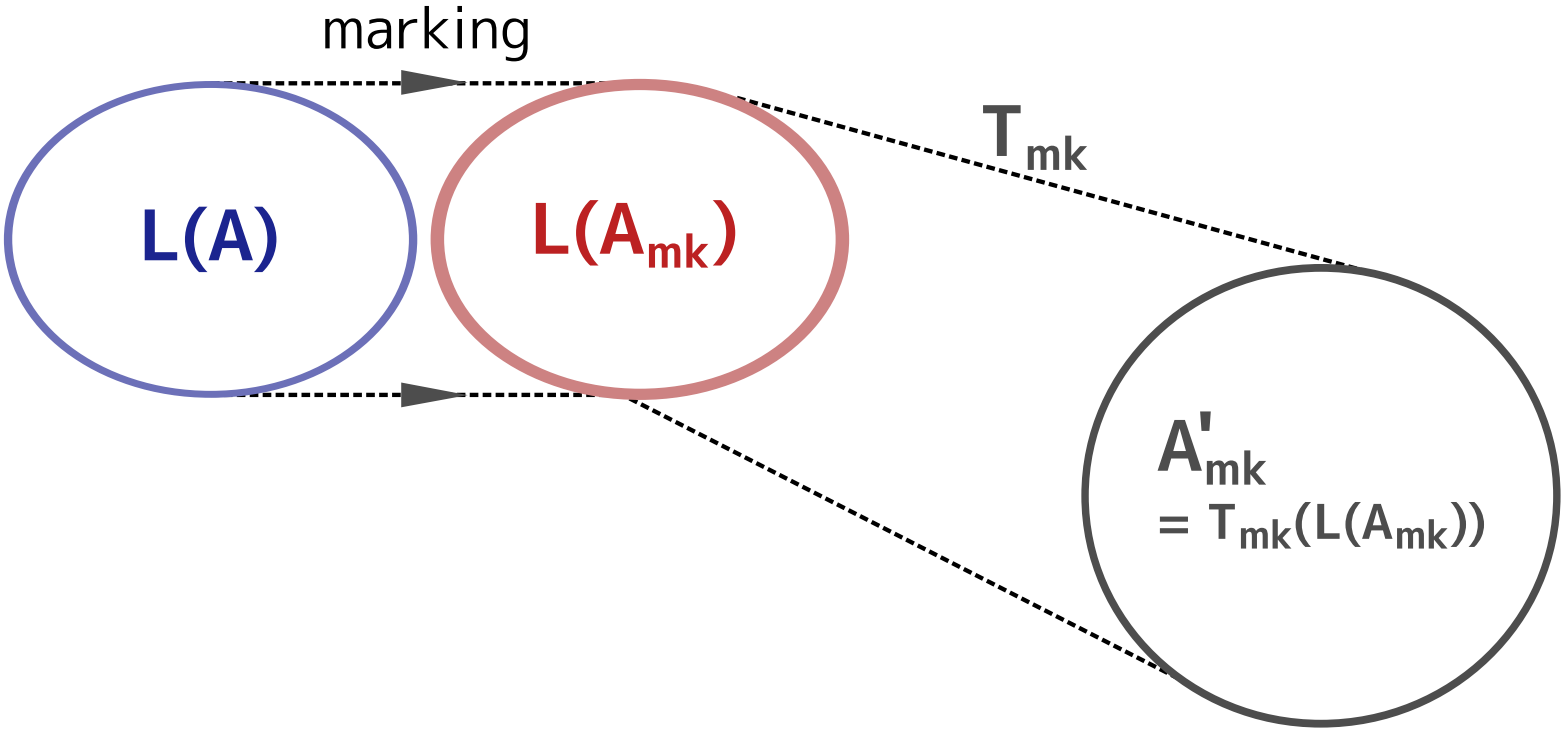
3. Construct **TA** $A_{\mathbf{idx}}$ from A.
   - $A_{\mathbf{idx}}$ accepts all variations of marked trees. Thus $L(A_{\mathbf{idx}}) \supseteq L(A_{\mathbf{mk}})$.
4. If $L(A_{\mathbf{mk}}) = L(A_{\mathbf{idx}}) \cap L(A''_{\mathbf{mk}})$, then output **Yes**, otherwise **No**.

