Efficient Algorithms for Frequently Asked Questions

6. Worst-Case Optimal Join Algorithms

Prof. Dan Olteanu

DaST
Data • (Systems+Theory)

University of Zurich[UZH]

March 21+28, 2022

https://lms.uzh.ch/url/RepositoryEntry/17185308706

What Makes a Join Algorithm Optimal?

Terminology

- Join = FAQ where all variables are free, i.e., no marginalisation
- Conjunctive query (CQ) = FAQ over the Boolean semiring
- Query output = Listing representation of all tuples in the query answer

We can reason about two types of output sizes for a join Φ

- Instance output size: The size of Φ's output for a specific input
- Worst-case output size: The maximum size of Φ's output for any input

Running time of optimal join algorithms is proportional to

- Input size (IN) plus output size (OUT) (Instance Optimality)
- Input size plus worst-case output size (Worst-Case Optimality)

Agenda for this Lecture

1. Instance optimality for free-connex acyclic CQs: Yannakakis's algorithm
   - Runtime becomes O(IN*OUT) for arbitrary acyclic CQs
   - This works for semirings with constant-size elements, e.g., sum-product
2. Worst-case optimality for arbitrary joins: LeapFrog TrieJoin algorithm
   - This only works when all variables are free, so no CQs
   - Instance optimality for cyclic joins not possible (unless P=NP)
3. Mainstream join algorithms are suboptimal for cyclic joins
4. Efficient processing of CQs with large output size

Next lecture: Deriving worst-case optimal size of join output

1. Computing Acyclic Conjunctive Queries using Yannakakis Algorithm

Recall: Evaluation Example for Acyclic CQ without Free Variables

$$\Phi() = \bigvee_{(x_1,\dots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1,x_2)\wedge\psi_{23}(x_2,x_3)\wedge\psi_{34}(x_3,x_4)\wedge\psi_{15}(x_1,x_5)$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

$$\Phi() = \bigvee_{(x_1,\dots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1,x_2) \wedge \psi_{23}(x_2,x_3) \wedge \psi_{34}(x_3,x_4) \wedge \psi_{15}(x_1,x_5)$$

A join tree for Φ:

$$\begin{tikzcd}[column sep=tiny]
 & \psi_{12}(x_1,x_2) \arrow[dl, no head] \arrow[dr, no head] & \\
\psi_{23}(x_2,x_3) \arrow[d, no head] & & \psi_{15}(x_1,x_5) \\
\psi_{34}(x_3,x_4) & &
\end{tikzcd}$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

$$\Phi() = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1,x_2) \wedge \psi_{23}(x_2,x_3) \wedge \psi_{34}(x_3,x_4) \wedge \psi_{15}(x_1,x_5)$$

A join tree for Φ:

$$\begin{tikzcd}
& \psi_{12}(x_1,x_2) \arrow[dl, dash] \arrow[dr, dash] & \\
\psi_{23}(x_2,x_3) \arrow[d, dash] & & \psi_{15}(x_1,x_5) \\
\psi_{34}(x_3,x_4) & &
\end{tikzcd}$$

We repeat how Φ can be evaluated efficiently on the next slide.

Recall: Evaluation Example for Acyclic CQ without Free Variables

Join tree of Φ:

$$
\begin{array}{ccc}
 & \psi_{12}(x_1, x_2) & \\
 \swarrow & & \searrow \\
\psi_{23}(x_2, x_3) & & \psi_{15}(x_1, x_5) \\
| & & \\
\psi_{34}(x_3, x_4) & &
\end{array}
$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

Join tree of Φ:

$$\begin{array}{ccc} & \psi_{12}(x_1, x_2) & \\ \swarrow & & \searrow \\ \psi_{23}(x_2, x_3) & & \psi_{15}(x_1, x_5) \\ | & & \\ \psi_{34}(x_3, x_4) & & \end{array}$$

$@\psi_{34}$ Send up its x_3-values:

$$V_{34\to 23}(x_3) = \bigvee_{x_4} \psi_{34}(x_3, x_4)$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

Join tree of Φ:

- $\psi_{12}(x_1, x_2)$
  - $\psi_{23}(x_2, x_3)$
    - $\psi_{34}(x_3, x_4)$
  - $\psi_{15}(x_1, x_5)$

@ψ_{34} Send up its x_3-values:

$$V_{34\to 23}(x_3) = \bigvee_{x_4} \psi_{34}(x_3, x_4)$$

@ψ_{23} Send up its x_2-values that are paired with x_3 common to $V_{34\to 23}(x_3)$ and ψ_{23}:

$$V_{23\to 12}(x_2) = \bigvee_{x_3} \psi_{23}(x_2, x_3) \wedge V_{34\to 23}(x_3)$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

Join tree of Φ:

$$
\begin{array}{ccc}
 & \psi_{12}(x_1, x_2) & \\
 \swarrow & & \searrow \\
\psi_{23}(x_2, x_3) & & \psi_{15}(x_1, x_5) \\
| & & \\
\psi_{34}(x_3, x_4) & &
\end{array}
$$

@ψ_{34} Send up its x_3-values:

$$V_{34\to 23}(x_3) = \bigvee_{x_4} \psi_{34}(x_3, x_4)$$

@ψ_{23} Send up its x_2-values that are paired with x_3 common to $V_{34\to 23}(x_3)$ and ψ_{23}:

$$V_{23\to 12}(x_2) = \bigvee_{x_3} \psi_{23}(x_2, x_3) \wedge V_{34\to 23}(x_3)$$

@ψ_{15} Send up its x_1-values:

$$V_{15\to 12}(x_1) = \bigvee_{x_5} \psi_{15}(x_1, x_5)$$

Recall: Evaluation Example for Acyclic CQ without Free Variables

Join tree of Φ:

$$\begin{array}{c} \psi_{12}(x_1, x_2) \\ \swarrow \quad \searrow \\ \psi_{23}(x_2, x_3) \quad \psi_{15}(x_1, x_5) \\ | \\ \psi_{34}(x_3, x_4) \end{array}$$

@ψ_{34} Send up its x_3-values:

$$V_{34\to 23}(x_3) = \bigvee_{x_4} \psi_{34}(x_3, x_4)$$

@ψ_{23} Send up its x_2-values that are paired with x_3 common to $V_{34\to 23}(x_3)$ and ψ_{23}:

$$V_{23\to 12}(x_2) = \bigvee_{x_3} \psi_{23}(x_2, x_3) \wedge V_{34\to 23}(x_3)$$

@ψ_{15} Send up its x_1-values:

$$V_{15\to 12}(x_1) = \bigvee_{x_5} \psi_{15}(x_1, x_5)$$

@ψ_{12} Is there a pair (x_1, x_2) of ψ_{12} with x_1 also in $V_{15\to 12}$ and x_2 also in $V_{23\to 12}$?

$$\Phi() = \bigvee_{x_1, x_2} \psi_{12}(x_1, x_2) \wedge V_{15\to 12}(x_1) \wedge V_{23\to 12}(x_2)$$

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{\mathcal{S} \in \mathcal{E}} \psi_{\mathcal{S}}(\mathbf{x}_{\mathcal{S}}) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

1. Initialisation: Create a view $V_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

1. Initialisation: Create a view $V_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Repeatedly transfer information from leaf to parent and delete leaf

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

1. Initialisation: Create a view $V_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Repeatedly transfer information from leaf to parent and delete leaf

   Pick a leaf $\psi_L(\mathbf{x}_L)$ with parent $\psi_P(\mathbf{x}_P)$ in $\mathcal{T}$
   Propagate information from leaf to parent and remove the leaf from $\mathcal{T}$
   Marginalise out variables of the leaf that are not in parent

$$V_{L\to P}(\mathbf{x}_{L\cap P}) = \bigoplus_{i \in L \setminus P : x_i} V_L(\mathbf{x}_L)$$

$$V_P(\mathbf{x}_P) := V_P(\mathbf{x}_P) \otimes V_{L\to P}(\mathbf{x}_{L\cap P})$$

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

1. Initialisation: Create a view $V_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Repeatedly transfer information from leaf to parent and delete leaf

   Pick a leaf $\psi_L(\mathbf{x}_L)$ with parent $\psi_P(\mathbf{x}_P)$ in $\mathcal{T}$
   Propagate information from leaf to parent and remove the leaf from $\mathcal{T}$
   Marginalise out variables of the leaf that are not in parent

   $$V_{L\to P}(\mathbf{x}_{L\cap P}) = \bigoplus_{i \in L \setminus P : x_i} V_L(\mathbf{x}_L)$$

   $$V_P(\mathbf{x}_P) := V_P(\mathbf{x}_P) \otimes V_{L\to P}(\mathbf{x}_{L\cap P})$$

3. Marginalise out remaining variables and output at root from view $V_R(\mathbf{x}_R)$

   $$\Phi() = \bigoplus_{\mathbf{x}_R} V_R(\mathbf{x}_R)$$

Yannakakis's Algorithm for Acyclic CQs without Free Variables

$$\Phi() = \bigoplus_{\mathbf{x}} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \text{ with acyclic hypergraph } \mathcal{H} \text{ and join tree } \mathcal{T}$$

Yannakakis's algorithm uses a bottom-up evaluation strategy over the join tree

1. Initialisation: Create a view $V_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Repeatedly transfer information from leaf to parent and delete leaf

   Pick a leaf $\psi_L(\mathbf{x}_L)$ with parent $\psi_P(\mathbf{x}_P)$ in $\mathcal{T}$
   Propagate information from leaf to parent and remove the leaf from $\mathcal{T}$
   Marginalise out variables of the leaf that are not in parent

$$V_{L \to P}(\mathbf{x}_{L \cap P}) = \bigoplus_{i \in L \setminus P : x_i} V_L(\mathbf{x}_L)$$

$$V_P(\mathbf{x}_P) := V_P(\mathbf{x}_P) \otimes V_{L \to P}(\mathbf{x}_{L \cap P})$$

