CSE599d: Advanced Query Processing

Lecture 4: Non-Reordable Operators

Dan Suciu

University of Washington

Motivation

Dynamic Programming: query graph $\Rightarrow$ optimal join tree.

Joins are "reordable" and any join tree is correct.

When operators are non-reordable: need to restrict to correct plans.

[Moerkotte and Neumann, 2008]
[Moerkotte et al., 2013]
[Birler and Neumann, 2025]

Non-Reordable Operators

Left semi join $\ltimes$

Left anti join $\triangleright$

Left outer join $⟕$

Right outer join $⟖$

Full outer join $⟗$

Query Definition

The Query Graph is insufficient to define the query.

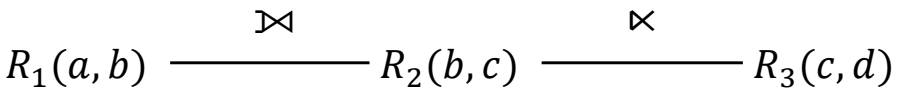
Query Definition

The Query Graph is insufficient to define the query.

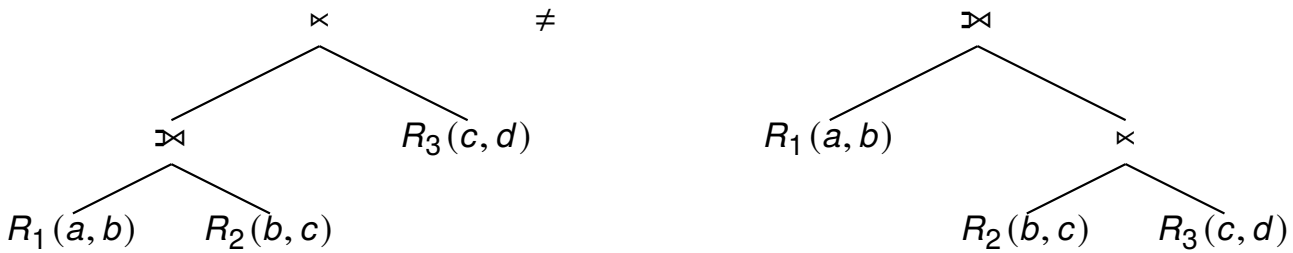
$$R_1(a,b) \overset{\bowtie}{\text{———}} R_2(b,c) \overset{\ltimes}{\text{———}} R_3(c,d)$$

Query Definition

The Query Graph is insufficient to define the query.

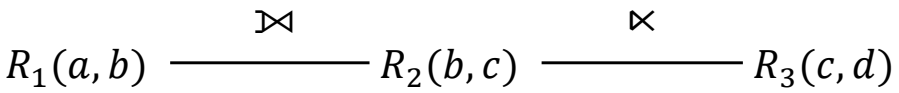


It may represent any of these two inequivalent queries:

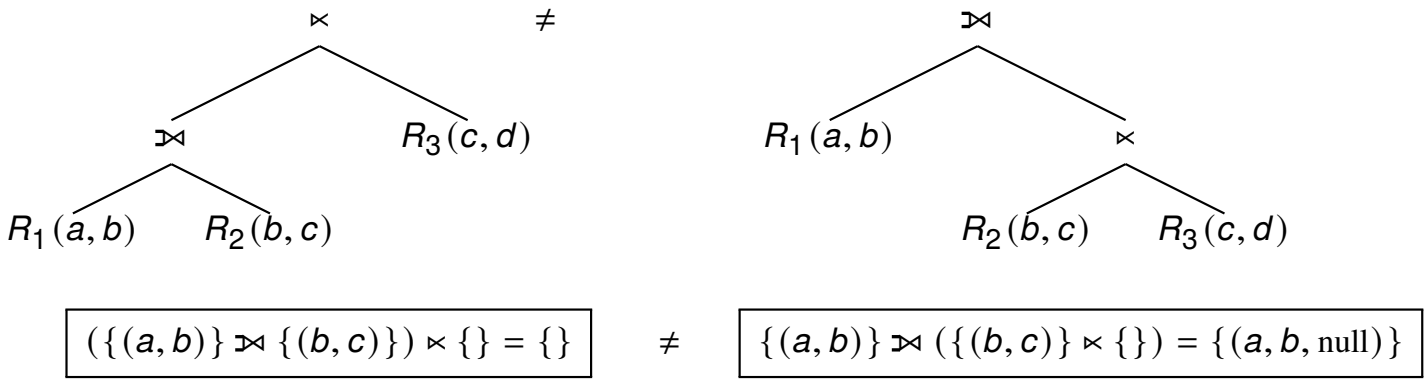


Query Definition

The Query Graph is insufficient to define the query.

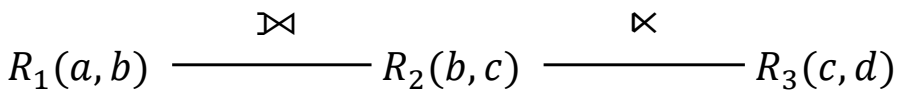


It may represent any of these two inequivalent queries:

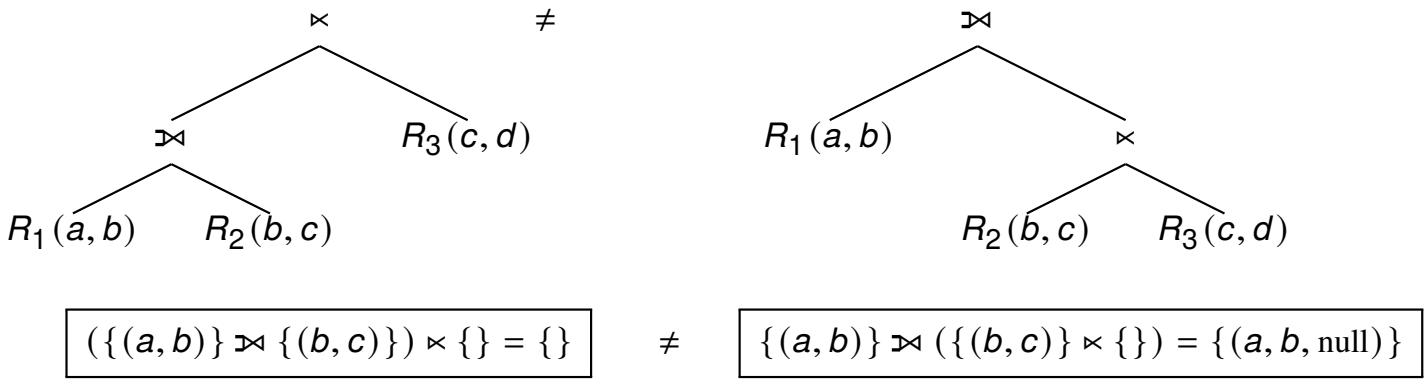


Query Definition

The Query Graph is insufficient to define the query.



It may represent any of these two inequivalent queries:



The query must be given as an Initial Operator Tree.

Notation for Operands

We use indices to indicate where the operands come from.

$$\boxed{(e_1 \circ_{12} e_2) \circ_{13} e_3} \text{ v.s. } \boxed{(e_1 \circ_{12} e_2) \circ_{23} e_3}$$

Identities for Non-reordable Operators

Query Equivalence

Two ways to prove an equivalence $\boxed{E_1 \equiv E_2}$.

Query Equivalence

Two ways to prove an equivalence $\boxed{E_1 \equiv E_2}$.

- Semantic.
  - Useful for pen-and-paper proofs.
  - Left- and right-linearity can simplify the proof (next slides).

Query Equivalence

Two ways to prove an equivalence $\boxed{E_1 \equiv E_2}$.

- Semantic.
  - Useful for pen-and-paper proofs.
  - Left- and right-linearity can simplify the proof (next slides).

- Using Identities.
  - Establish a fix, small set of simple identities.
  - Check that they are sound.
  - Use them to derive $E_1 \equiv E_2$.
  - Basis of query optimizers.

Left- and Right-Linearity

An operator $\circ$ is left linear if

$$(R_1 \cup R_2) \circ S = (R_1 \circ S) \cup (R_2 \circ S)$$

Left- and Right-Linearity

An operator $\circ$ is left linear if

$$\boxed{(R_1 \cup R_2) \circ S = (R_1 \circ S) \cup (R_2 \circ S)}$$

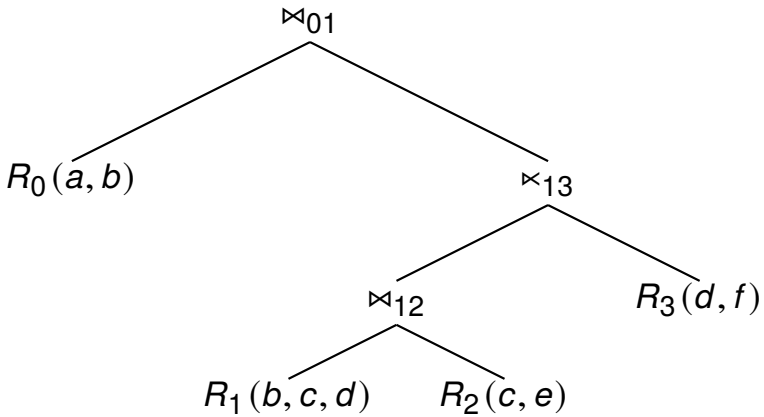
Left linear operators: $\bowtie, \rightthreetimes, \triangleright, \ltimes$

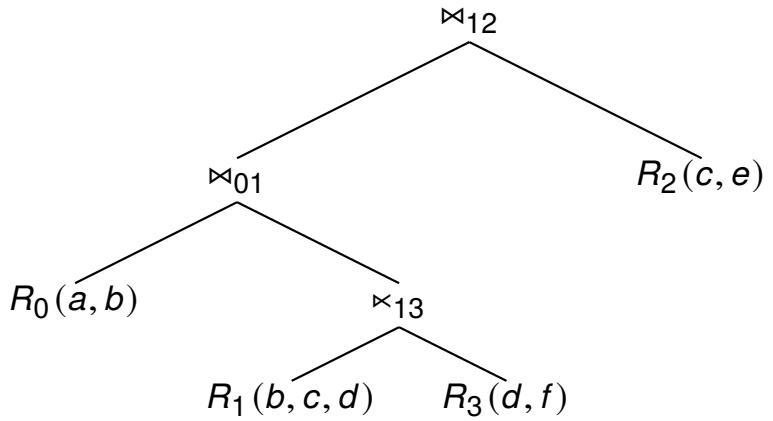
$$(R_1 \cup R_2) \triangleright S = (R_1 \triangleright S) \cup (R_2 \triangleright S)$$

Right linear operators: $\bowtie, \leftthreetimes$

If E_1, E_2 are linear in R, then check $E_1 \equiv E_2$ by setting $R :=$ single tuple.