Marked Tree

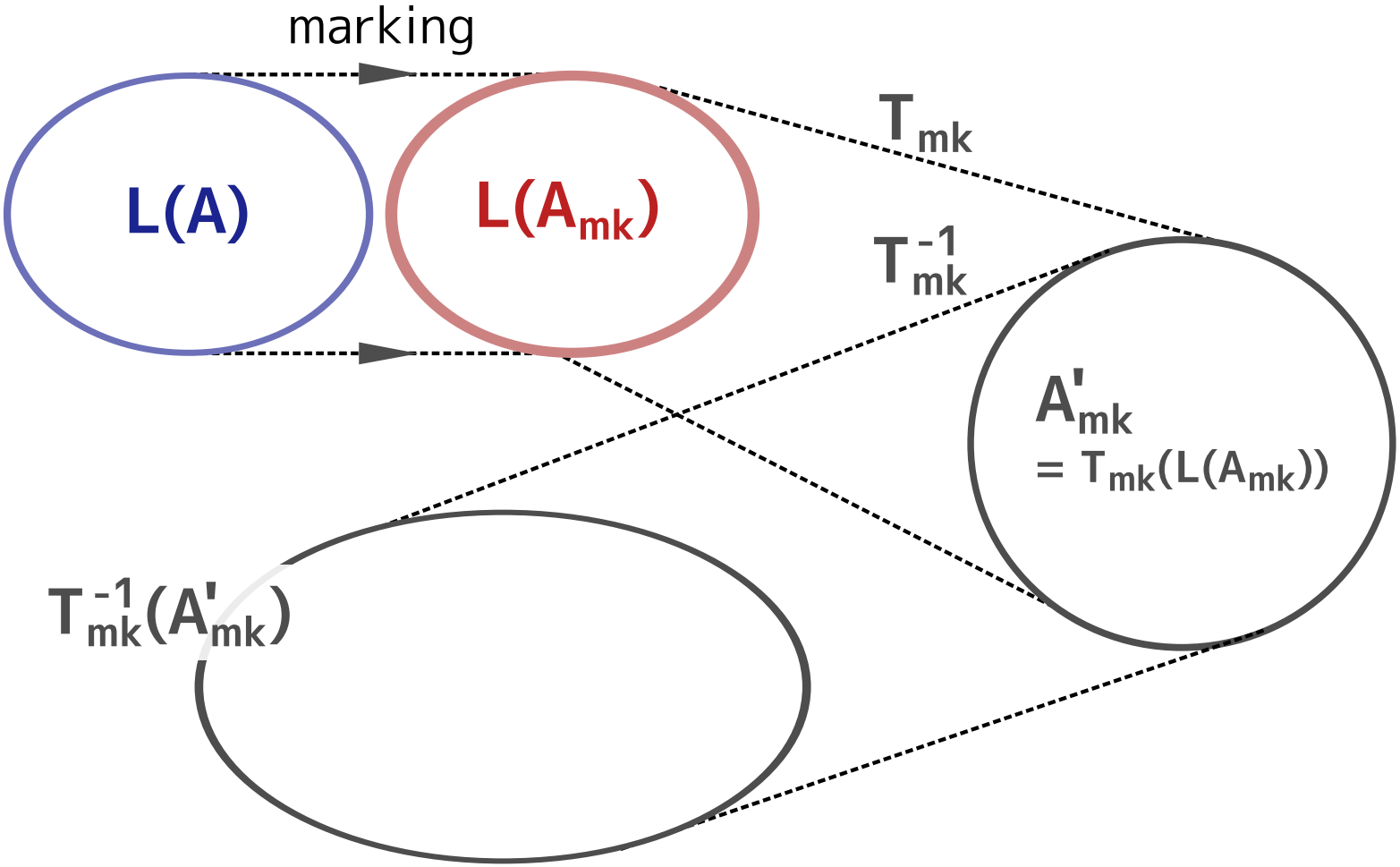
A tree into which run of query automaton is embedded



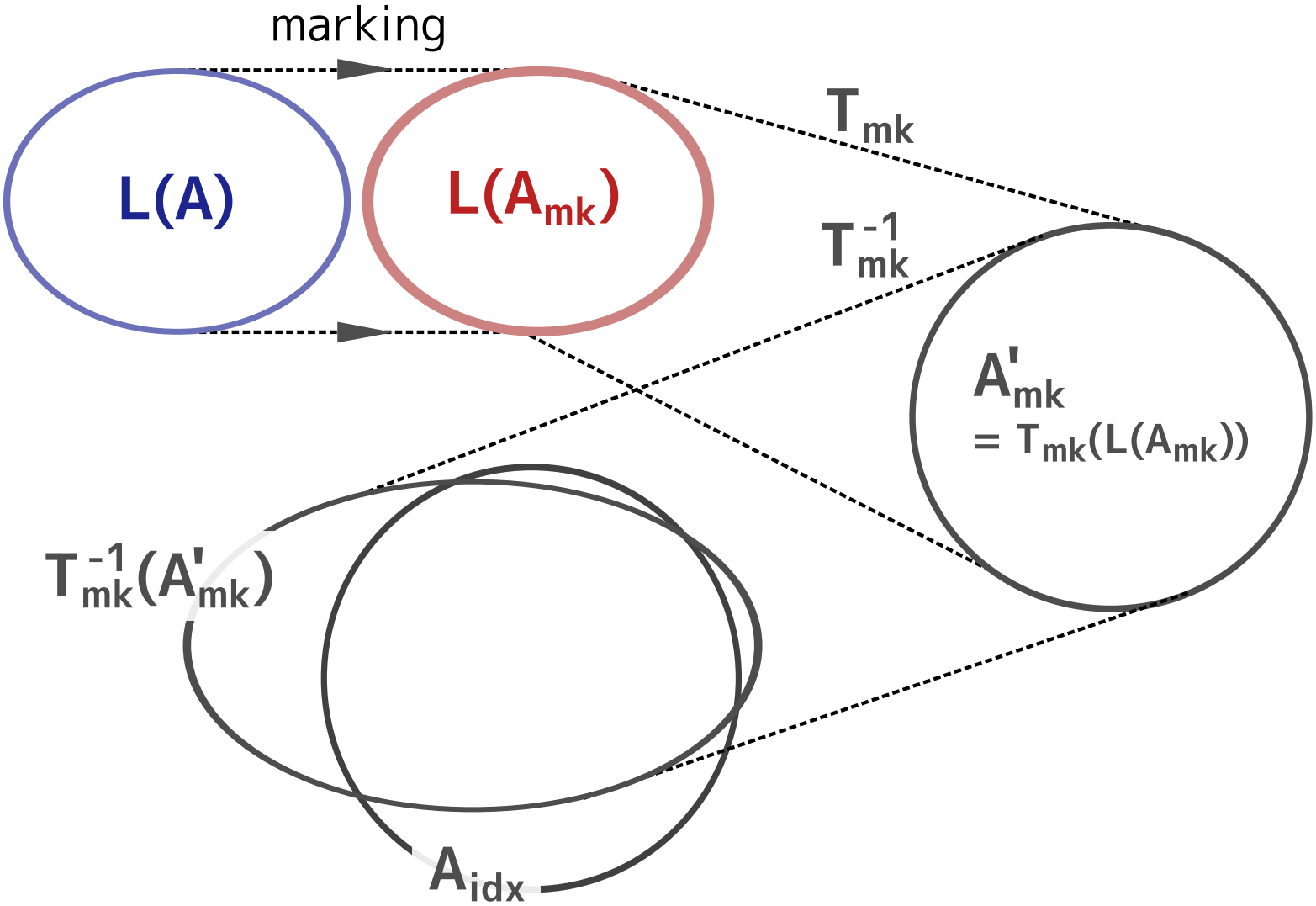
If $L(A_{\mathbf{mk}}) = L(A_{\mathbf{idx}}) \cap L(A''_{\mathbf{mk}})$, then output **Yes**, otherwise **No**.