3. Marginalise out remaining variables and output at root from view $V_R(\mathbf{x}_R)$

$$\Phi() = \bigoplus_{\mathbf{x}_R} V_R(\mathbf{x}_R)$$

Time complexity: Linear in the size of the input factors (after sorting them)

Yannakakis's Algorithm for Acyclic CQs with Free Variables

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{(x_{f+1},\ldots,x_n)} \bigotimes_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S), \text{ where } X_1,\ldots,X_f \text{ are free variables}$$

Yannakakis's Algorithm for Acyclic CQs with Free Variables

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{(x_{f+1},\ldots,x_n)} \bigotimes_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S), \text{ where } X_1,\ldots,X_f \text{ are free variables}$$

The output size may be non-linear in the size of the input

Yannakakis's Algorithm for Acyclic CQs with Free Variables

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{(x_{f+1},\ldots,x_n)} \bigotimes_{S\in\mathcal{E}} \psi_S(\mathbf{x}_S), \text{ where } X_1,\ldots,X_f \text{ are free variables}$$

The output size may be non-linear in the size of the input

Adaptation of Steps 2 and 3: do NOT marginalise out free variables

2. We marginalise out $L' = (L \setminus P) \setminus [f]$ and keep $P' = P \cup (L \cap [f])$

$$V_{L\to P'}(\mathbf{x}_{L\cap P'}) := \bigoplus_{i\in L':x_i} V_L(\mathbf{x}_L)$$

$$V_{P'}(\mathbf{x}_{P'}) := V_P(\mathbf{x}_P) \otimes V_{L\to P'}(\mathbf{x}_{L\cap P'})$$

3. Marginalise out non-free variables and output at root from view $V_R(\mathbf{x}_R)$

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{i\in R\setminus[f]:x_i} V_R(\mathbf{x}_R)$$

Yannakakis's Algorithm for Acyclic CQs with Free Variables

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{(x_{f+1},\ldots,x_n)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S), \text{ where } X_1, \ldots, X_f \text{ are free variables}$$

The output size may be non-linear in the size of the input

Adaptation of Steps 2 and 3: do NOT marginalise out free variables

2. We marginalise out $L' = (L \setminus P) \setminus [f]$ and keep $P' = P \cup (L \cap [f])$

$$V_{L \to P'}(\mathbf{x}_{L \cap P'}) := \bigoplus_{i \in L' : x_i} V_L(\mathbf{x}_L)$$

$$V_{P'}(\mathbf{x}_{P'}) := V_P(\mathbf{x}_P) \otimes V_{L \to P'}(\mathbf{x}_{L \cap P'})$$

3. Marginalise out non-free variables and output at root from view $V_R(\mathbf{x}_R)$

$$\Phi(\mathbf{x}_{[f]}) = \bigoplus_{i \in R \setminus [f] : x_i} V_R(\mathbf{x}_R)$$

Question: Since $P' \supseteq P$, $V_{P'}$ may be larger than the input, yet how much larger?

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5) \in \prod_{i \in [5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2) — \psi_{23}(x_2, x_3) — \psi_{345}(x_3, x_4, x_5)$

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$ — $\psi_{345}(x_3, x_4, x_5)$

ψ_{12}

X_1	X_2
a_0	b_0
a_1	b_1

ψ_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$ — $\psi_{345}(x_3, x_4, x_5)$

V_{12}

X_1	X_2
a_0	b_0
a_1	b_1

V_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

V_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$ — $\psi_{345}(x_3, x_4, x_5)$

V_{12}

X_1	X_2
a_0	b_0
a_1	b_1

V_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

V_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$ — $\psi_{345}(x_3, x_4, x_5)$

V_{12}

X_1	X_2
a_0	b_0
a_1	b_1

V_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

$V_{345\to 234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$ — $\psi_{345}(x_3, x_4, x_5)$

V_{12}

X_1	X_2
a_0	b_0
a_1	b_1

V_{234}

X_2	X_3	X_4
b_1	c_1	d_1
b_2	c_2	d_2
$\cdots$	$\cdots$	$\cdots$
b_2	c_2	d_N
$\cdots$	$\cdots$	$\cdots$
b_N	c_2	d_2
$\cdots$	$\cdots$	$\cdots$
b_N	c_2	d_N

$V_{345\to 234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$

V_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

V_{234}	X_2	X_3	X_4
	b_1	c_1	d_1
	b_2	c_2	d_2
	$\cdots$	$\cdots$	$\cdots$
	b_2	c_2	d_N
	$\cdots$	$\cdots$	$\cdots$
	b_N	c_2	d_2
	$\cdots$	$\cdots$	$\cdots$
	b_N	c_2	d_N

$V_{345\to 234}$	X_3	X_4
	c_1	d_1
	c_2	d_2
	c_2	d_3
	$\cdots$	$\cdots$
	c_2	d_N

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]}\mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$

V_{12}

X_1	X_2
a_0	b_0
a_1	b_1

$V_{234\to 124}$

X_2	X_4
b_1	d_1
b_2	d_1
$\cdots$	$\cdots$
b_2	d_N
$\cdots$	$\cdots$
b_N	d_1
$\cdots$	$\cdots$
b_N	d_N

$V_{345\to 234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$ — $\psi_{23}(x_2, x_3)$

V_{124}

X_1	X_2	X_4
a_1	b_1	d_1

$V_{234\to 124}$

X_2	X_4
b_1	d_1
b_2	d_1
$\cdots$	$\cdots$
b_2	d_N
$\cdots$	$\cdots$
b_N	d_1
$\cdots$	$\cdots$
b_N	d_N

$V_{345\to 234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\dots,x_5)\in\prod_{i\in[5]} \mathrm{Dom}(X_i)} \psi_{12}(x_1, x_2) \land \psi_{23}(x_2, x_3) \land \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$

V_{124}

X_1	X_2	X_4
a_1	b_1	d_1

$V_{234\to124}$

X_2	X_4
b_1	d_1
b_2	d_1
$\cdots$	$\cdots$
b_2	d_N
$\cdots$	$\cdots$
b_N	d_1
$\cdots$	$\cdots$
b_N	d_N

$V_{345\to234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Φ

X_1	X_4
a_1	d_1

Evaluation Example for a Conjunctive Query

$$\Phi(x_1, x_4) = \bigvee_{(x_1,\ldots,x_5)\in\prod_{i\in[5]} \mathsf{Dom}(X_i)} \psi_{12}(x_1, x_2) \wedge \psi_{23}(x_2, x_3) \wedge \psi_{345}(x_3, x_4, x_5)$$

A join tree for Φ: $\psi_{12}(x_1, x_2)$

V_{124}

X_1	X_2	X_4
a_1	b_1	d_1

$V_{234\to124}$

X_2	X_4
b_1	d_1
b_2	d_1
$\cdots$	$\cdots$
b_2	d_N
$\cdots$	$\cdots$
b_N	d_1
$\cdots$	$\cdots$
b_N	d_N

$V_{345\to234}$

X_3	X_4
c_1	d_1
c_2	d_2
c_2	d_3
$\cdots$	$\cdots$
c_2	d_N

Φ

X_1	X_4
a_1	d_1

Problem: Intermediate results are of quadratic size!

Reducing the Size of Intermediate Results

Consider again the following join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

ψ_{23}	X_2	X_3
	b_0	c_0
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

Reducing the Size of Intermediate Results

Consider again the following join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}

X_1	X_2
a_0	b_0
a_1	b_1

ψ_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

No tuple (b_i, c_2) of ψ_{23} is in the join result: ψ_{12} has no matching tuple

Reducing the Size of Intermediate Results

Consider again the following join tree and factors:

$$\psi_{12}(x_1, x_2) \; — \; \psi_{23}(x_2, x_3) \; — \; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

ψ_{23}	X_2	X_3
	b_0	c_0
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

No tuple (b_i, c_2) of ψ_{23} is in the join result: ψ_{12} has no matching tuple

Tuple (a_0, b_0) of ψ_{12} is not in the join result: ψ_{345} has no matching tuple

Reducing the Size of Intermediate Results

Consider again the following join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}

X_1	X_2
a_0	b_0
a_1	b_1

ψ_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

No tuple (b_i, c_2) of ψ_{23} is in the join result: ψ_{12} has no matching tuple

Tuple (a_0, b_0) of ψ_{12} is not in the join result: ψ_{345} has no matching tuple

These are examples of dangling tuples

Reducing the Size of Intermediate Results

Consider again the following join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

ψ_{23}	X_2	X_3
	b_0	c_0
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

No tuple (b_i, c_2) of ψ_{23} is in the join result: ψ_{12} has no matching tuple

Tuple (a_0, b_0) of ψ_{12} is not in the join result: ψ_{345} has no matching tuple

These are examples of dangling tuples

Adaptation: remove all dangling tuples at each factor *before* we do the join

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

1. Initialisation: Create a view $R_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

1. Initialisation: Create a view $R_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$

2. Remove dangling tuples bottom-up

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

1. Initialisation: Create a view $R_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Remove dangling tuples bottom-up

   In bottom-up traversal of $\mathcal{T}$, filter each node ψ_P using its child ψ_C

   Remove tuples from R_P with no match in R_C

   $$R_P(\mathbf{x}_P) := R_P(\mathbf{x}_P) \otimes \mathbf{1}_{\bigoplus_{i \in C \setminus P : x_i} R_C(\mathbf{x}_C)}$$

   Indicator $\mathbf{1}_\Psi$: Returns $\mathbf{1}$ if $\Psi \neq \mathbf{0}$ and $\mathbf{0}$ otherwise