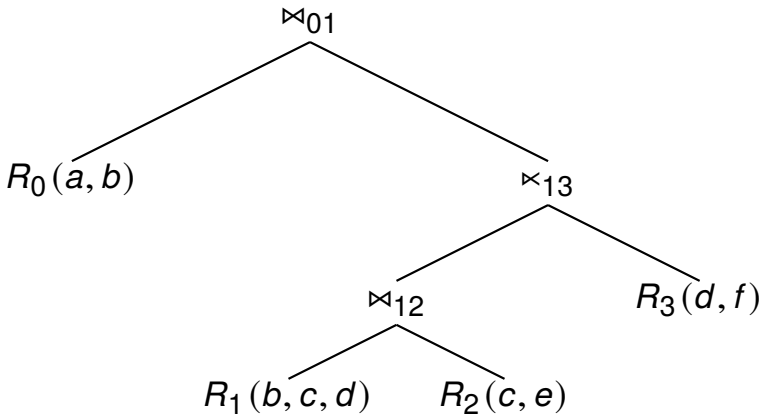
Example: Prove $E_1 \equiv E_2$

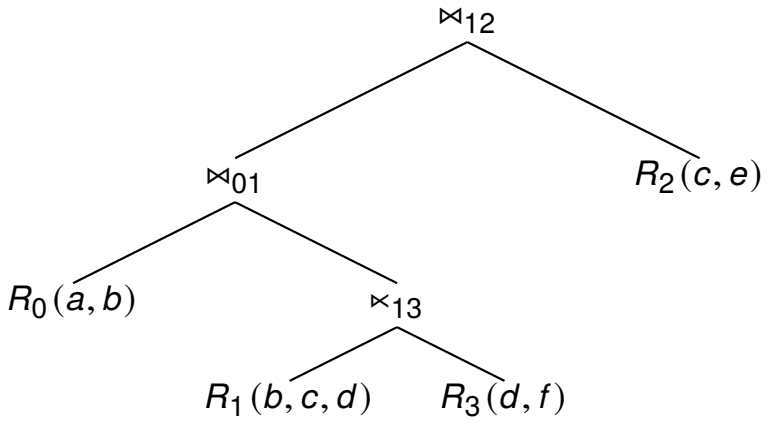




Question 1: what is the output schema?

Example: Prove $E_1 \equiv E_2$

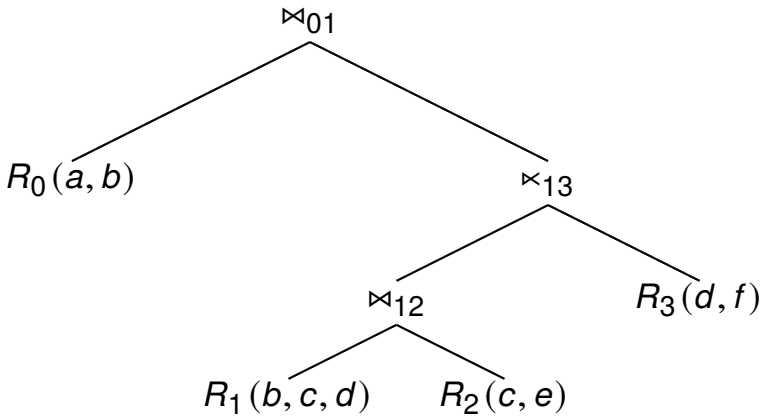


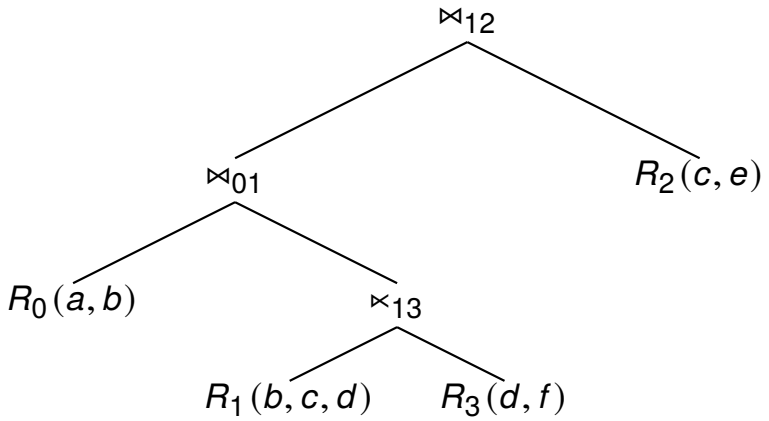


Question 1: what is the output schema?

$Q(a, b, c, d, e)$

Example: Prove $E_1 \equiv E_2$



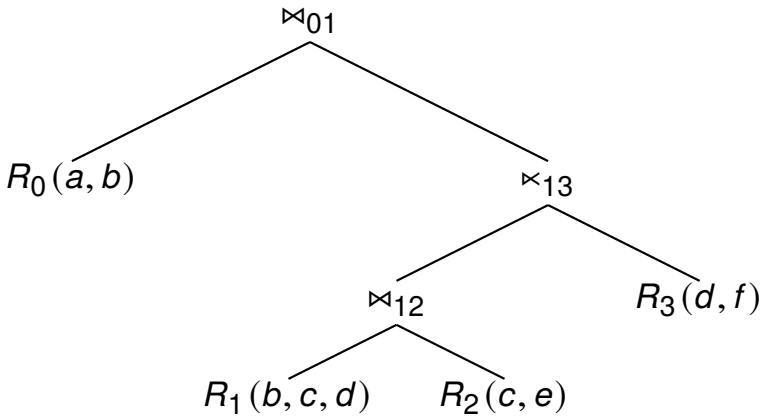


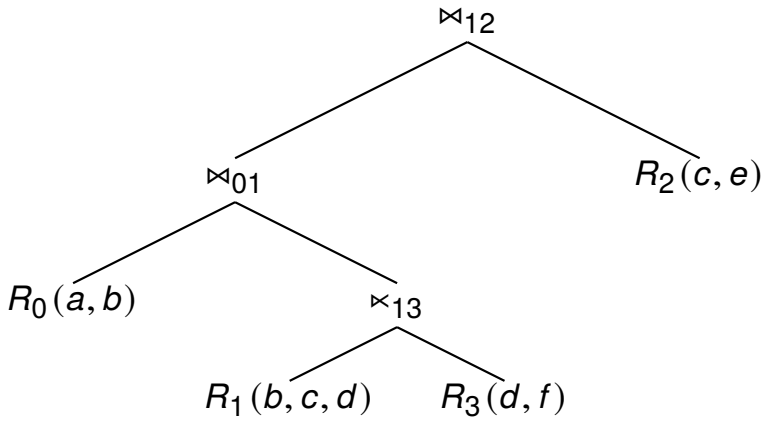
Question 1: what is the output schema?

$Q(a, b, c, d, e)$

Question 2: prove equivalence.

Example: Prove $E_1 \equiv E_2$

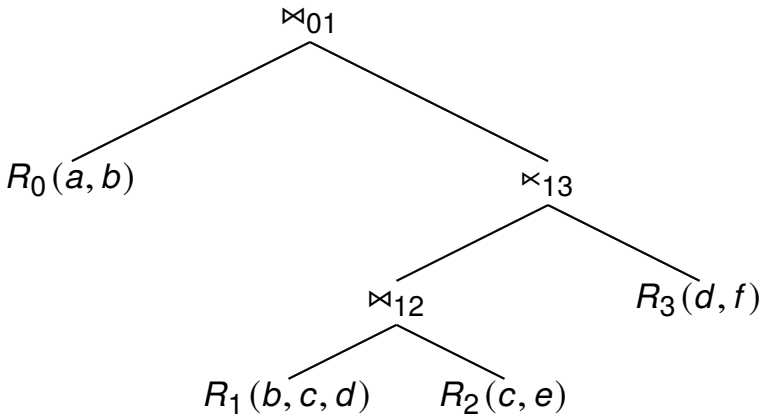


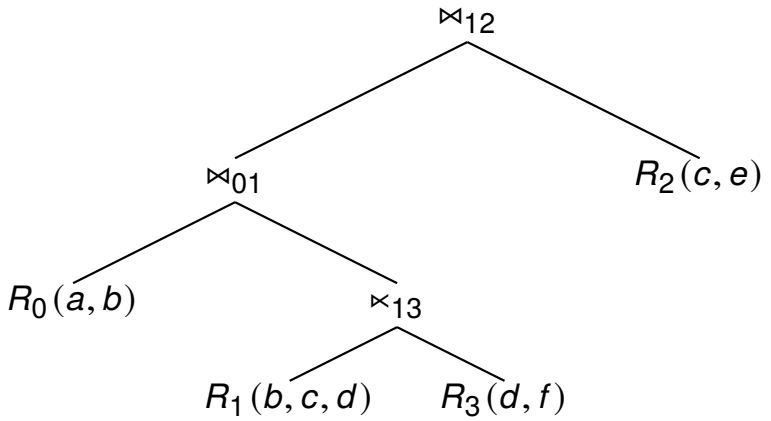


Question 1: what is the output schema? $Q(a, b, c, d, e)$

Question 2: prove equivalence. Linear in R_1.