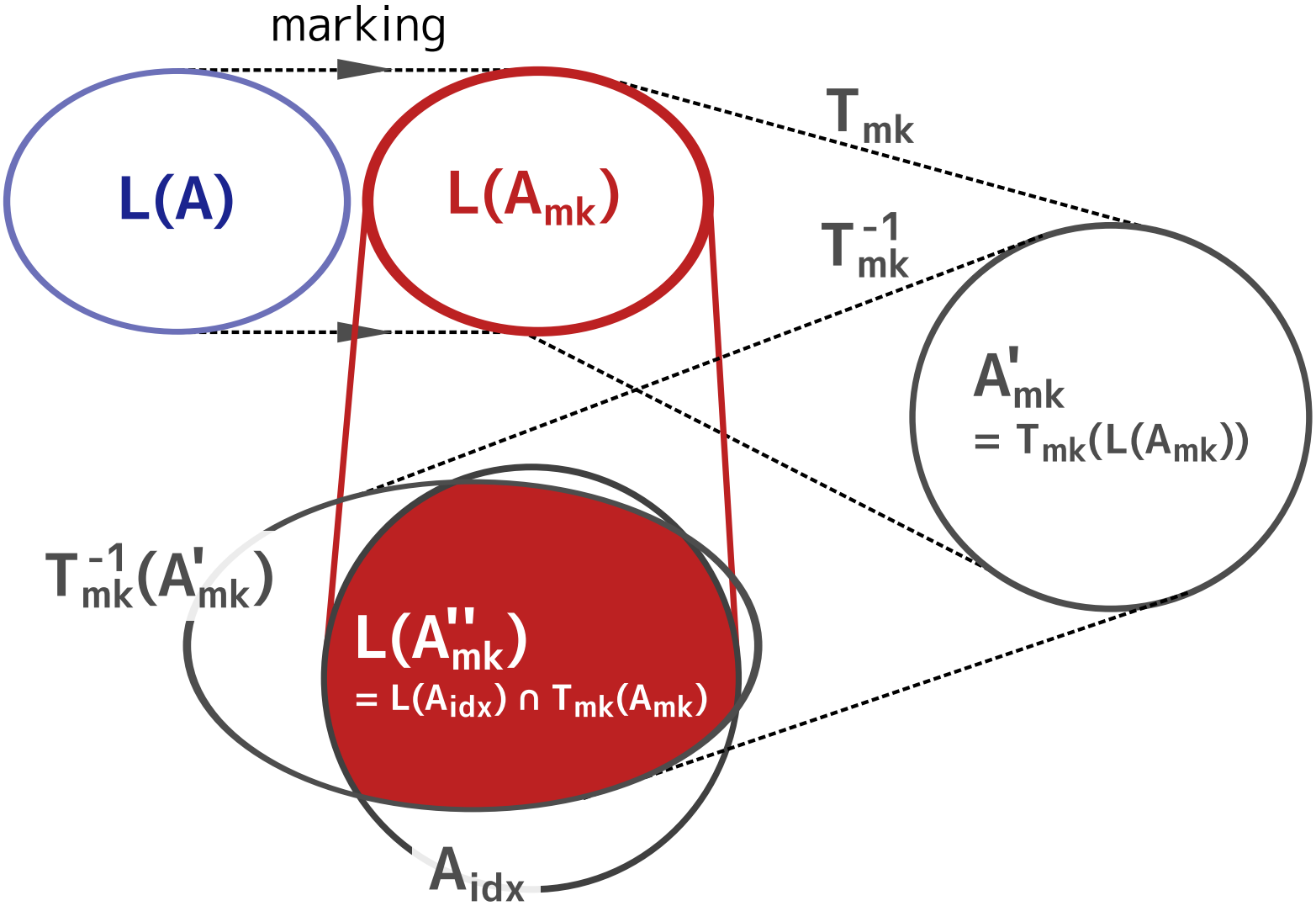
If $L(A_{\mathbf{mk}}) = L(A_{\mathbf{idx}}) \cap L(A''_{\mathbf{mk}})$, then output **Yes**, otherwise **No**.



If $L(A_{\mathbf{mk}}) = L(A_{\mathbf{idx}}) \cap L(A''_{\mathbf{mk}})$, then output **Yes**, otherwise **No**.



If $L(A_{\mathbf{mk}}) = L(A_{\mathbf{idx}}) \cap L(A''_{\mathbf{mk}})$, then output **Yes**, otherwise **No**.