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

1. Initialisation: Create a view $R_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Remove dangling tuples bottom-up

   In bottom-up traversal of $\mathcal{T}$, filter each node ψ_P using its child ψ_C

   Remove tuples from R_P with no match in R_C

   $$R_P(\mathbf{x}_P) := R_P(\mathbf{x}_P) \otimes \mathbf{1}_{\bigoplus_{i \in C \setminus P : x_i} R_C(\mathbf{x}_C)}$$

   Indicator $\mathbf{1}_\Psi$: Returns $\mathbf{1}$ if $\Psi \neq \mathbf{0}$ and $\mathbf{0}$ otherwise

3. Remove dangling tuples top-down

Reducing the Size of Intermediate Results: Full Reducer

Fully reduce input factors along a join tree $\mathcal{T}$

1. Initialisation: Create a view $R_S(\mathbf{x}_S) = \psi_S(\mathbf{x}_S)$ for every $S \in \mathcal{E}$
2. Remove dangling tuples bottom-up

   In bottom-up traversal of $\mathcal{T}$, filter each node ψ_P using its child ψ_C

   Remove tuples from R_P with no match in R_C

   $$R_P(\mathbf{x}_P) := R_P(\mathbf{x}_P) \otimes \mathbf{1}_{\bigoplus_{i \in C \setminus P : x_i} R_C(\mathbf{x}_C)}$$

   Indicator $\mathbf{1}_\Psi$: Returns $\mathbf{1}$ if $\Psi \neq \mathbf{0}$ and $\mathbf{0}$ otherwise
3. Remove dangling tuples top-down

   In top-down traversal of $\mathcal{T}$, filter each node ψ_C using its parent ψ_P

   Remove tuples from R_C with no match in R_P

   $$R_C(\mathbf{x}_C) := R_C(\mathbf{x}_C) \otimes \mathbf{1}_{\bigoplus_{i \in P \setminus C : x_i} R_P(\mathbf{x}_P)}$$

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}

X_1	X_2
a_0	b_0
a_1	b_1

ψ_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

R_{12}

X_1	X_2
a_0	b_0
a_1	b_1

R_{23}

X_2	X_3
b_0	c_0
b_1	c_1
b_2	c_2
b_3	c_2
$\cdots$	$\cdots$
b_N	c_2

R_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

R_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

R_{12}	X_1	X_2
	a_0	b_0
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

R_{12}	X_1	X_2
	$\cancel{a_0}$	$\cancel{b_0}$
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	b_2	c_2
	b_3	c_2
	$\cdots$	$\cdots$
	b_N	c_2

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

R_{12}

X_1	X_2
$\cancel{a_0}$	$\cancel{b_0}$
a_1	b_1

R_{23}

X_2	X_3
$\cancel{b_0}$	$\cancel{c_0}$
b_1	c_1
b_2	c_2
b_3	c_2
$\dots$	$\dots$
b_N	c_2

R_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\dots$	$\dots$	$\dots$
c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

R_{12}

X_1	X_2
$\cancel{a_0}$	$\cancel{b_0}$
a_1	b_1

R_{23}

X_2	X_3
$\cancel{b_0}$	$\cancel{c_0}$
b_1	c_1
$\cancel{b_2}$	$\cancel{c_2}$
$\cancel{b_3}$	$\cancel{c_2}$
$\cancel{\cdots}$	$\cancel{\cdots}$
$\cancel{b_N}$	$\cancel{c_2}$

R_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_2	d_2	e_1
c_2	d_3	e_1
$\cdots$	$\cdots$	$\cdots$
c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\begin{tikzcd} \psi_{12}(x_1, x_2) \arrow[r, dash] & \psi_{23}(x_2, x_3) \arrow[r, dash] \arrow[r, bend left] & \psi_{345}(x_3, x_4, x_5) \end{tikzcd}$$

R_{12}	X_1	X_2
	$\cancel{a_0}$	$\cancel{b_0}$
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	$\cancel{b_2}$	$\cancel{c_2}$
	$\cancel{b_3}$	$\cancel{c_2}$
	$\cancel{\cdots}$	$\cancel{\cdots}$
	$\cancel{b_N}$	$\cancel{c_2}$

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_2	d_2	e_1
	c_2	d_3	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_2	d_N	e_1

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

R_{12}	X_1	X_2
	$\cancel{a_0}$	$\cancel{b_0}$
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	$\cancel{b_2}$	$\cancel{c_2}$
	$\cancel{b_3}$	$\cancel{c_2}$
	$\cancel{\cdots}$	$\cancel{\cdots}$
	$\cancel{b_N}$	$\cancel{c_2}$

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	$\cancel{c_2}$	$\cancel{d_2}$	$\cancel{e_1}$
	$\cancel{c_2}$	$\cancel{d_3}$	$\cancel{e_1}$
	$\cancel{\cdots}$	$\cancel{\cdots}$	$\cancel{\cdots}$
	$\cancel{c_2}$	$\cancel{d_N}$	$\cancel{e_1}$

Full Reducer: Example

Consider again the previous join tree and factors:

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

R_{12}	X_1	X_2
	$\cancel{a_0}$	$\cancel{b_0}$
	a_1	b_1

R_{23}	X_2	X_3
	$\cancel{b_0}$	$\cancel{c_0}$
	b_1	c_1
	$\cancel{b_2}$	$\cancel{c_2}$
	$\cancel{b_3}$	$\cancel{c_2}$
	$\cancel{\cdots}$	$\cancel{\cdots}$
	$\cancel{b_N}$	$\cancel{c_2}$

R_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	$\cancel{c_2}$	$\cancel{d_2}$	$\cancel{e_1}$
	$\cancel{c_2}$	$\cancel{d_3}$	$\cancel{e_1}$
	$\cancel{\cdots}$	$\cancel{\cdots}$	$\cancel{\cdots}$
	$\cancel{c_2}$	$\cancel{d_N}$	$\cancel{e_1}$

Only tuples remain that contribute to at least one tuple in the join result

Yannakakis's Algorithm for Acyclic CQs: Wrapping Up

Adaptation of Step 1: initialise views as fully reduced

Yannakakis's Algorithm for Acyclic CQs: Wrapping Up

Adaptation of Step 1: initialise views as fully reduced

Size of intermediate results is bounded by product of input size and output size

Yannakakis's Algorithm for Acyclic CQs: Wrapping Up

Adaptation of Step 1: initialise views as fully reduced

Size of intermediate results is bounded by product of input size and output size

Time complexity: Linear in the product of input size and output size (after sorting)

Yannakakis's Algorithm for Acyclic CQs: Wrapping Up

Adaptation of Step 1: initialise views as fully reduced

Size of intermediate results is bounded by product of input size and output size

Time complexity: Linear in the product of input size and output size (after sorting)

For free-connex acyclic CQs, the algorithm is instance-optimal:

Time complexity is linear in the sum of input size and output size (after sorting)

1. Bottom-up *local* computation in time proportional to IN
   - Pick one node with free variables as root
   - The nodes with free variables form a connected subtree in the join tree
   - Values for free variables pushed up when these variables are also in parent

   We are now left with a reduced join tree only over free variables
2. We can now join all the remaining factors in time proportional to OUT

2. Computing Joins using LeapFrog TrieJoin

Beyond Acyclicity

Various applications call for cyclic joins

- graph problems, e.g., looking for cyclic patterns in networks, social media
- Typical cyclic queries: loops, triangles, Loomis-Whitney
- Compute the factors representing the bags in hypertree decompositions

Surprisingly, mainstream join algorithms are sub-optimal for cyclic joins

- nested-loops join, hash join, sort-merge join, [your favourite join algorithm]
- Sub-optimal: It takes asymptotically more time than worst-case join size
- We will see this by means of an example later in this lecture

We next discuss a worst-case optimal join algorithm: LeapFrog TrieJoin

- Runtime proportional to the size of the join output (proof in paper)
- Recall: Join output size $O(N^{\rho^*})$; ρ^* is the fractional edge cover number
- We later show that $O(N^{\rho^*})$ is worst-case optimal join size

LFTJ: The LeapFrog TrieJoin Algorithm

State-of-the art worst-case optimal join (WCOJ) algorithm

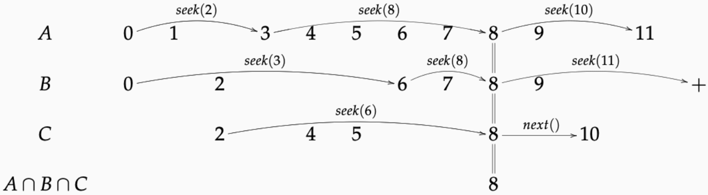
- Available at `http://arxiv.org/abs/1210.0481`
- Variants implemented in commercial and open-source query engines
- LFTJ can be orders of magnitude faster for cyclic queries on large datasets than existing commercial and open-source query engines
- Adapting an existing engine to support joins worst-case optimally requires significant design changes

Join of Unary Factors: Standard Approach

Compute the join of unary factors (intersection): $\Phi(x) = A(x) \otimes B(x) \otimes C(x)$

A	0	1		3	4	5	6	7	8	9		11
B	0		2				6	7	8	9		
C			2		4	5			8		10	
$A \cap B \cap C$									8			

- Standard approach (for sorted factors): multi-way sort-merge join
- Iterators over the three factors proceed in lockstep to find common values
- Each iterator scans the entire list
- Time to compute: proportional to the sizes of the lists



Complexity: let $N_{min} = \min\{|A|, |B|, |C|\}$ and $N_{max} = \max\{|A|, |B|, |C|\}$. Then leapfrog join runs in time $\Theta(N_{min}(1 + \log(N_{max}/N_{min})))$.