Example: Prove $E_1 \equiv E_2$

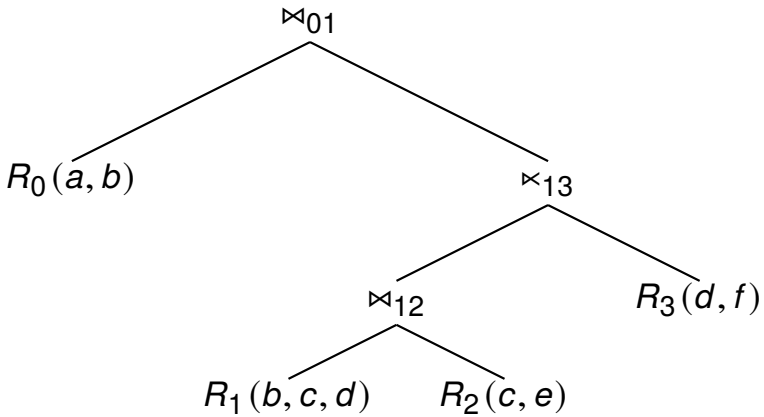


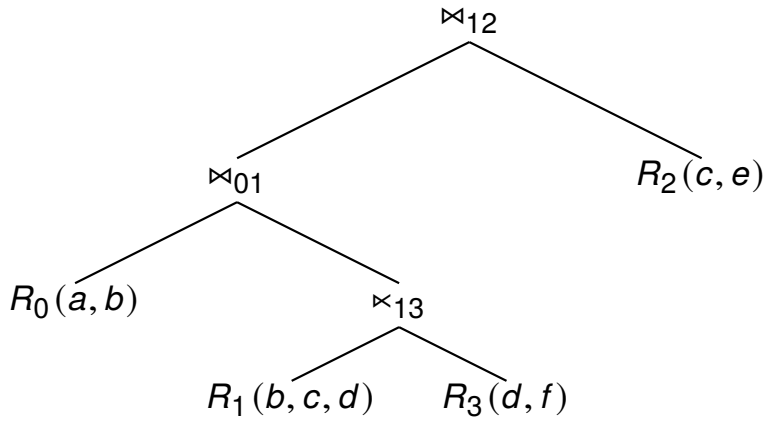


Question 1: what is the output schema? $Q(a, b, c, d, e)$

Question 2: prove equivalence. Linear in R_1. $R_1 := \{(b, c, d)\}$

Example: Prove $E_1 \equiv E_2$



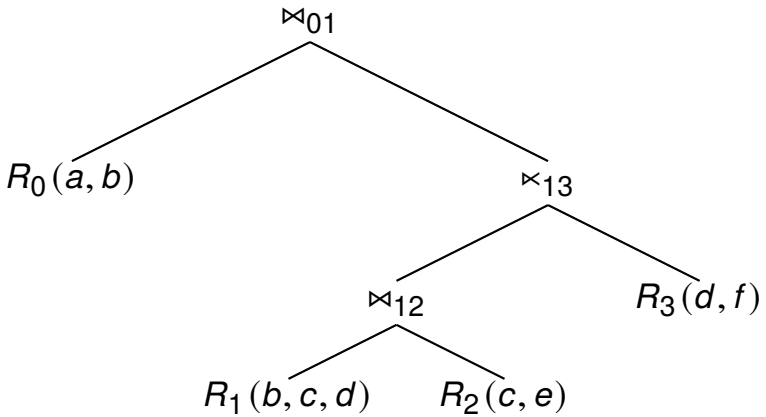


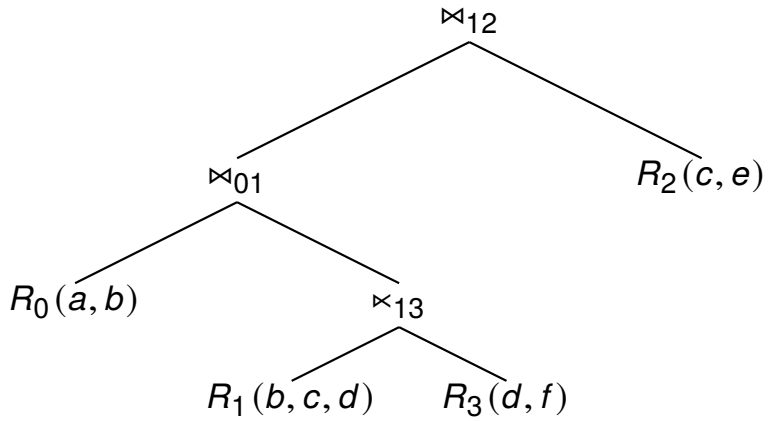
Question 1: what is the output schema? $Q(a, b, c, d, e)$

Question 2: prove equivalence. Linear in R_1. $R_1 := \{(b, c, d)\}$

If $d \notin R_3$ then both expressions are $\emptyset$.

Example: Prove $E_1 \equiv E_2$

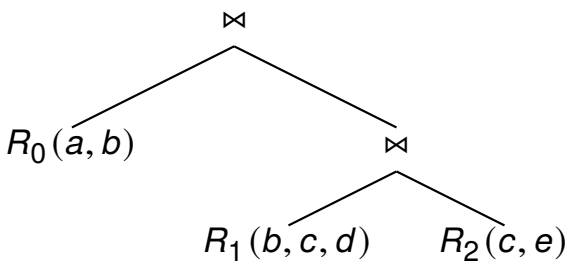




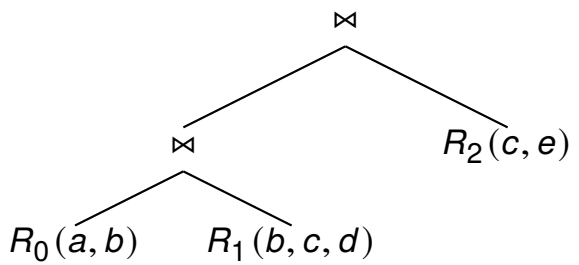
Question 1: what is the output schema? $Q(a, b, c, d, e)$

Question 2: prove equivalence. Linear in R_1. $R_1 := \{(b, c, d)\}$

If $d \notin R_3$ then both expressions are $\emptyset$.



If $d \in R_3$ then:



Identity Patterns

$\circ$-Commutativity:

$$e_1 \circ_{12} e_2 = e_2 \circ_{12} e_1$$

$(\circ^a, \circ^b)$-Associativity:

$$(e_1 \circ^a_{12} e_2) \circ^b_{23} e_3 = e_1 \circ^a_{12} (e_2 \circ^b_{23} e_3)$$

$(\circ^a, \circ^b)$-Left-Asscom (l-asscom)

$$(e_1 \circ^a_{12} e_2) \circ^b_{13} e_3 = (e_1 \circ^b_{13} e_3) \circ^a_{12} e_2$$

$(\circ^a, \circ^b)$-Right-Asscom (r-asscom)

$$e_1 \circ^a_{13} (e_2 \circ^b_{23} e_3) = e_2 \circ^b_{23} (e_1 \circ^a_{13} e_3)$$

∘-Commutativity

∘	×	⋈	⟕	▷	⟗	⟖	⟙
comm(∘)	+	+	-	-	-	+	-

Table 1: The comm(∘)-property

$$e_1 \circ_{12} e_2 = e_2 \circ_{12} e_1$$

[Moerkotte et al., 2013]

Assoc($\circ^a$, $\circ^b$)

$\circ^a$	$\circ^b$						
	$\times$	$\bowtie$	$\ltimes$	$\triangleright$	$\overline{\bowtie}$	$\overline{\bowtie}$	$\overline{\bowtie}$
$\times$	+	+	+	+	+	-	+
$\bowtie$	+	+	+	+	+	-	+
$\ltimes$	-	-	-	-	-	-	-
$\triangleright$	-	-	-	-	-	-	-
$\overline{\bowtie}$	-	-	-	-	+[1]	-	-
$\overline{\bowtie}$	-	-	-	-	+[1]	+[2]	-
$\overline{\bowtie}$	-	-	-	-	-	-	-

[1] if p_{23} rejects nulls on $\mathcal{A}(e_2)$ (Eqv. 1)
[2] if p_{12} and p_{23} reject nulls on $\mathcal{A}(e_2)$ (Eqv. 1)

Table 2: The assoc($\circ^a$, $\circ^b$)-property

$$(e_1 \circ^a_{12} e_2) \circ^b_{23} e_3 = e_1 \circ^a_{12} (e_2 \circ^b_{23} e_3)$$

Prove in class:

$$(R(a,b) \overline{\bowtie} S(b,c)) \overline{\bowtie} T(c,d) \equiv R(a,b) \overline{\bowtie} (S(b,c) \overline{\bowtie} T(c,d))$$

[Moerkotte et al., 2013]

Null Rejection

A predicate p is null rejecting for a set of attributes A if it evaluates to FALSE or UNKNOWN when all attributes A are NULL.

Null Rejection

A predicate p is null rejecting for a set of attributes A if it evaluates to FALSE or UNKNOWN when all attributes A are NULL.

$$\boxed{(e_1 \rtimes_{12} e_2) \rtimes_{23} e_3 = e_1 \rtimes_{12} (e_2 \rtimes_{23} e_3)}$$

holds if p_{23} rejects NULLs on $\mathcal{A}(e_2)$.