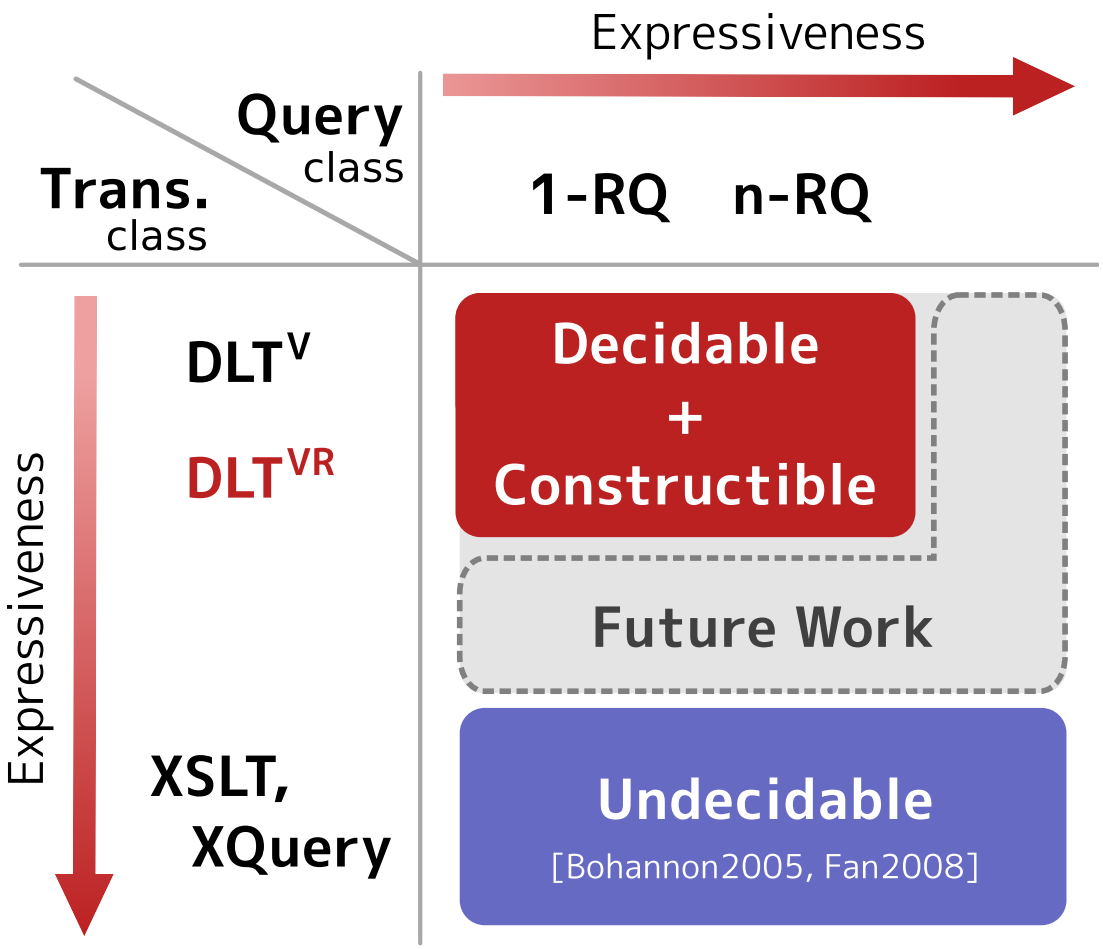
Complexity

Time complexities of $\mathbf{DLT}^{V}$ and $\boldsymbol{n}$-$\mathbf{RQ}$.

Weak Query Preservation	coNP-complete
Query Preservation	in Double-EXPTIME

Conclusion and Future Work

Decidability of Query Preservation



- M. Benedikt, J. Engelfriet, and S. Maneth. **Determinacy and rewriting of top-down and MSO tree transformations.** In *MFCS*, LNCS 8087, pages 146–158, 2013.

- P. Bohannon, W. Fan, M. Flaster, and P. P. S. Narayan. **Information preserving XML schema embedding.** In *VLDB*, pages 85–96, 2005.

- B. Groz, S. Staworko, A.-C. Caron, Y. Roos, and S. Tison. **Static analysis of XML security views and query rewriting.** *Information and Computation*, 2013 (to appear).

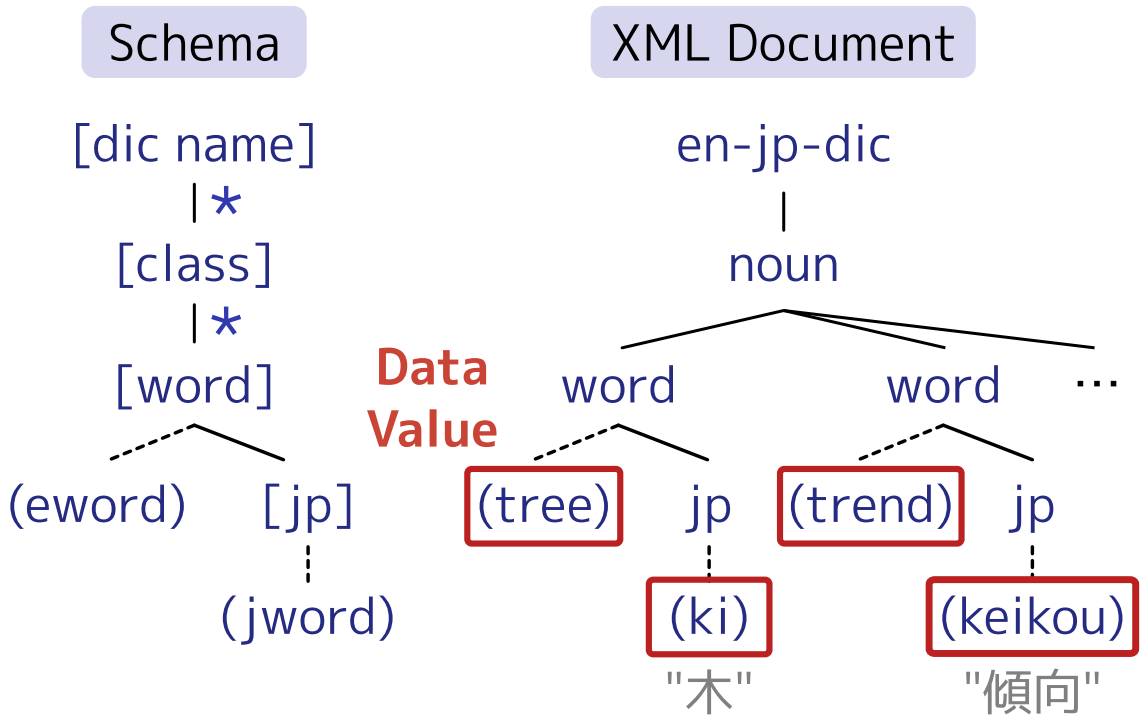
- J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. **N-ary queries by tree automata.** In *DBPL*, LNCS 3774, pages 217–231, 2005.