- Leapfrog Join: Multi-way sort-merge join using smart seeks instead of scans
- Seeking m keys amongst N possible keys in ascending order has amortised complexity $O(1 + \log(N/m))$ as for balanced search tree data structures

Linear Iterator

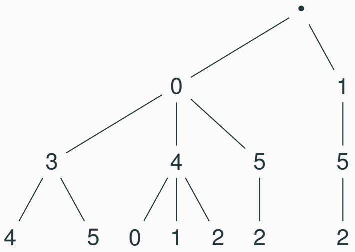
We navigate a unary factor using an iterator that sees it as an ordered list.

The linear iterator interface:

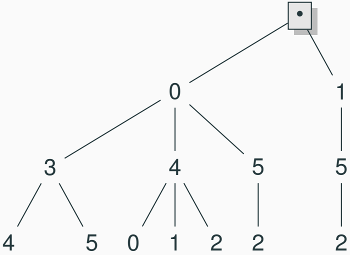
int key()	Returns the key at the current iterator position
next()	Proceeds to the next key
seek(int seekKey)	Positions the iterator at a least upper bound for seekKey
	i.e., the least key $\geq$ seekKey, or move to end if no such key exists.
	The sought key must be $\geq$ the key at the current position.
bool atEnd()	Returns true if the iterator is at the end.

Trie Presentation of Factors With Non-Unary Arity

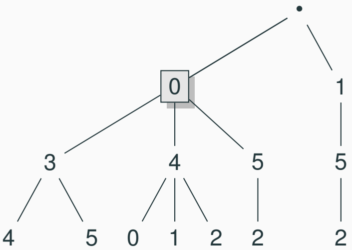
$R(x, y, z)$
(0,3,4)
(0,3,5)
(0,4,0)
(0,4,1)
(0,4,2)
(0,5,2)
(1,5,2)



Trie Iterator in Action: Example

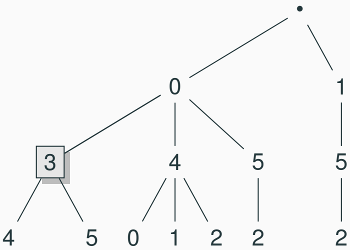


Trie Iterator in Action: Example



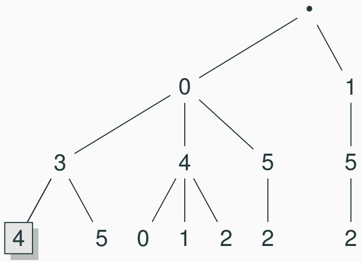
Call: open()

Trie Iterator in Action: Example



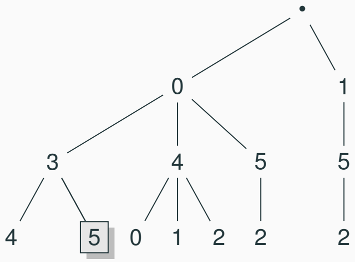
Call: open()

Trie Iterator in Action: Example



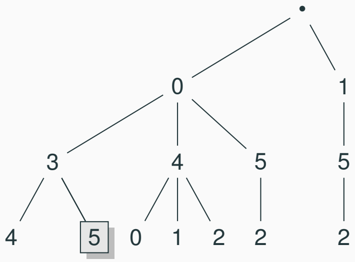
Call: open()

Trie Iterator in Action: Example



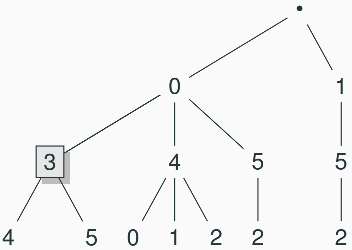
Call: next()

Trie Iterator in Action: Example



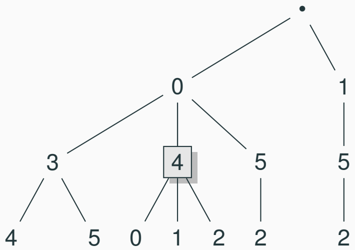
Call: atEnd() true

Trie Iterator in Action: Example



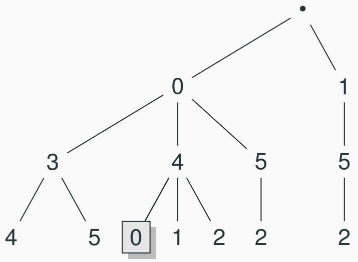
Call: up()

Trie Iterator in Action: Example



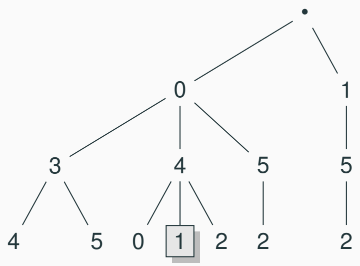
Call: next()

Trie Iterator in Action: Example



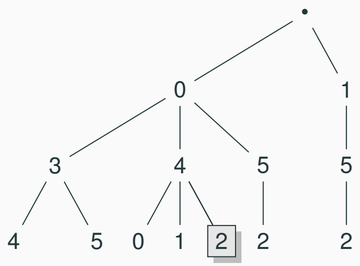
Call: open()

Trie Iterator in Action: Example



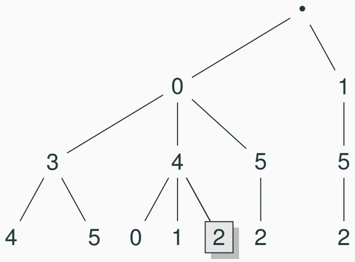
Call: next()

Trie Iterator in Action: Example



Call: next()

Trie Iterator in Action: Example



Call: atEnd() true

and so on

Trie Iterator

We navigate a factor using an iterator that sees it as a trie.

The trie iterator interface:

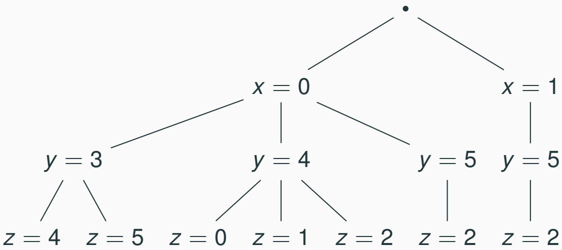
void open()	Proceed to the first key at the next depth
void up()	Return to the parent key at the previous depth

---

int key()	Returns the key at the current iterator position
next()	Proceeds to the next key
seek(int seekKey)	Positions the iterator at a least upper bound for seekKey
	i.e., the least key $\geq$ seekKey, or move to end if no such key exists.
	The sought key must be $\geq$ the key at the current position.
bool atEnd()	Returns true if the iterator is at the end.

Binding Trie: Variables Mapped to Values During Trie Traversal

- The variables of a factor are bound to values following a backtracking search
- Satisfying assignments are emitted when leaves are reached
- Consider our trie below for factor $R(x, y, z)$

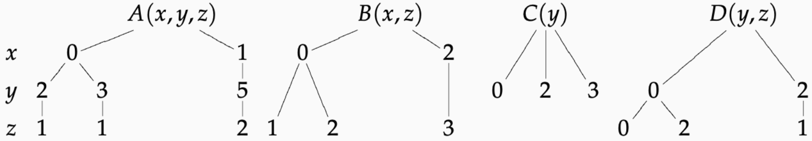


Execution Strategy of LeapFrog TrieJoin

- Choose a global and total variable ordering for all factors
- Each factor is traversed following this variable ordering
- Backtracking search through binding trie to find result tuples
  - Example join: $R(a, b) * S(b, c) * T(a, c)$ under variable ordering: $[a, b, c]$
  - Leapfrog join for a occurring in both R and T
  - For each such a, leapfrog join for b occurring in S and R^a
  - For each such b, leapfrog join for c occurring in S^b and T^a
- The result is presented as a non-materialised view using a trie iterator

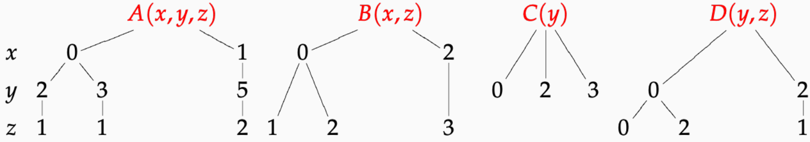
LeapFrog TrieJoin in Action: Example (1/7)

Tree join example



LeapFrog TrieJoin in Action: Example (2/7)

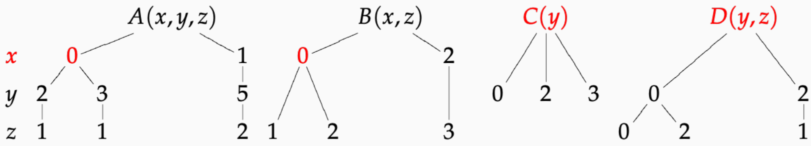
Tree join example



Position iterators at root of trees.

LeapFrog TrieJoin in Action: Example (3/7)

Tree join example

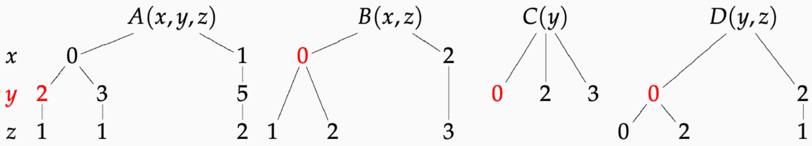


open() iterators for trees that bind x.

Join for $A(x, \_, \_), B(x, \_)$ finds $x = 0$.