Null Rejection

A predicate p is null rejecting for a set of attributes A if it evaluates to FALSE or UNKNOWN when all attributes A are NULL.

$$\boxed{(e_1 \rtimes_{12} e_2) \rtimes_{23} e_3 = e_1 \rtimes_{12} (e_2 \rtimes_{23} e_3)}$$

holds if p_{23} rejects NULLs on $\mathcal{A}(e_2)$.

$$(R(a,b) \rtimes_{b=c} S(c,d)) \rtimes_{d=e} T(e,f) \equiv R(a,b) \rtimes_{b=c} (S(c,d) \rtimes_{d=e} T(e,f))$$

$$(R(a,b) \rtimes_{b=c} S(c,d)) \rtimes_{\text{coalesce}(d,0)=e} T(e,f) \not\equiv R(a,b) \rtimes_{b=c} (S(c,d) \rtimes_{\text{coalesce}(d,0)=e} T(e,f))$$

Null Rejection

A predicate p is null rejecting for a set of attributes A if it evaluates to FALSE or UNKNOWN when all attributes A are NULL.

$$\boxed{(e_1 \rtimes_{12} e_2) \rtimes_{23} e_3 = e_1 \rtimes_{12} (e_2 \rtimes_{23} e_3)}$$

holds if p_{23} rejects NULLs on $\mathcal{A}(e_2)$.

$$(R(a,b) \rtimes_{b=c} S(c,d)) \rtimes_{d=e} T(e,f) \equiv R(a,b) \rtimes_{b=c} (S(c,d) \rtimes_{d=e} T(e,f))$$

$$(R(a,b) \rtimes_{b=c} S(c,d)) \rtimes_{\text{coalesce}(d,0)=e} T(e,f) \not\equiv R(a,b) \rtimes_{b=c} (S(c,d) \rtimes_{\text{coalesce}(d,0)=e} T(e,f))$$

SQL refresher: $\text{coalesce}(\text{NULL}, 7) = 7, \quad \text{coalesce}(5, 7) = 5.$

L/R-Asscom($\circ^a$, $\circ^b$)

$\circ$	$\times$	$\bowtie$	$\ltimes$	$\triangleright$	$\overline{\bowtie}$	$\overline{\ltimes}$	$\overline{\bowtie}$
$\times$	+/+	+/+	+/-	+/-	+/-	-/-	+/-
$\bowtie$	+/+	+/+	+/-	+/-	+/-	-/-	+/-
$\ltimes$	+/-	+/-	+/-	+/-	+/-	-/-	+/-
$\triangleright$	+/-	+/-	+/-	+/-	+/-	-/-	+/-
$\overline{\bowtie}$	+/-	+/-	+/-	+/-	+/-	$+^1$/-	+/-
$\overline{\ltimes}$	-/-	-/-	-/-	-/-	$+^2$/-	$+^3/+^4$	-/-
$\overline{\bowtie}$	+/-	+/-	+/-	+/-	+/-	-/-	+/-

[1] if p_{12} rejects nulls on $\mathcal{A}(e_1)$ (Eqv. 2)
[2] if p_{13} rejects nulls on $\mathcal{A}(e_3)$ (Eqv. 2)
[3] if p_{12} and p_{13} rejects nulls on $\mathcal{A}(e_1)$ (Eqv. 2)
[4] if p_{13} and p_{23} reject nulls on $\mathcal{A}(e_3)$ (Eqv. 3)

Table 3: The l-/r-asscom($\circ^a, \circ^b$) property

$$(e_1 \circ^a_{12} e_2) \circ^b_{13} e_3 = (e_1 \circ^b_{13} e_3) \circ^a_{12} e_2$$

L/R-Asscom($\circ^a$, $\circ^b$)

$\circ$	$\times$	$\bowtie$	$\ltimes$	$\triangleright$	⟕	⟗	$\bowtie^{\Gamma}$
$\times$	+/+	+/+	+/-	+/-	+/-	-/-	+/-
$\bowtie$	+/+	+/+	+/-	+/-	+/-	-/-	+/-
$\ltimes$	+/-	+/-	+/-	+/-	+/-	-/-	+/-
$\triangleright$	+/-	+/-	+/-	+/-	+/-	-/-	+/-
⟕	+/-	+/-	+/-	+/-	+/-	+[1]/-	+/-
⟗	-/-	-/-	-/-	-/-	+[2]/-	+[3]/+[4]	-/-
$\bowtie^{\Gamma}$	+/-	+/-	+/-	+/-	+/-	-/-	+/-

[1] if p_{12} rejects nulls on $\mathcal{A}(e_1)$ (Eqv. 2)
[2] if p_{13} rejects nulls on $\mathcal{A}(e_3)$ (Eqv. 2)
[3] if p_{12} and p_{13} rejects nulls on $\mathcal{A}(e_1)$ (Eqv. 2)
[4] if p_{13} and p_{23} reject nulls on $\mathcal{A}(e_3)$ (Eqv. 3)

Table 3: The l-/r-asscom($\circ^a$, $\circ^b$) property

$$(e_1 \circ^a_{12} e_2) \circ^b_{13} e_3 = (e_1 \circ^b_{13} e_3) \circ^a_{12} e_2$$

Prove in class:

$$(R(a,b,c) ⟕ S(a,d)) ⟕ T(b,e) = (R(a,b,c) ⟕ T(b,e)) ⟕ S(a,d)$$

[Moerkotte et al., 2013]

Notation and Discussion

$\boxed{E_1 \equiv E_2}$ if, for any database D, $E_1(D) = E_2(D)$.

$\boxed{E_1 \simeq E_2}$ if E_1 can be written to E_2 using COMM, ASSOC, L/R-ASSCOM

Notation and Discussion

$\boxed{E_1 \equiv E_2}$ if, for any database D, $E_1(D) = E_2(D)$.

$\boxed{E_1 \simeq E_2}$ if E_1 can be written to E_2 using COMM, ASSOC, L/R-ASSCOM

Soundness $\boxed{E_1 \simeq E_2} \Rightarrow \boxed{E_1 \equiv E_2}$ easy to check.

Completeness $\boxed{E_1 \simeq E_2} \Leftarrow \boxed{E_1 \equiv E_2}$ Does it hold??

None of the papers asks the completeness question!

Conflict Detection Approach

Overview

- The query is given by some initial operator tree E.
- Consider some DP algorithm:
  [Moerkotte et al., 2013] use DPsub
  [Birler and Neumann, 2025] use DPccp
- Add a conflict detector to prevent it from returning a plan $E' \neq E$.

From DPsub to DPsube

Algorithm 1: DPsub

for $\emptyset \subsetneq S \subseteq \boldsymbol{R}$ **do**
  for $\emptyset \subsetneq S_1 \subsetneq S$ **do**
    $S_2 := S - S_1$;
    if *Applicable*$(\circ, S_1, S_2)$ **then**
      $(P_1, c_1) :=$ PlanCost$[S_1]$
      $(P_2, c_2) :=$ PlanCost$[S_2]$
      $c := c_1 + c_2 +$ Cost$(P_1 \bowtie P_2)$;
      if $c <$ *PlanCost*$[S_1 \cup S_2].cost$ **then**
        PlanCost$[S_1 \cup S_2] := (P_1 \bowtie P_2, c)$

Need to design Applicable$(\circ, S_1, S_2)$.

Soundness and Completeness of Dynamic Programming

The function Applicable$(\circ, S_1, S_2)$ is sound if for any input E, DPsub returns an optimized E' s.t. $E \simeq E'$.

The function Applicable$(\circ, S_1, S_2)$ is complete if, for any input E, if $E' \simeq E$ and E' is optimal, then DPsub returns E'.

Soundness is necessary. Completeness is nice to have.

Conflict Detection (CD)

Applicable$(\circ, S_1, S_2)$ is called a conflict detector, CD.

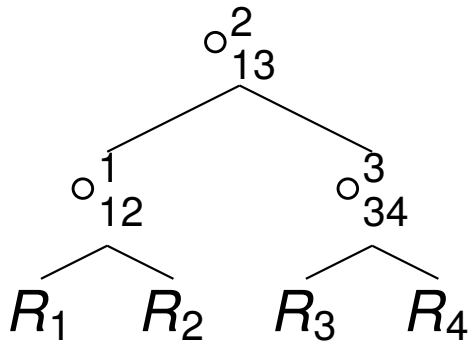
[Moerkotte et al., 2013] define 3 variants: CD-A, CD-B, CD-C.

[Birler and Neumann, 2025] define CD-E, a more efficient variant of CD-C.

Assumptions

The input is given by:

- Relations $R_1, R_2, \ldots, R_n$
- Initial operator tree E
- Operators $\circ^1, \ldots, \circ^{n-1}$
- $\mathcal{A}_T(\circ^1), \ldots, \mathcal{A}_T(\circ^{n-1})$.



$\mathcal{A}_T(\circ^2_{13}) = \{R_1, R_3\}$, etc.

Syntactic Eligibility Set: SES

$$\boxed{\mathrm{SES}(\circ_p) \stackrel{\text{def}}{=} \mathcal{A}_T(p)}$$

[1] $(R \bowtie S) \bowtie_{R.a+S.b=T.c+K.d} (T \bowtie K)$ requires R, S on the left, T, K on the right.

Syntactic Eligibility Set: SES

$$\boxed{\text{SES}(\circ_p) \stackrel{\text{def}}{=} \mathcal{A}_T(p)}$$

$$\text{L-SES}(\circ_p) \stackrel{\text{def}}{=} \text{SES}(\circ_p) \cap \mathcal{T}(\text{left}(\circ_p))$$
$$\text{R-SES}(\circ_p) \stackrel{\text{def}}{=} \text{SES}(\circ_p) \cap \mathcal{T}(\text{right}(\circ_p))$$

where $\text{left}(\circ_p)$, $\text{right}(\circ_p)$ refer to the initial operator tree.

[1] $(R \bowtie S) \bowtie_{R.a+S.b=T.c+K.d} (T \bowtie K)$ requires R, S on the left, T, K on the right.