Appendix

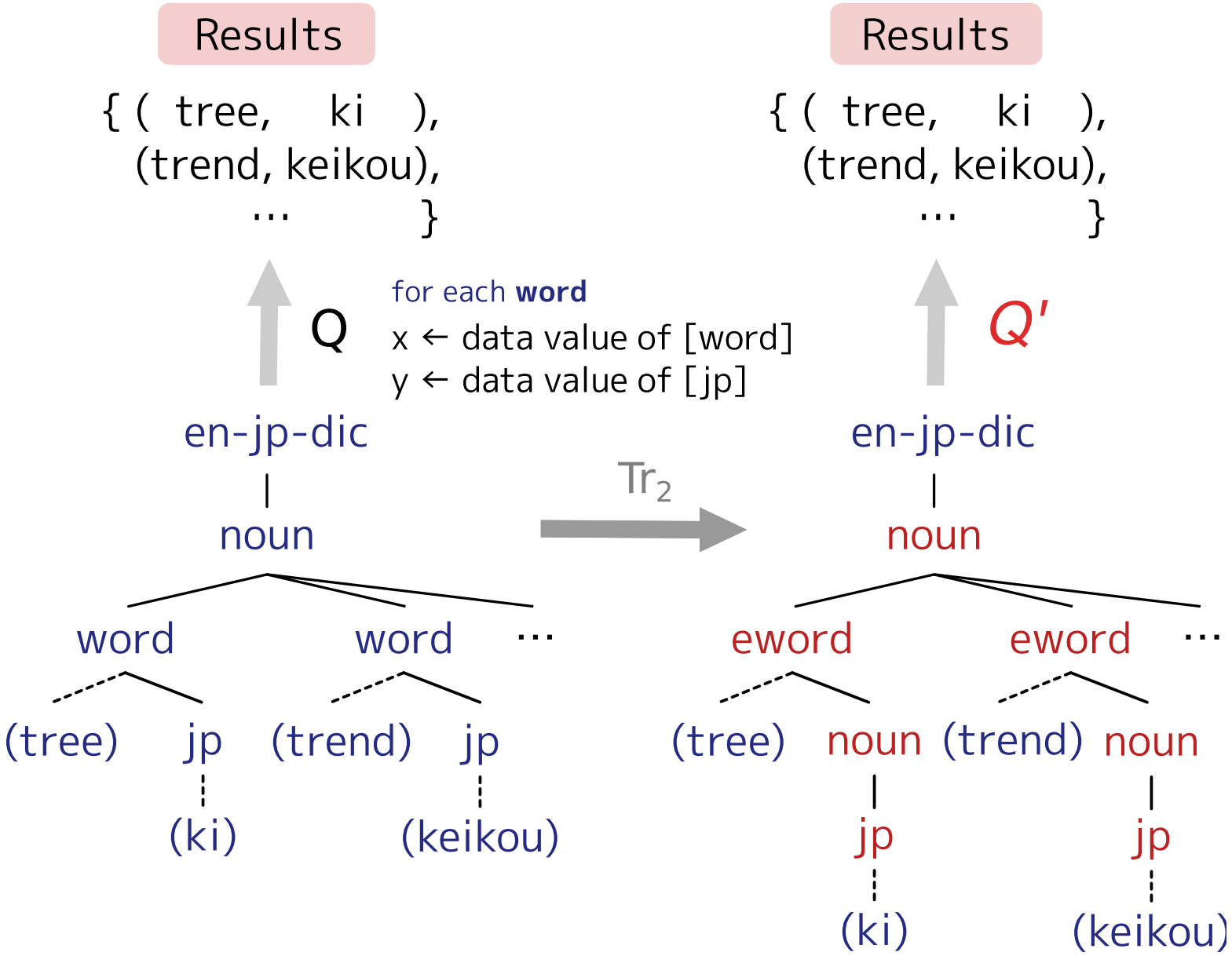
Data Tree

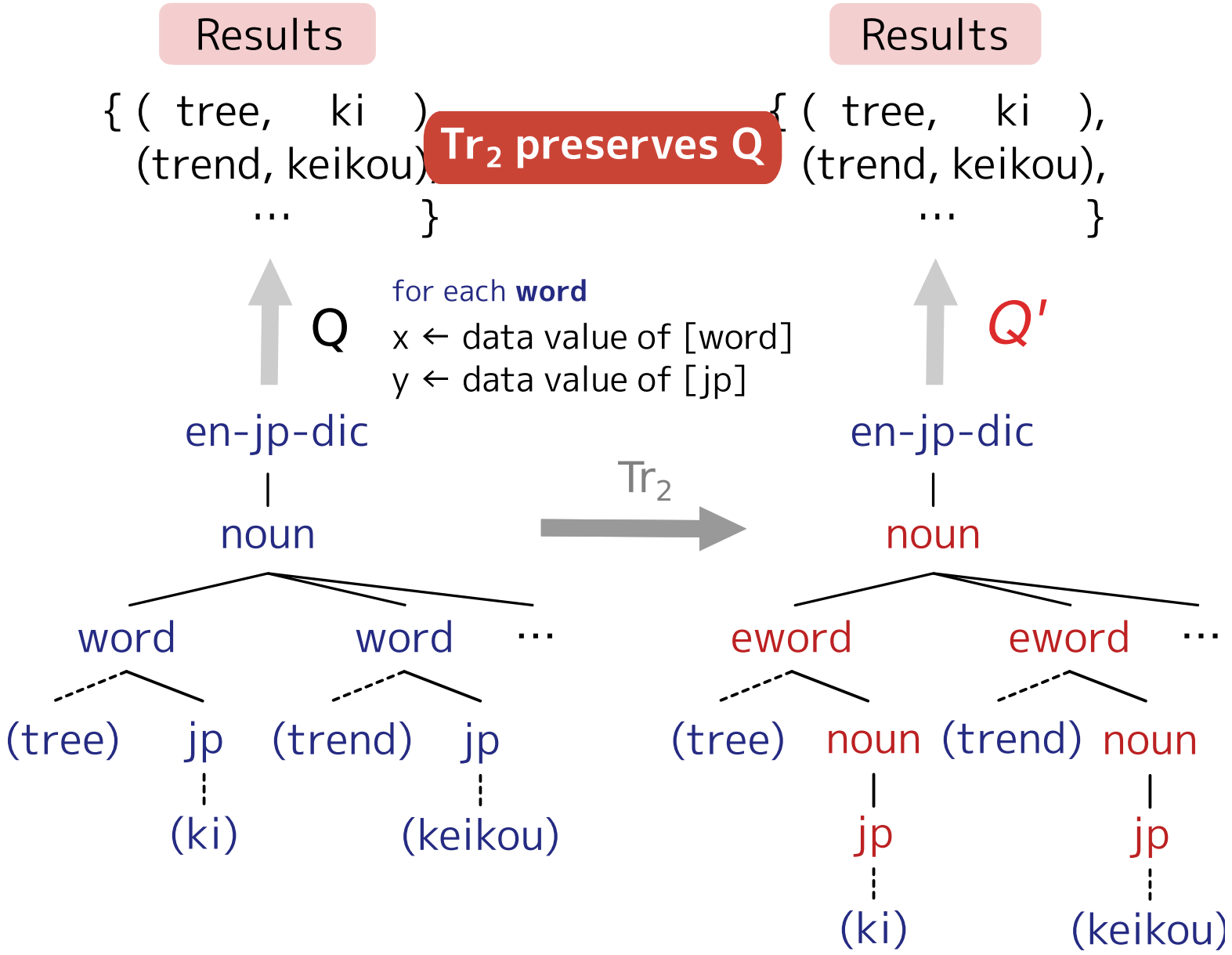
Tree-structured data whose nodes have a data value.