LeapFrog TrieJoin in Action: Example (4/7)

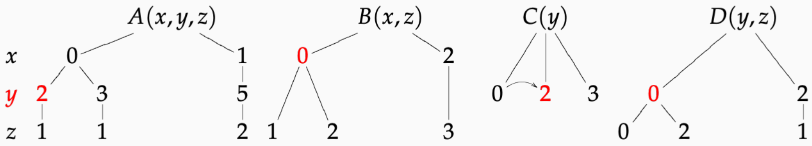
Tree join example



open() iterators for trees that bind y.

LeapFrog TrieJoin in Action: Example (5/7)

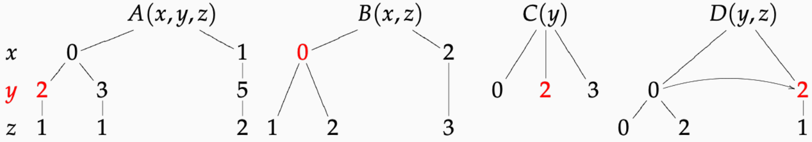
Tree join example



seek(2) on C iterator

LeapFrog TrieJoin in Action: Example (6/7)

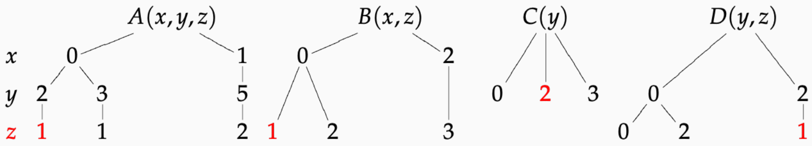
Tree join example



seek(2) on D iterator; join for $A(0,y,\_), C(y), D(y,\_)$ finds $y = 2$

LeapFrog TrieJoin in Action: Example (7/7)

Tree join example



open() on iterators for z

join for $A(0,2,z), B(0,z), D(2,z)$ produces $z = 1$: emit $(0,2,1)$

3. Suboptimality of Mainstream Join Algorithms

The Triangle Join on Factors with Heavy and Light Values

$$\Phi(x_1, x_2, x_3) = \psi_{12}(x_1, x_2) \otimes \psi_{13}(x_1, x_3) \otimes \psi_{23}(x_2, x_3)$$

$\psi_{12}(x_1, x_2)$	
a_0	b_0
a_0	$\dots$
a_0	b_m
a_1	b_0
$\dots$	b_0
a_m	b_0

$\psi_{13}(x_1, x_3)$	
a_0	c_0
a_0	$\dots$
a_0	c_m
a_1	c_0
$\dots$	c_0
a_m	c_0

$\psi_{23}(x_2, x_3)$	
b_0	c_0
b_0	$\dots$
b_0	c_m
b_1	c_0
$\dots$	c_0
b_m	c_0

$\Phi(x_1, x_2, x_3)$		
a_0	b_0	c_0
a_0	b_0	$\dots$
a_0	b_0	c_m
a_0	b_1	c_0
a_0	$\dots$	c_0
a_0	b_m	c_0
a_1	b_0	c_0
$\dots$	b_0	c_0
a_m	b_0	c_0

Each input factor has size $2m + 1$, the output factor has size $3m + 1$.

Values a_0, b_0, c_0 are *heavy* in the input factors, all other values are *light*

Ideally, a join algorithm takes time proportional to the input and output sizes.

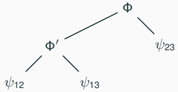
How would existing join algorithms compute this query?

Mainstream Join Algorithms Compute One Join at a Time

Traditional join: Join two of the three factors, then join in the remaining factor

$$\Phi'(x_1, x_2, x_3) = \psi_{12}(x_1, x_2) \otimes \psi_{13}(x_1, x_3)$$

$$\Phi(x_1, x_2, x_3) = \Phi'(x_1, x_2, x_3) \otimes \psi_{23}(x_2, x_3)$$



- Φ' takes quadratic time to compute: It has $(m+1)^2 + m$ tuples

$\psi_{12}(x_1, x_2)$

a_0	b_0
a_0	$\dots$
a_0	b_m
a_1	b_0
$\dots$	b_0
a_m	b_0

$\psi_{13}(x_1, x_3)$

a_0	c_0
a_0	$\dots$
a_0	c_m
a_1	c_0
$\dots$	c_0
a_m	c_0

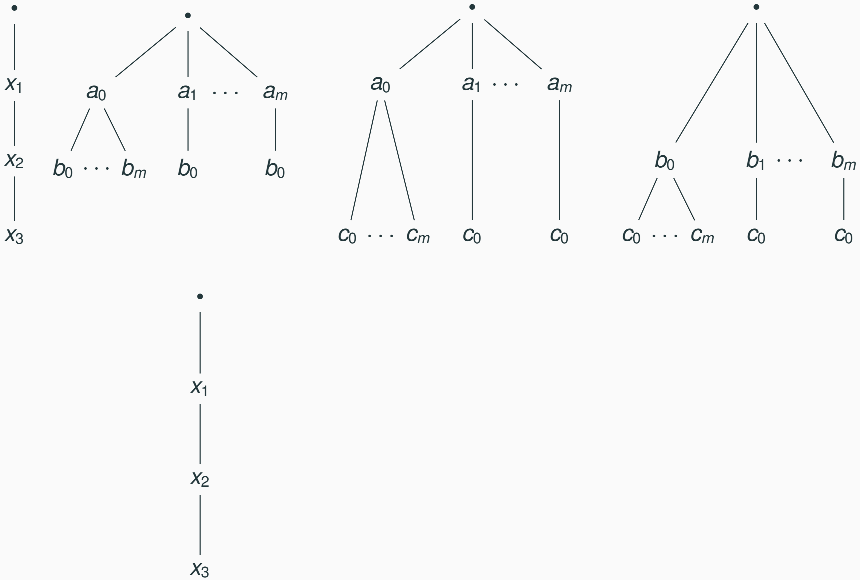
$\Phi'(x_1, x_2, x_3)$

a_0	b_0	c_0
a_0	b_0	$\dots$
a_0	b_0	c_m
$\dots$	$\dots$	$\dots$
a_0	b_m	c_0
a_0	$\dots$	c_0
a_0	b_m	c_0
a_1	b_0	c_0
$\dots$	b_0	c_0
a_m	b_0	c_0

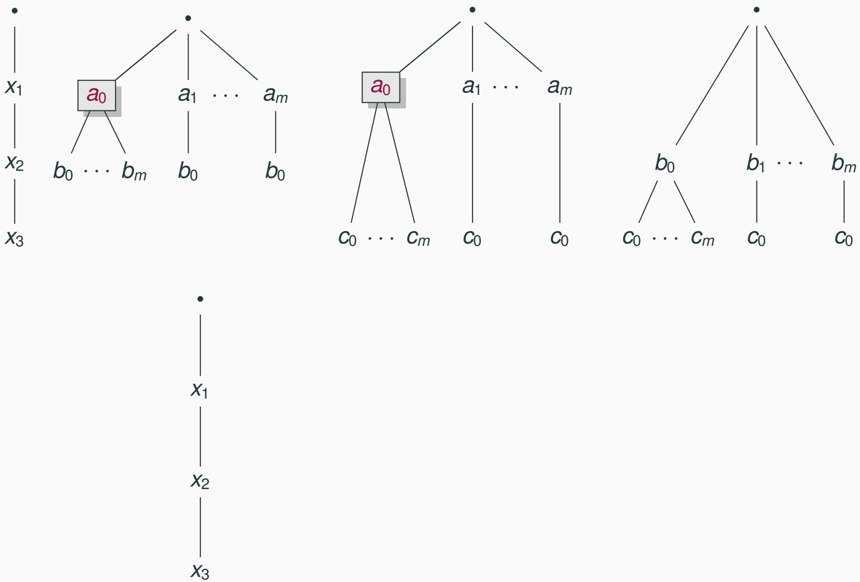
- This behaviour happens regardless of which two factors we join first

This is not optimal: The intermediate result is larger than the final join result

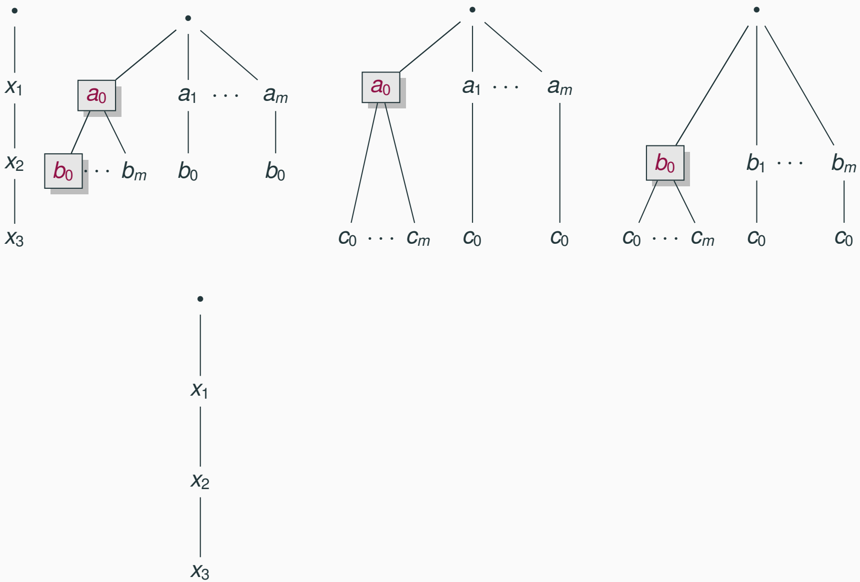
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