Syntactic Eligibility Set: SES

$$\boxed{\text{SES}(\circ_p) \stackrel{\text{def}}{=} \mathcal{A}_T(p)}$$

$$\text{L-SES}(\circ_p) \stackrel{\text{def}}{=} \text{SES}(\circ_p) \cap \mathcal{T}(\text{left}(\circ_p))$$
$$\text{R-SES}(\circ_p) \stackrel{\text{def}}{=} \text{SES}(\circ_p) \cap \mathcal{T}(\text{right}(\circ_p))$$

where $\text{left}(\circ_p)$, $\text{right}(\circ_p)$ refer to the initial operator tree.

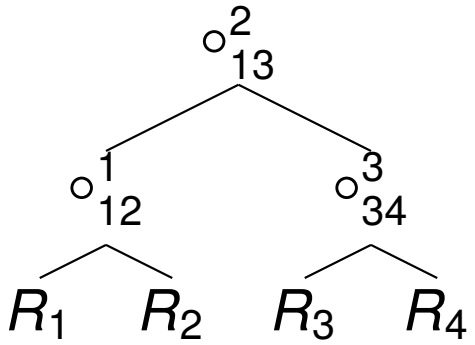
In any CD, Applicable$(\circ, S_1, S_2)$ must check:[1]

$$\boxed{\text{L-SES}(\circ) \subseteq S_1 \text{ and } \text{R-SES}(\circ) \subseteq S_2}$$

[1] $(R \bowtie S) \bowtie_{R.a+S.b=T.c+K.d} (T \bowtie K)$ requires R, S on the left, T, K on the right.

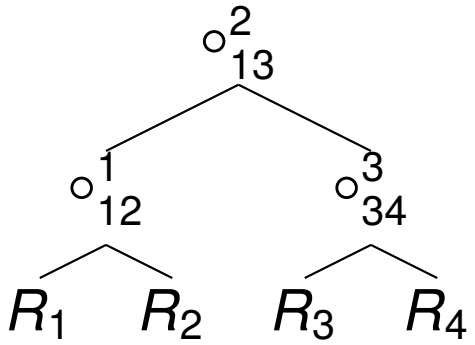
Example of Using SES

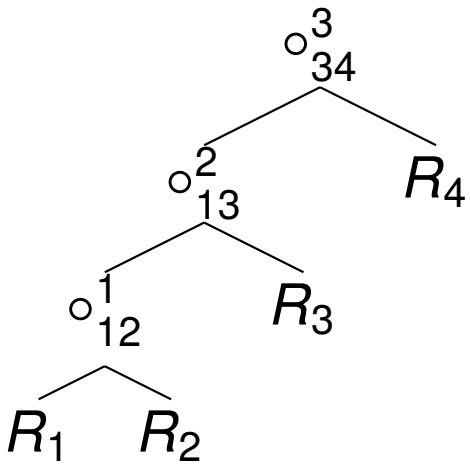
Initial operator tree:



Example of Using SES

Initial operator tree:

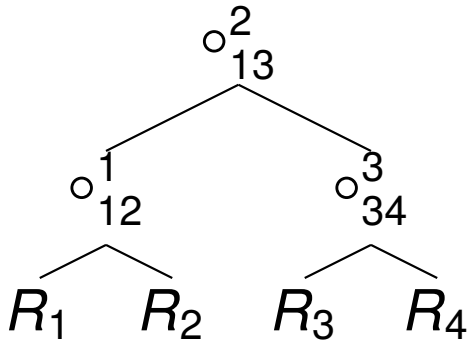


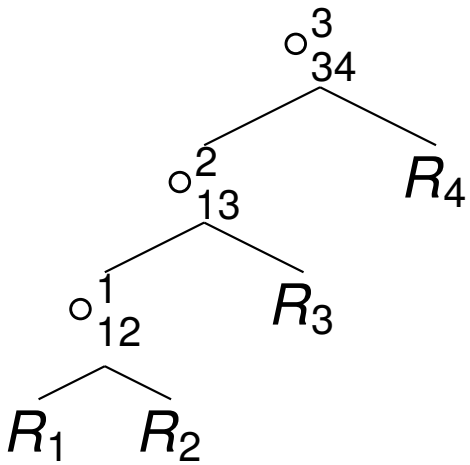


SES OK?

Example of Using SES

Initial operator tree:

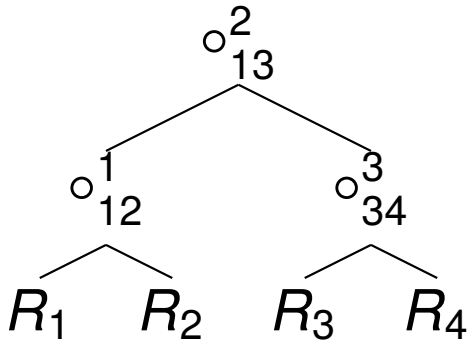


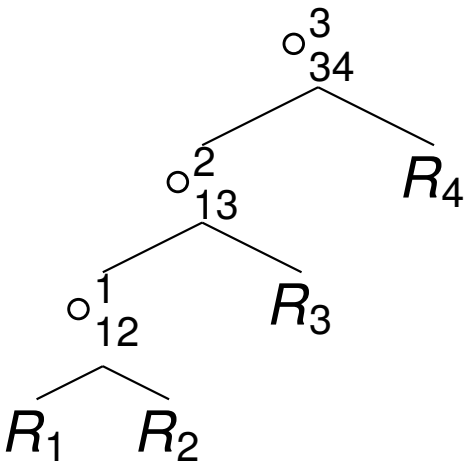


SES OK? YES

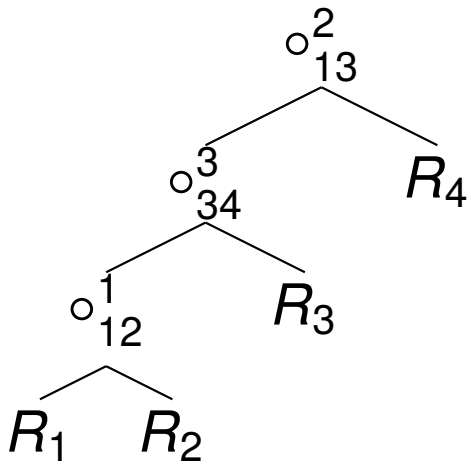
Example of Using SES

Initial operator tree:





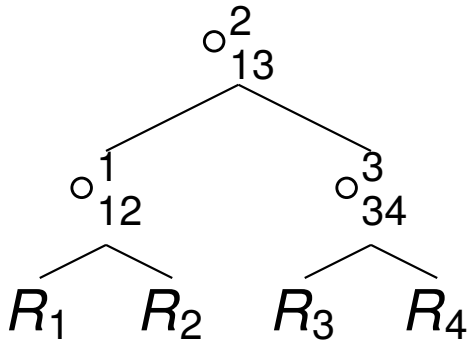
SES OK? YES

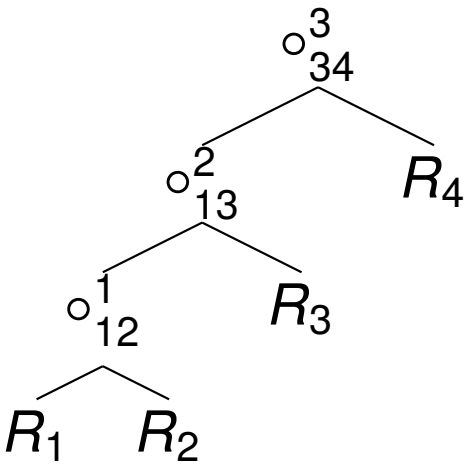


SES OK?

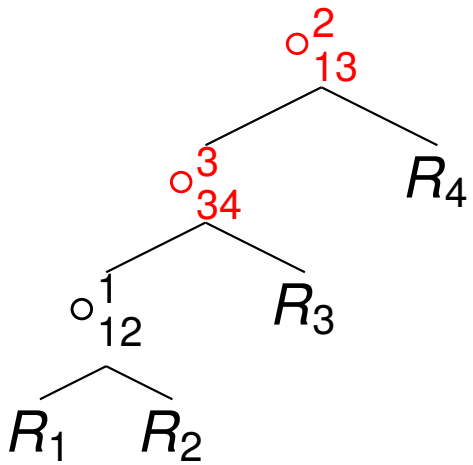
Example of Using SES

Initial operator tree:





SES OK? YES



SES OK? NO

CD-A

Overview

Extend SES to a larger set: TES = Total Eligibility Set.

Applicable$(\circ, S_1, S_2)$ checks:

$$\boxed{\text{L-TES}(\circ) \subseteq S_1 \text{ and } \text{R-TES}(\circ) \subseteq S_2}$$

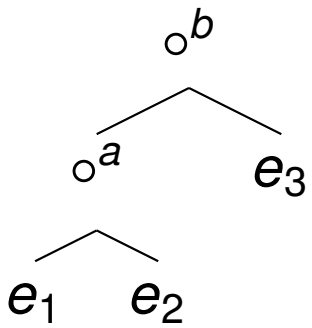
Will define TES next, as used by CD-A.

TES

Start with $\boxed{\text{TES}(\circ) := \text{SES}(\circ)}$ Then:

TES

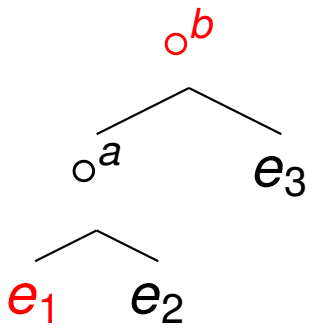
Start with $\boxed{\text{TES}(\circ) := \text{SES}(\circ)}$ Then:



Initial tree.