Example: An XML document can be represented by a data tree

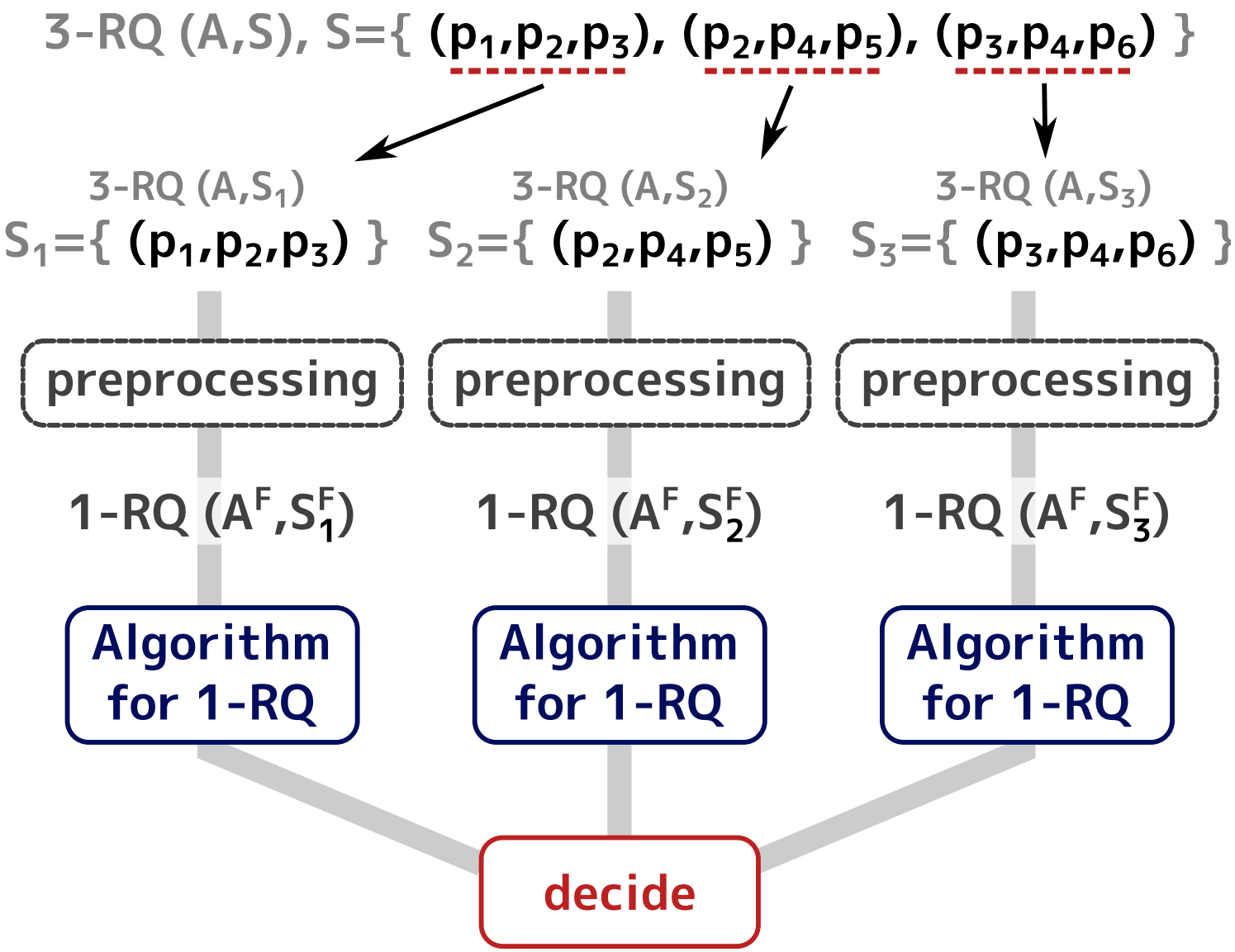



```
<en-jp-dic value="">
 <noun value="">
  <word value="tree">
   <jp value="ki" />
  </word>
  <word value="trend">
   <jp value="keikou" />
  </word>
     ⋮
 </noun>
</en-jp-dic>
```