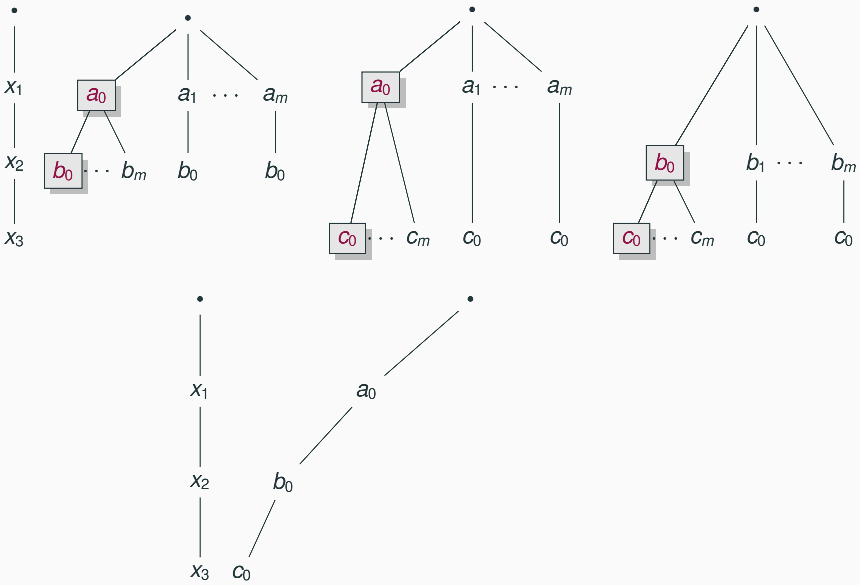
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



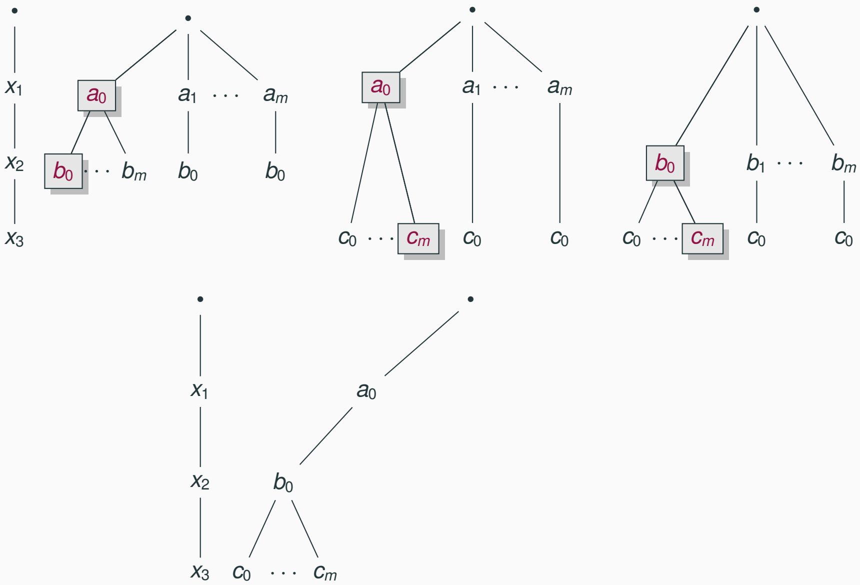
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



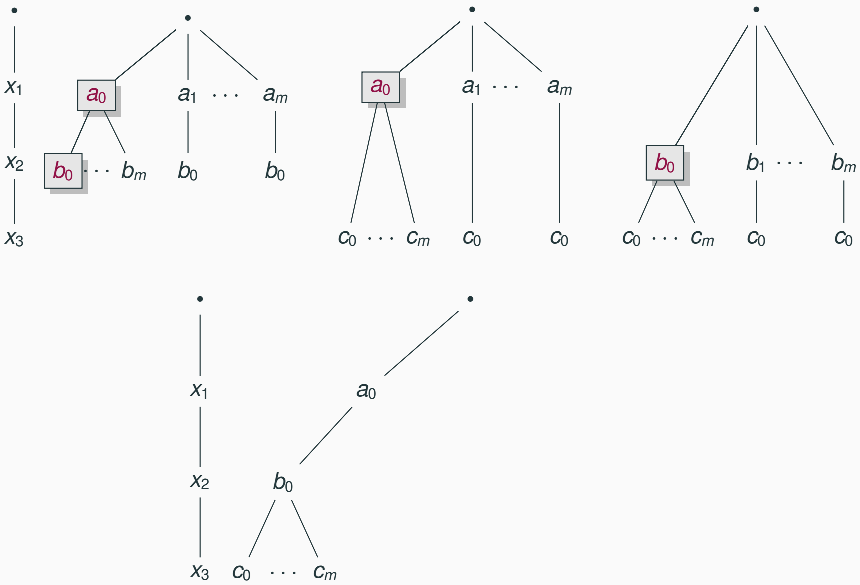
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



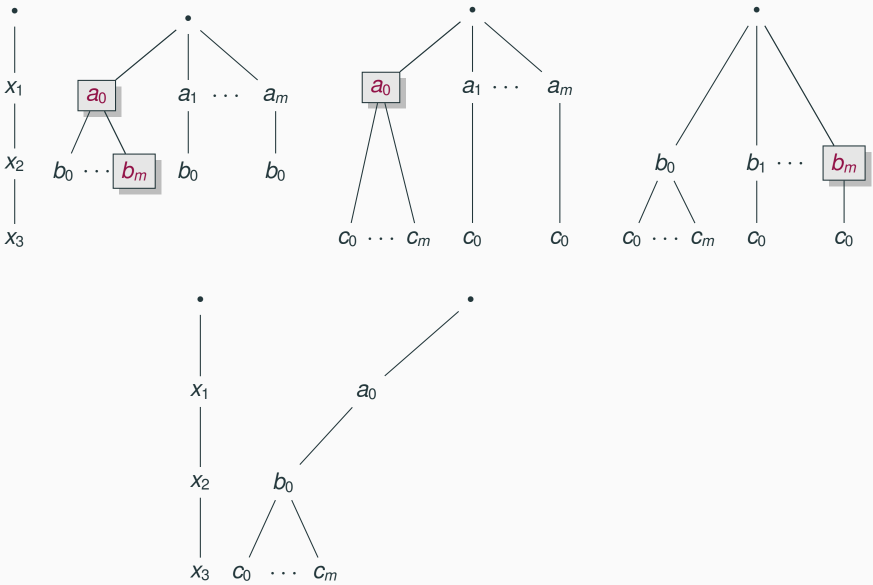
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



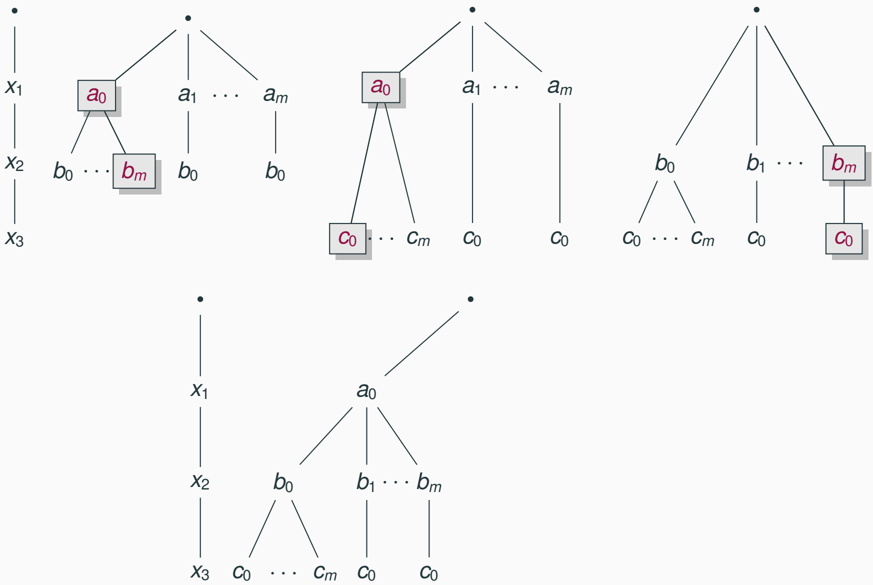
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



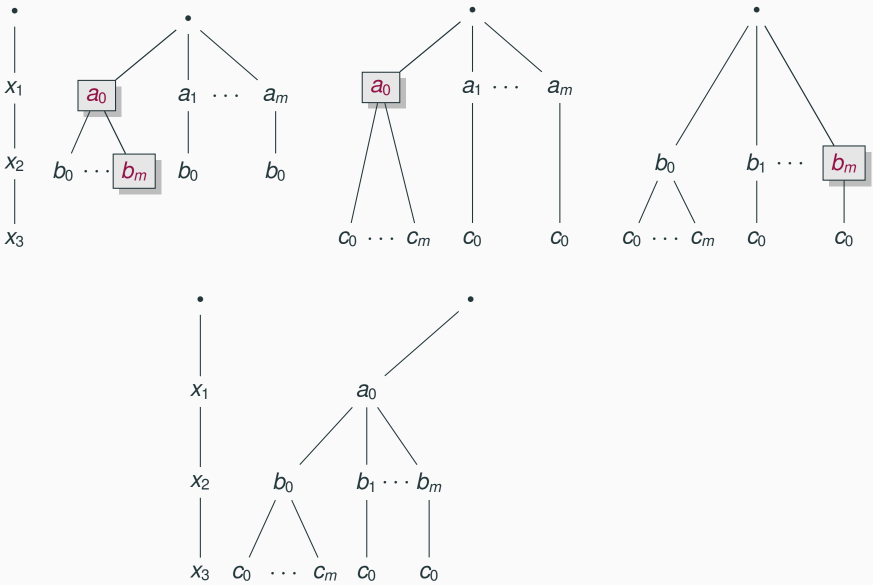
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