TES

Start with $\boxed{\text{TES}(\circ) := \text{SES}(\circ)}$ Then:



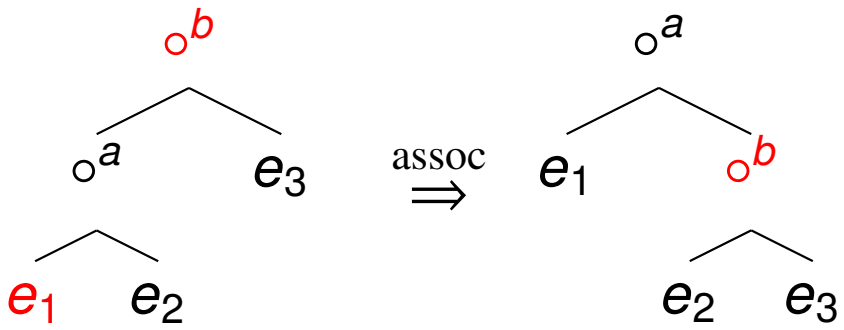
Initial tree.

If $\neg\text{assoc}(\circ^a, \circ^b)$ then

$\boxed{\text{TES}(\circ^b) \cup= \mathcal{T}(e_1)}$

TES

Start with $\boxed{\text{TES}(\circ) := \text{SES}(\circ)}$ Then:



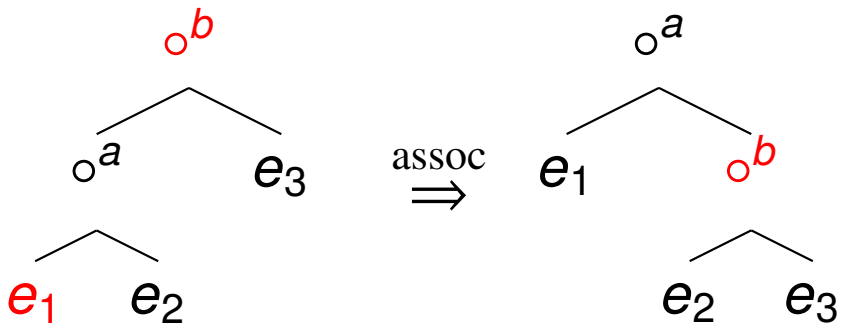
Initial tree. Conflict.

If $\neg\text{assoc}(\circ^a, \circ^b)$ then

$\boxed{\text{TES}(\circ^b) \cup= \mathcal{T}(e_1)}$

TES

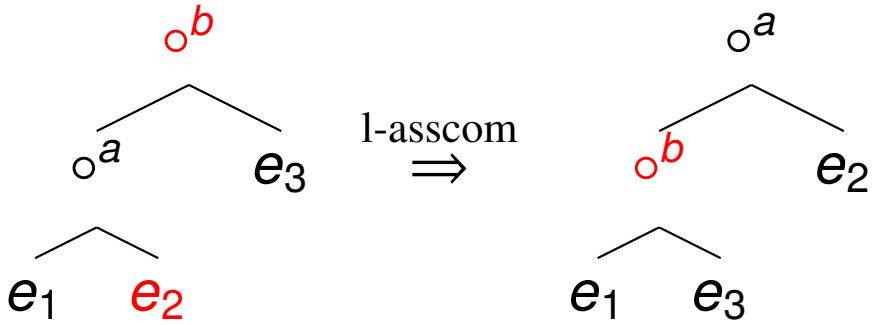
Start with $\boxed{\text{TES}(\circ) := \text{SES}(\circ)}$ Then:



Initial tree. Conflict.

If $\neg\text{assoc}(\circ^a, \circ^b)$ then

$\boxed{\text{TES}(\circ^b) \cup= \mathcal{T}(e_1)}$



If $\neg\text{l-asscom}(\circ^a, \circ^b)$ then

$\boxed{\text{TES}(\circ^b) \cup= \mathcal{T}(e_2)}$

From the Paper [Moerkotte et al., 2013]

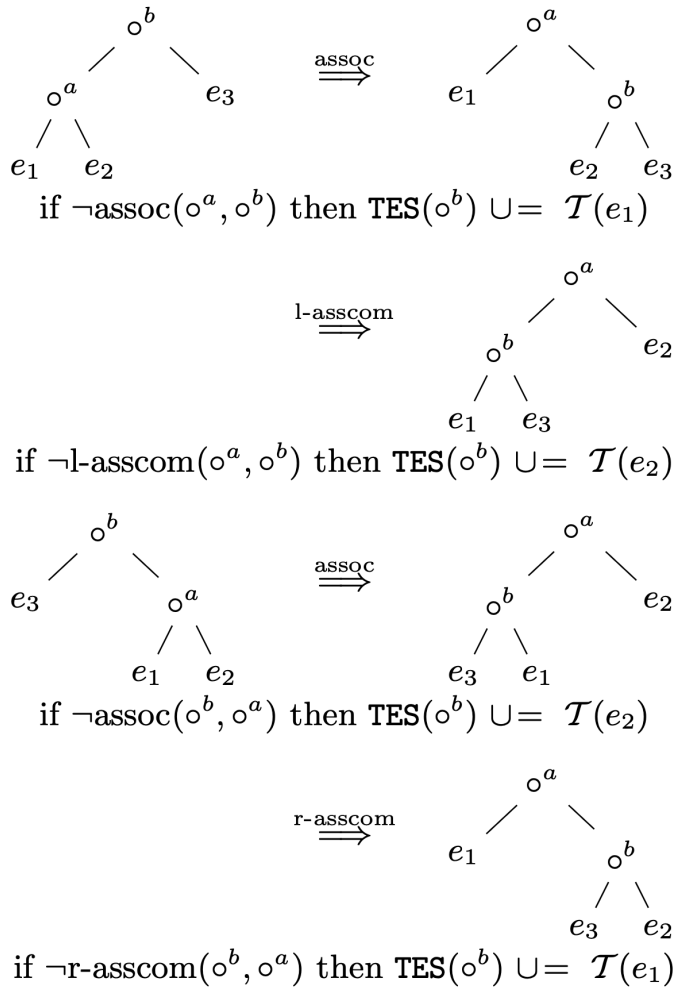


Figure 5: Calculating TES for simple operator trees

CD-A Summary

Start by computing TES.

CD-A Summary

Start by computing TES.

CD-A($\circ^b$)

▷ **Input:** operator $\circ^b$

1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\ \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\ \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$

CD-A Summary

Start by computing TES.

STO = subtree operators
$\circ^a$ = any descendent of $\circ^b$

CD-A($\circ^b$)

▷ **Input:** operator $\circ^b$
1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\, \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\, \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$

CD-A Summary

Start by computing TES.

STO = subtree operators
$\circ^a$ = any descendent of $\circ^b$

CD-A($\circ^b$)

▷ **Input:** operator $\circ^b$

1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\, \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\, \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{TES}(\circ^b) \leftarrow \texttt{TES}(\circ^b) \cup \mathcal{T}(\text{left}(\circ^a))$

Finally:

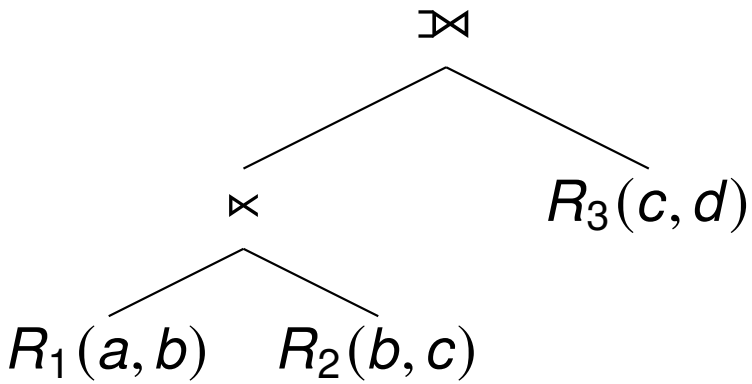
$\textsc{applicable}_A(\circ, S_1, S_2)$

▷ **Input:** binary operator $\circ$, set of tables S_1, S_2

1 **return** $\texttt{L-TES}(\circ) \subseteq S_1 \wedge \texttt{R-TES}(\circ) \subseteq S_2$

Example

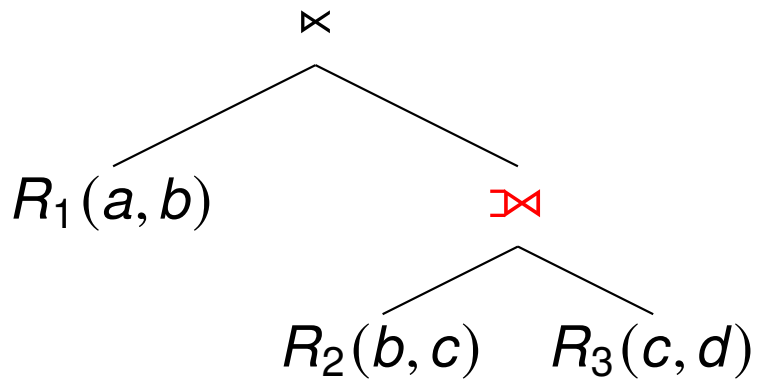
Initial operator tree:



$\neg\text{assoc}(\ltimes, \rtimes)$

$\text{TES}(\rtimes) \cup= \{R_1\}$

Candidate plan:



$\neg$APPLICABLE

Discussion

- The paper claims that CD-A is "correct", i.e. that it is sound.
- The claim may be true, but the proof provided is bogus.
- Another concern: can DPsub get stuck? Maybe the optimal plan for S_1 is such, for all S_2, $\neg$Applicable$(S_1, S_2, \circ)$. This is not discussed in either [Moerkotte et al., 2013, Birler and Neumann, 2025].
- The paper shows that CD-A is incomplete (next).