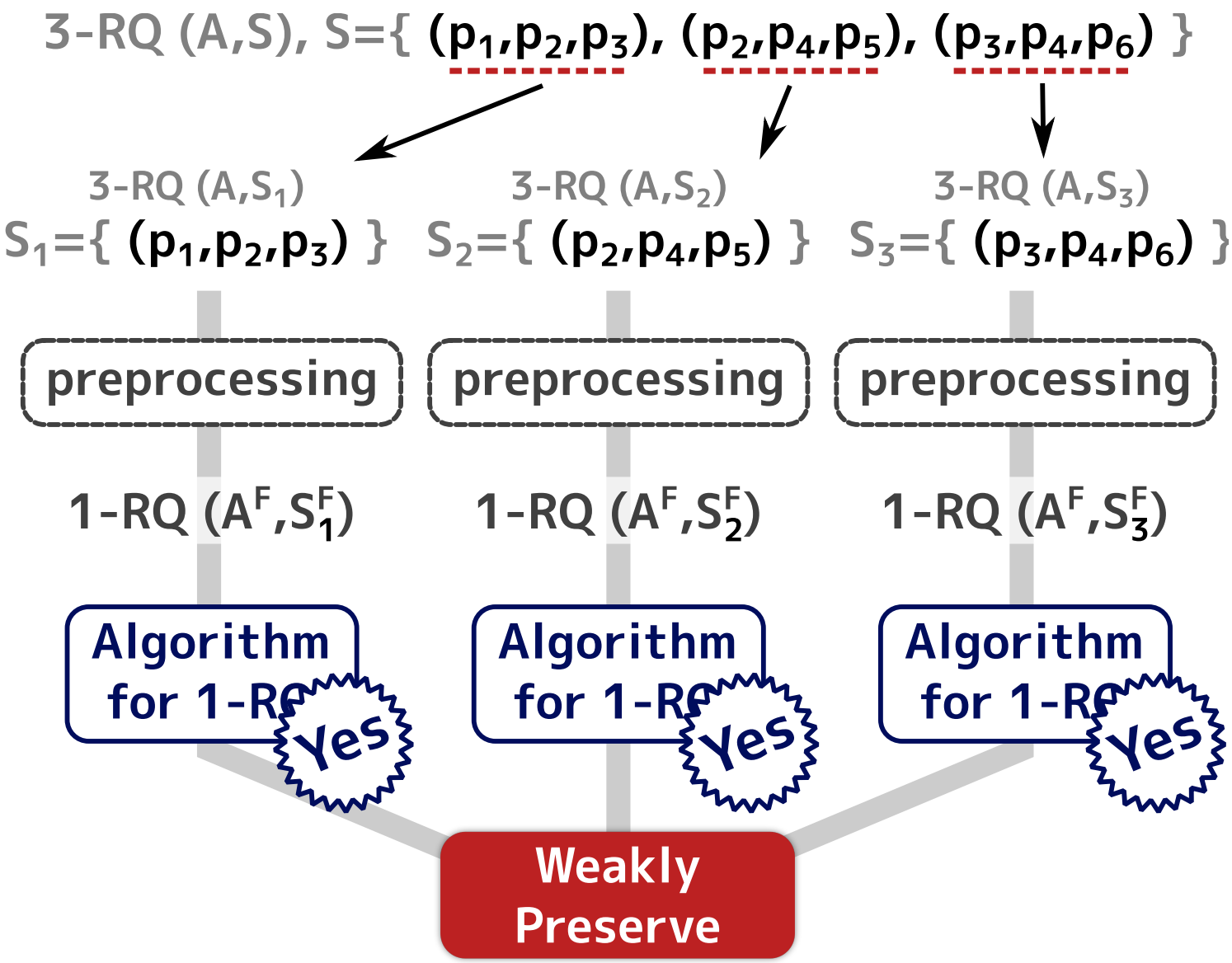

Results
{ (tree, ki),
(trend, keikou),
… }
Q
for each **word**
x ← data value of [word]
y ← data value of [jp]
en-jp-dic
noun
word
word
…
(tree)
jp
(trend)
jp
(ki)
(keikou)
Tr2
Results
{ (tree, ki),
(trend, keikou),
… }
Q′
en-jp-dic
noun
eword
eword
…
(tree)
noun
(trend)
noun
jp
jp
(ki)
(keikou)


Results
{ (tree, ki)
(trend, keikou)
… }
Tr2 preserves Q
Results
{ (tree, ki),
(trend, keikou),
… }
Q
for each word
x ← data value of [word]
y ← data value of [jp]
Q′
en-jp-dic
noun
word
word
…
(tree)
jp
(ki)
(trend)
jp
(keikou)
Tr2
en-jp-dic
noun
eword
eword
…
(tree)
noun
jp
(ki)
(trend)
noun
jp
(keikou)

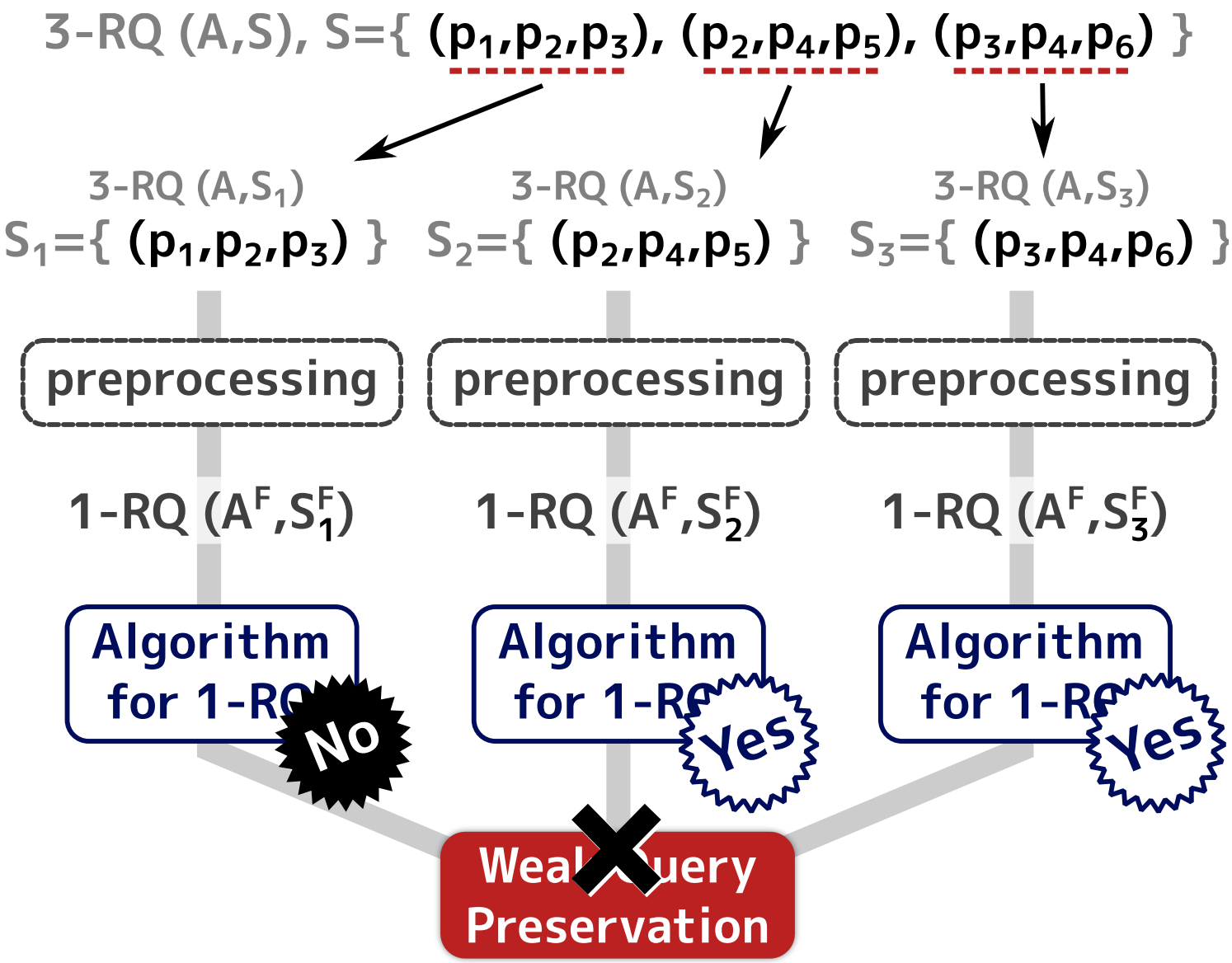
Weak Query Preservation: *n*-ary Case (*n*-RQ)