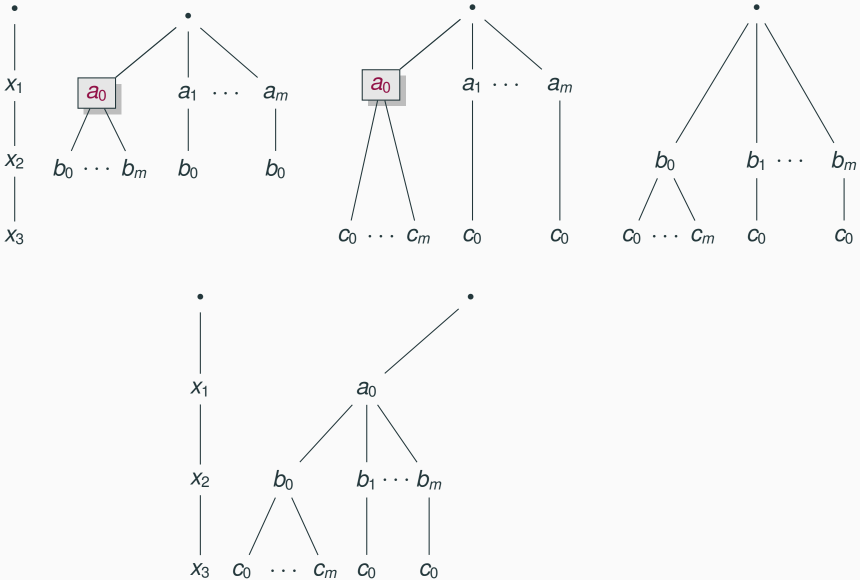
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



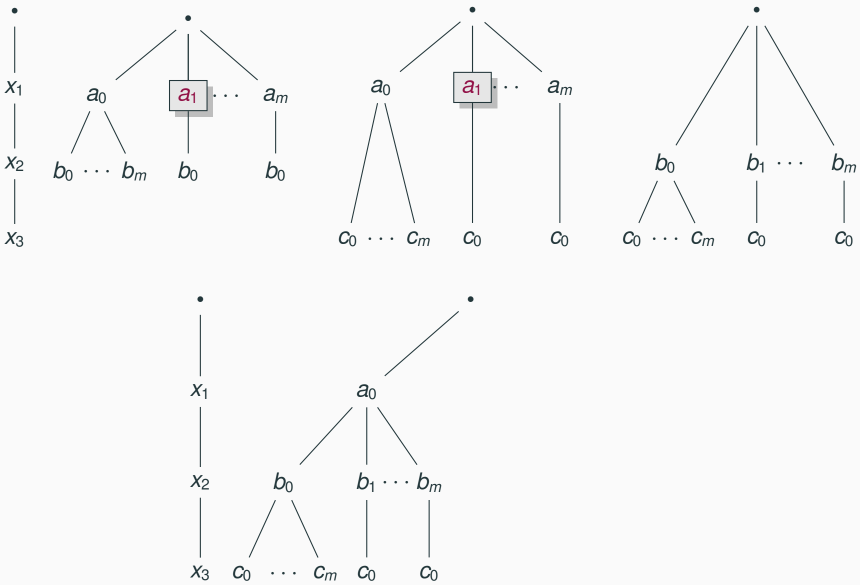
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



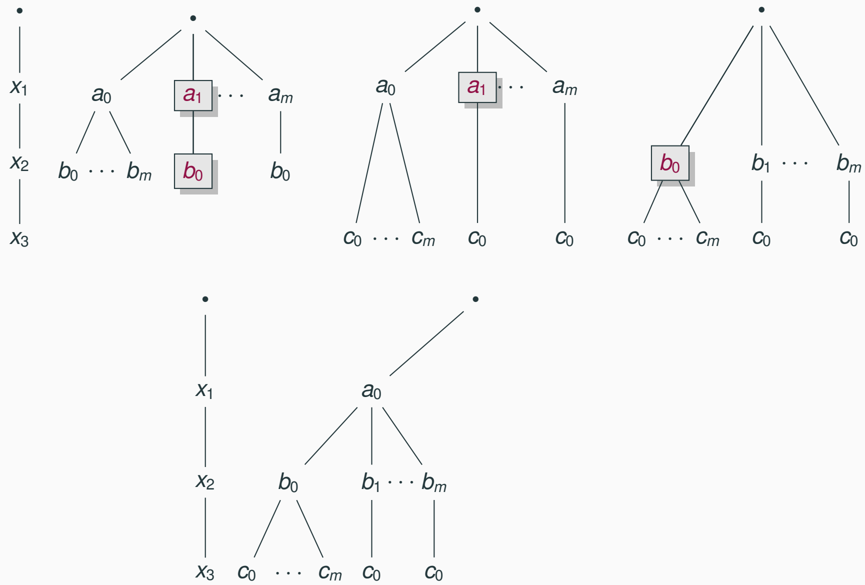
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



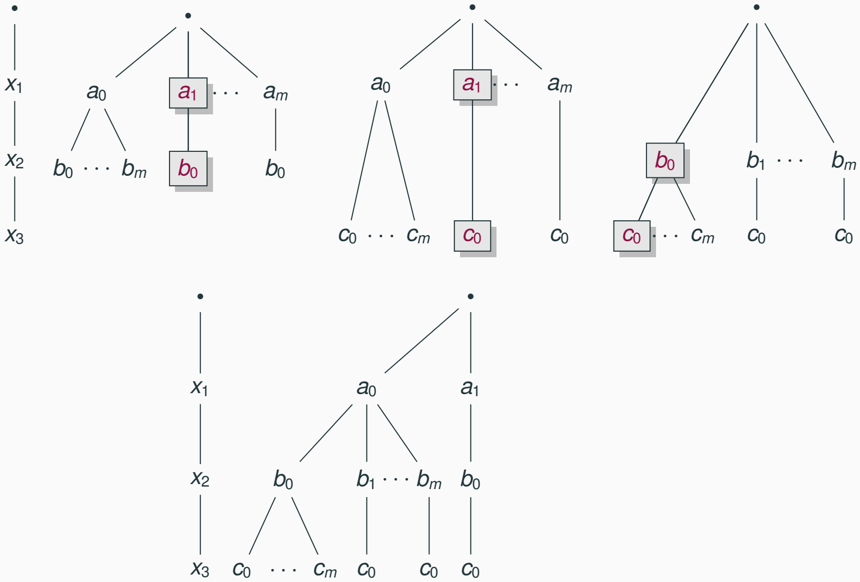
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



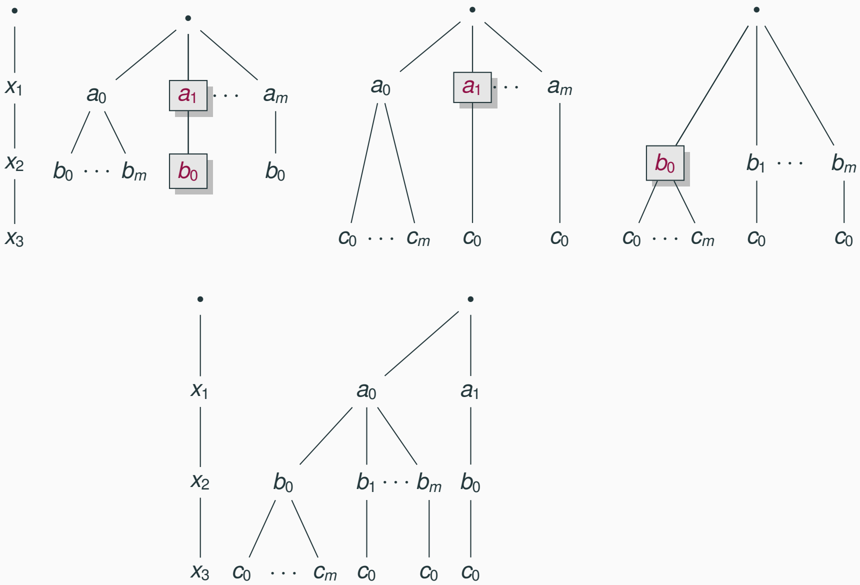
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



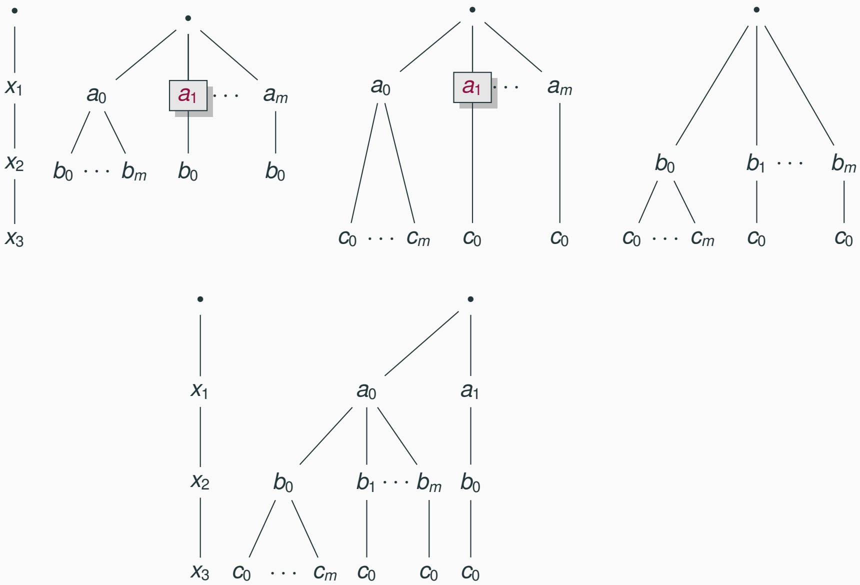
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