Incompleteness of CD-A

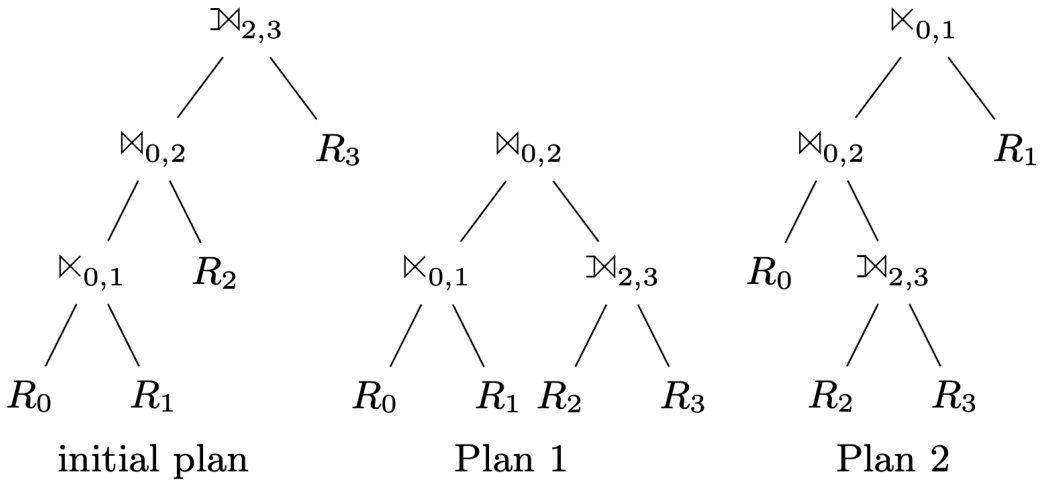


Figure 6: Example for incompleteness of CD-A

Incompleteness of CD-A

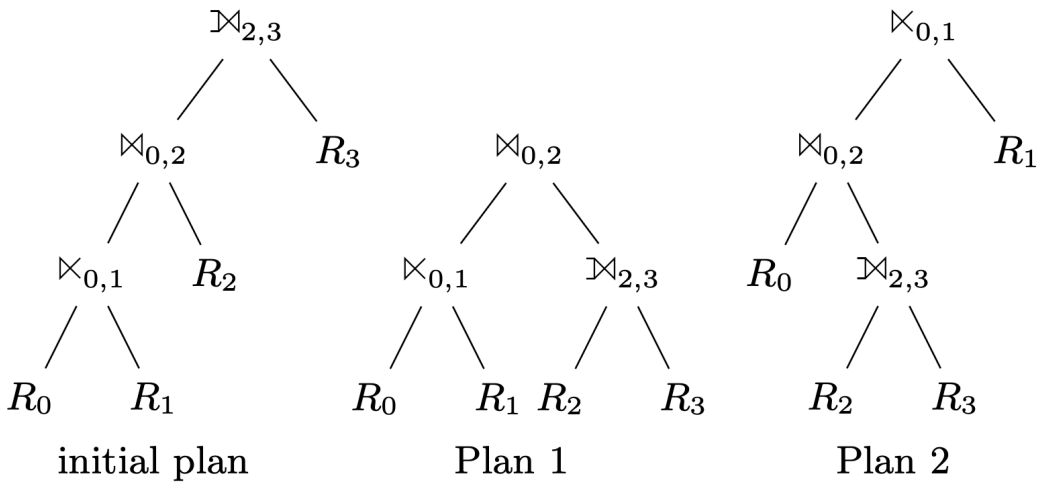


Figure 6: Example for incompleteness of CD-A

In class: prove $\boxed{\text{Initial-Plan} \equiv \text{Plan-1} \equiv \text{Plan-2}}$

Incompleteness of CD-A

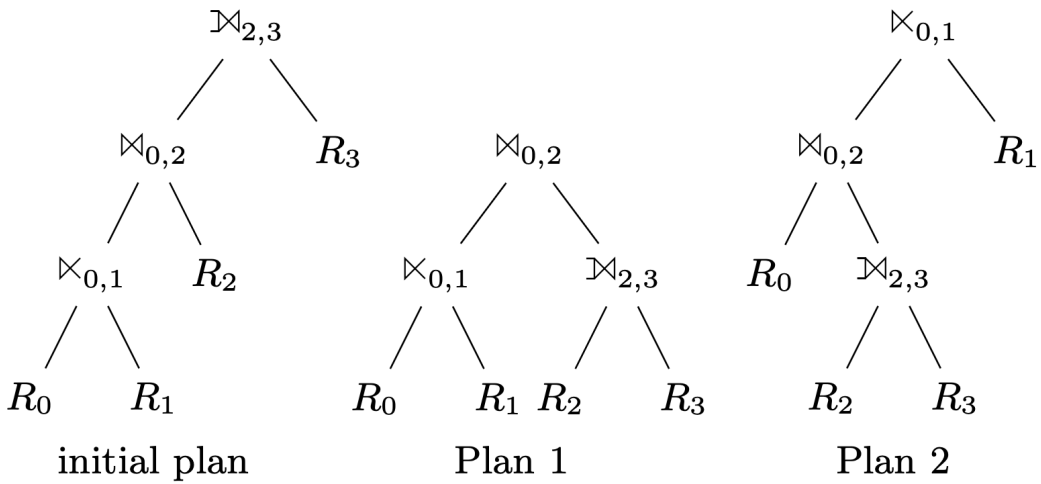


Figure 6: Example for incompleteness of CD-A

In class: prove $\boxed{\text{Initial-Plan} \equiv \text{Plan-1} \equiv \text{Plan-2}}$

$\neg\text{assoc}(\ltimes_{0,1}, \rtimes_{2,3})$ implies $R_0 \in \text{TES}(\rtimes_{2,3})$: Plans 1&2 have conflicts.

CD-B

Overview

Replace TES($\circ$) = set of tables T_2, with CD($\circ$) = set of rules $T_1 \rightarrow T_2$

Overview

Replace TES($\circ$) = set of tables T_2, with CD($\circ$) = set of rules $T_1 \rightarrow T_2$

Applicable($\circ, S_1, S_2$) checks:

- Check SES as before:

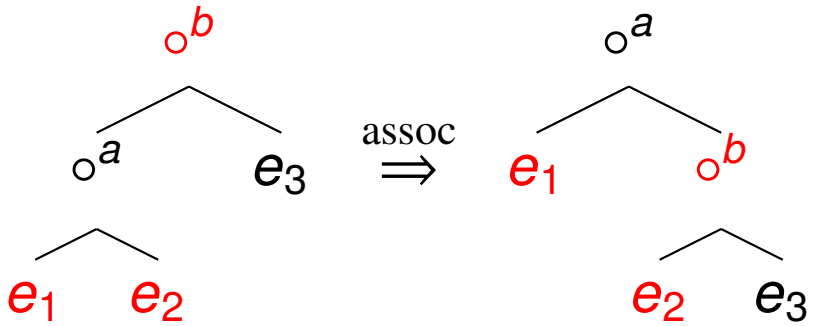
$$\boxed{\text{L-SES}(\circ) \subseteq S_1, \text{R-SES}(\circ) \subseteq S_2}$$

- For all $T_1 \rightarrow T_2 \in \text{CD}(\circ)$, check:

$$\boxed{\text{If } T_1 \cap S \neq \emptyset \text{ then } T_2 \subseteq S}$$

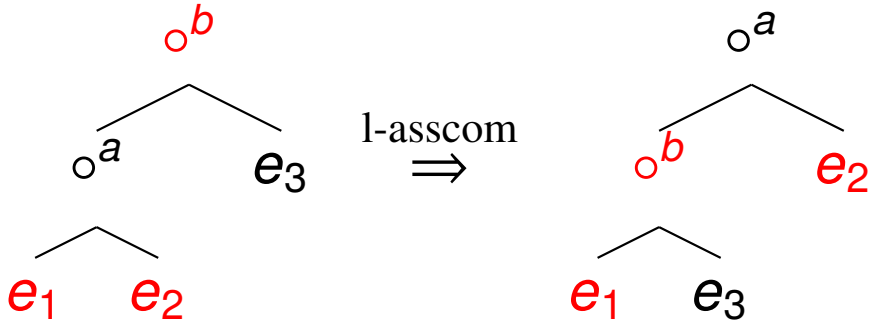
where $S = S_1 \cup S_2$.

CD-B



If $\neg\text{assoc}(\circ^a, \circ^b)$ then

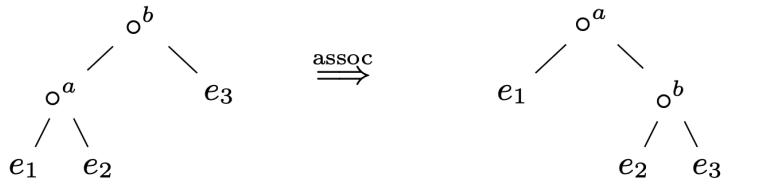
$$\text{CR}(\circ^b)+ = \mathcal{T}(e_2) \rightarrow \mathcal{T}(e_1)$$



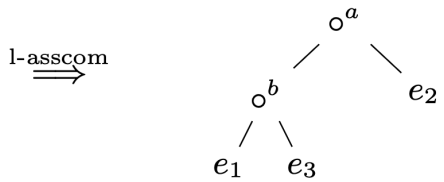
If $\neg\text{l-asscom}(\circ^a, \circ^b)$ then

$$\text{TES}(\circ^b)+ = \mathcal{T}(e_1) \rightarrow \mathcal{T}(e_2)$$

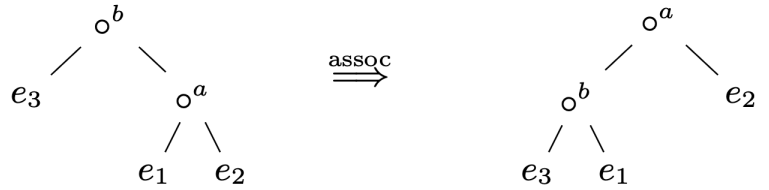
From the Paper [Moerkotte et al., 2013]



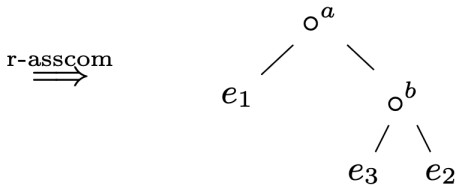
if $\neg\text{assoc}(\circ^a, \circ^b)$ then $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(e_2) \rightarrow \mathcal{T}(e_1)$



if $\neg\text{l-asscom}(\circ^a, \circ^b)$ then $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(e_1) \rightarrow \mathcal{T}(e_2)$



$\neg\text{assoc}(\circ^b, \circ^a)$ then $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(e_1) \rightarrow \mathcal{T}(e_2)$



if $\neg\text{r-asscom}(\circ^b, \circ^a)$ then $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(e_2) \rightarrow \mathcal{T}(e_1)$

CD-B Summary

Start by computing CD.