Weak Query Preservation: *n*-ary Case (*n*-RQ)



Weak Query Preservation: *n*-ary Case (*n*-RQ)

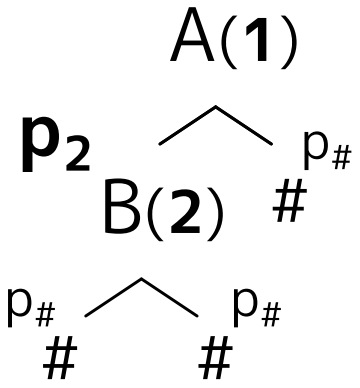


Why Preprocessing Is Needed

Co-occurrence constraint may be lost by simple flattening.

Flattening $\{ (p_1, p_2, \ldots, p_n) \} \rightarrow \{ p_1, p_2, \ldots, p_n \}$

States are assigned by A



Results

3-RQ (A,S)

$S = \{ (p_1, p_2, p_3) \}$

→ **empty set Φ**

$= \{1\} \times \{2\} \times \Phi$

1-RQ (A,S´)

$S' = \{ p_1, p_2, p_3 \}$

→ **{ 1, 2 }**

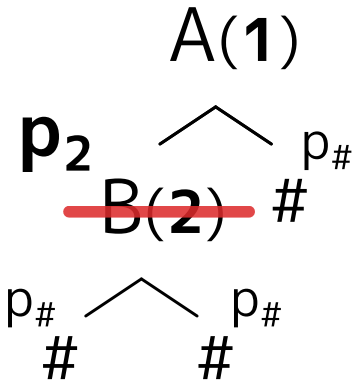
$= \{1\} \cup \{2\} \cup \Phi$

Why Preprocessing Is Needed

Co-occurrence constraint may be lost by simple flattening.

Flattening $\{ (p_1, p_2, \ldots, p_n) \} \rightarrow \{ p_1, p_2, \ldots, p_n \}$

States are assigned by A



Results

3-RQ (A,S)

S = { (p_1, p_2, p_3) }

empty set

Φ

$= \{1\} \times \cancel{\{2\}} \times \Phi$

1-RQ (A,S´)

S´ = { p_1, p_2, p_3 }

{ 1, ~~2~~ }

$= \{1\} \cup \cancel{\{2\}} \cup \Phi$

Preprocessing

Construct A^F and S^F from flattened **1-RQ** (A, S'). A^F accepts tree t if all states in S are assigned to nodes of t by A.

(A, S) is preserved $\Longleftrightarrow$ (A^F, S^F) is preserved

3-RQ (A,S),
S = { (p 1, p2, p3) }

↓ flattening

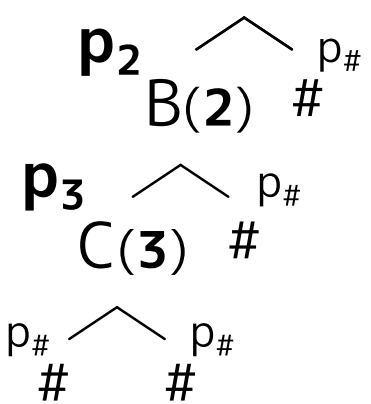
1-RQ (A,S´),
S´ = { p1, p2, p3 }

↓ construct

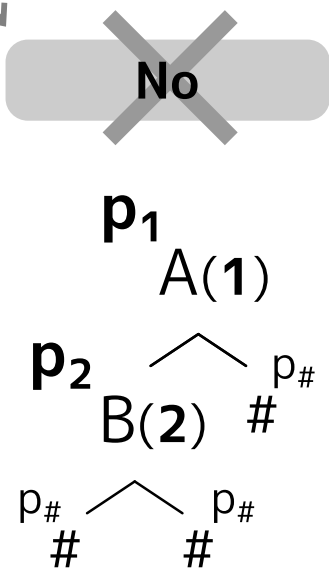
1-RQ (A^F, S^F)

Does A^F accept?

Yes



No



Tree automaton $A_{\mathbf{idx}}$

$A_{\mathbf{idx}}$ accepts **all variations** of marked trees.

Example: There are $3^3 = 27$ variations of marked trees.