CD-B Summary

Start by computing CD.

CD-B($\circ^b$)

▷ **Input:** operator $\circ^b$

1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\ \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\ \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$

Figure 8: Pseudocode for CD-B

CD-B Summary

Start by computing CD.

$\circ^a$ = any descendant of $\circ^a$.

CD-B($\circ^b$)

▷ **Input:** operator $\circ^b$
1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\ \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\ \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$

Figure 8: Pseudocode for CD-B

CD-B Summary

Start by computing CD.

$\circ^a$ = any descendant of $\circ^a$.

Finally

CD-B($\circ^b$)

▷ **Input:** operator $\circ^b$

1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\ \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\ \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{CR}(\circ^b)\ +=\ \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$

Figure 8: Pseudocode for CD-B

$\textsc{applicable}_{B/C}(\circ, S_1, S_2)$

▷ **Input:** binary operator $\circ$, set of tables S_1, S_2

1 **if** $\texttt{L-TES}(\circ) \subseteq S_1 \wedge \texttt{R-TES}(\circ) \subseteq S_2$
2 **for all** $(T_1 \rightarrow T_2) \in \texttt{CR}(\circ)$
3 **if** $T_1 \cap (S_1 \cup S_2) \neq \emptyset$
4 **if** $T_2 \not\subseteq (S_1 \cup S_2)$
5 **return** FALSE
6 **return** TRUE
7 **else**
8 **return** FALSE

Figure 9: Pseudocode for $\textsc{applicable}_{B/C}$

CD-B Summary

Start by computing CD.

$\circ^a$ = any descendant of $\circ^a$.

Finally

Warning:
here TES:=SES

CD-B($\circ^b$)

▷ **Input:** operator $\circ^b$
1 $\texttt{TES}(\circ^b) \leftarrow \textsc{Calc}_{SES}(\circ^b)$
2 **for** $\forall\ \circ^a \in \texttt{STO}(\text{left}(\circ^b))$
3 **if** $\neg\text{assoc}(\circ^a, \circ^b)$
4 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$
5 **if** $\neg\text{l-asscom}(\circ^a, \circ^b)$
6 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
7 **for** $\forall\ \circ^a \in \texttt{STO}(\text{right}(\circ^b))$
8 **if** $\neg\text{assoc}(\circ^b, \circ^a)$
9 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{left}(\circ^a)) \rightarrow \mathcal{T}(\text{right}(\circ^a))$
10 **if** $\neg\text{r-asscom}(\circ^b, \circ^a)$
11 $\texttt{CR}(\circ^b) \mathrel{+}= \mathcal{T}(\text{right}(\circ^a)) \rightarrow \mathcal{T}(\text{left}(\circ^a))$

Figure 8: Pseudocode for CD-B

$\textsc{applicable}_{B/C}(\circ, S_1, S_2)$

▷ **Input:** binary operator $\circ$, set of tables S_1, S_2
1 **if** $\texttt{L-TES}(\circ) \subseteq S_1 \wedge \texttt{R-TES}(\circ) \subseteq S_2$
2 **for all** $(T_1 \rightarrow T_2) \in \texttt{CR}(\circ)$
3 **if** $T_1 \cap (S_1 \cup S_2) \neq \emptyset$
4 **if** $T_2 \not\subseteq (S_1 \cup S_2)$
5 **return** FALSE
6 **return** TRUE
7 **else**
8 **return** FALSE

Figure 9: Pseudocode for $\textsc{applicable}_{B/C}$

Example

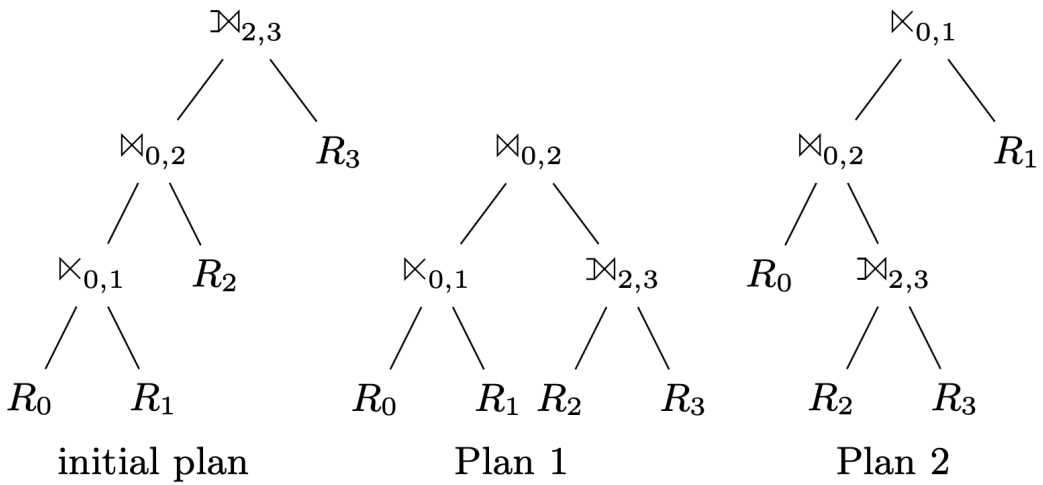


Figure 6: Example for incompleteness of CD-A

$\neg\text{assoc}(\ltimes_{0,1}, \rtimes_{2,3})$ implies $\text{CR}(\rtimes_{2,3})$ contains $R_1 \rightarrow R_0$

Both PLAN 1 and PLAN 2 are applicable.

Incompleteness of CD-B

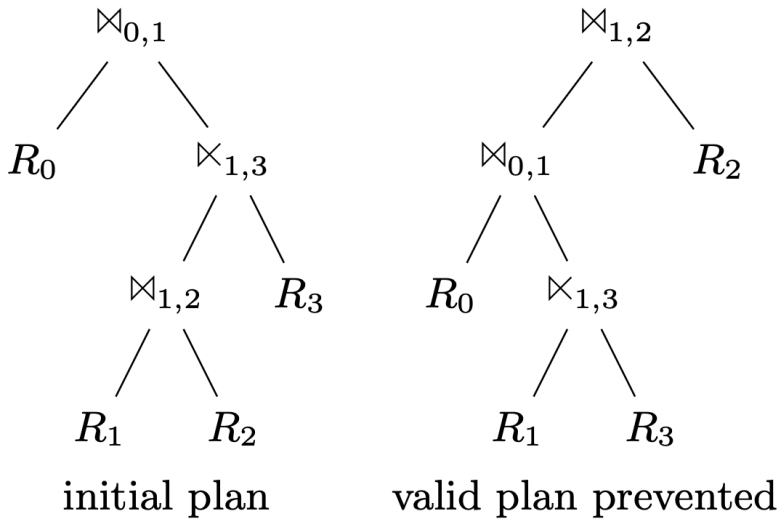


Figure 10: Example for incompleteness of CD-B

$\neg$r-asscom($\bowtie_{01}, \ltimes_{13}$) thus CR($\bowtie_{01}$) contains $R_3 \rightarrow R_1R_2$.

CD-C

CD-C

CD-C relaxes the rules of CD-B

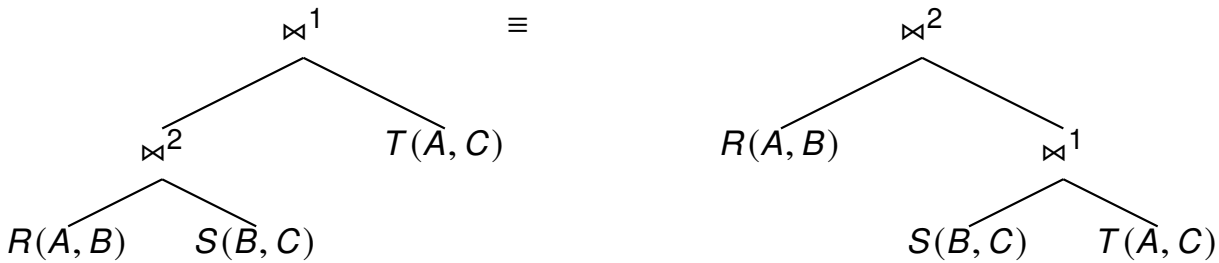
But we wont discuss them.

Paper claims that CD-C is complete; no proof is offered.

[Birler and Neumann, 2025] shows that CD-C fails to be complete for "arbitrary hypothetical operators". But may still be complete for the "usual" operators.

Discussion

- Non-reordable operators are complicated!!!
- CD-A/CD-B/CD-C appear incapable of handling simple cyclic queries:



Not permitted because $\mathrm{SES}(\bowtie^1) = \{R, S, T\}$.

- Soundness claims are credible, but proofs are bogus.
- Completeness proof of CD-C is missing.
- Progress of DPsub is not discussed at all.

Birler, A. and Neumann, T. (2025).

Efficient enumeration of the complete join search space.

In *Proceedings of the 19th International Symposium on Database Programming Languages, DBPL 2025, Berlin, Germany, June 22-27, 2025*, pages 5:1–5:12. ACM.

Moerkotte, G., Fender, P., and Eich, M. (2013).

On the correct and complete enumeration of the core search space.

In Ross, K. A., Srivastava, D., and Papadias, D., editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 493–504. ACM.

Moerkotte, G. and Neumann, T. (2008).

Dynamic programming strikes back.

In Wang, J. T., editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 539–552. ACM.