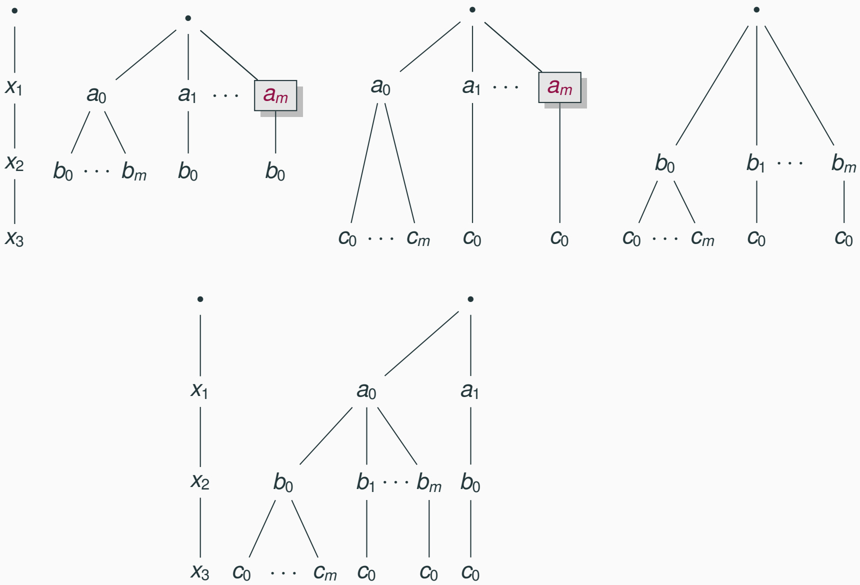
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



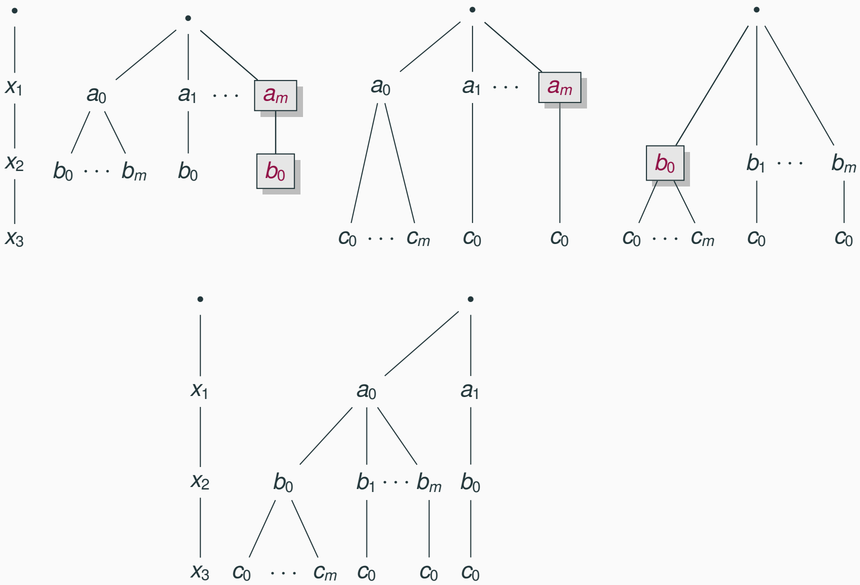
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



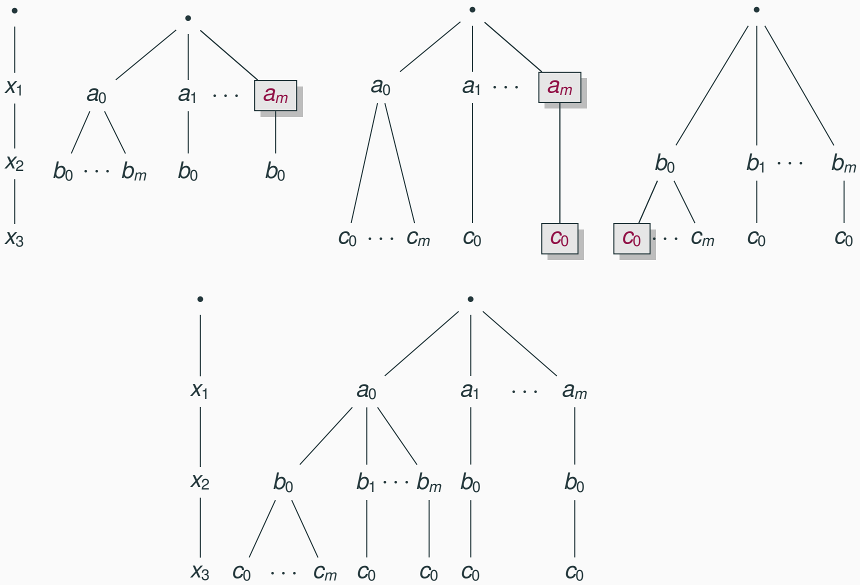
LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



LeapFrog TrieJoin Computes All Joins Together in IN + OUT Time



4. Conjunctive Queries with Large Output

Simple Queries May Have Large Output

Output size is not a good measure for the computational effort of a query

Cartesian product $\Phi(x_1, \ldots, x_m) = \psi_1(x_1) \otimes \cdots \otimes \psi_m(x_m)$ has output size N^m

Simple Queries May Have Large Output

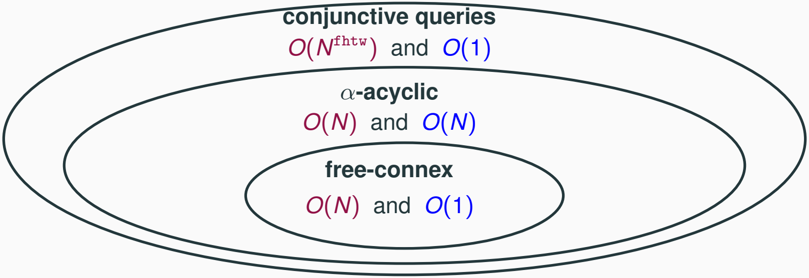
Output size is not a good measure for the computational effort of a query

Cartesian product $\Phi(x_1, \ldots, x_m) = \psi_1(x_1) \otimes \cdots \otimes \psi_m(x_m)$ has output size N^m

Decompose the computational effort into two steps:

1. Preprocessing step
   - Construct a compact data structure for all tuples in the query result
   - Construction time = true measure of the query's computational effort
2. Enumeration step
   - Enumerate the distinct tuples from this data structure one after the other
   - Delay: The time needed to return one tuple after returning the previous one
   - Constant delay is as good as enumerating tuples from a listing representation
   - One can enumerate top-k tuples in a desired order

Overview of Approaches Covered in Lecture



The above diagram shows preprocessing time and enumeration delay

Two broad strategies:

1. All computational effort in the preprocessing step to achieve constant delay
   - Still lower than materialising the entire query result: $\texttt{fhtw} \leq \rho^*$
2. Distribute the computational effort between preprocessing and enumeration

Strategy 1: All Computational Effort in Preprocessing

Input: FAQ Φ with hypergraph $\mathcal{H}$ and free variables $[f]$, input factors of size N

Preprocessing Step

- Construct a hypertree decomposition for $\mathcal{H}$ in $O(N^{\texttt{fhtw}(\mathcal{H},[f])})$
  - Hypertree becomes join tree: each bag is materialised as one factor
  - Free variables form a connected subtree including wlog the root of the join tree
- Calibrate the factors using a full reducer to remove dangling tuples
- Marginalise out bound (i.e., not free) variables
- Sort factors following an order of free variables compatible with top-down traversal of join tree

Output of preprocessing step:

- Reduced join tree whose nodes are factors over free variables only
- Expensive and unnecessary: Joining all factors in the reduced join tree
- $\Rightarrow$ Time to compute Φ is $O(N^{\rho^*(\mathcal{H})})$, yet $\rho^*(\mathcal{H}) \geq \texttt{fhtw}(\mathcal{H}, [f])$

Strategy 1: Preprocessing Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\cdots$	$\cdots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\cdots$	$\cdots$
b_1	c_N
b_2	c_1
$\cdots$	$\cdots$
b_2	c_N
b_3	c_1

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_1	d_2	e_1
c_2	d_1	e_1
c_2	d_2	e_1
$\cdots$	$\cdots$	$\cdots$
c_N	d_1	e_1
c_N	d_2	e_1

Strategy 1: Preprocessing Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;\text{---}\; \psi_{23}(x_2, x_3) \;\text{---}\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\cdots$	$\cdots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\cdots$	$\cdots$
b_1	c_N
b_2	c_1
$\cdots$	$\cdots$
b_2	c_N
b_3	c_1

ψ_{345}

X_3	X_4	X_5
c_1	d_1	e_1
c_1	d_2	e_1
c_2	d_1	e_1
c_2	d_2	e_1
$\cdots$	$\cdots$	$\cdots$
c_N	d_1	e_1
c_N	d_2	e_1

- Factors are already calibrated

Strategy 1: Preprocessing Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\cdots$	$\cdots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\cdots$	$\cdots$
b_1	c_N
b_2	c_1
$\cdots$	$\cdots$
b_2	c_N
b_3	c_1

$\psi_{345\to 23}$

X_3
c_1
c_2
$\cdots$
c_N

- Factors are already calibrated
- Variables X_4, X_5 are marginalised out

Strategy 1: Preprocessing Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\cdots$	$\cdots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\cdots$	$\cdots$
b_1	c_N
b_2	c_1
$\cdots$	$\cdots$
b_2	c_N
b_3	c_1

$\psi_{345\to 23}$

X_3
c_1
c_2
$\cdots$
c_N

- Factors are already calibrated
- Variables X_4, X_5 are marginalised out
- Factors are already sorted appropriately

Strategy 1: All Computational Effort in Preprocessing

Constant-Delay Enumeration of Tuples from Reduced Join Tree

- Variable order $(X_1, \ldots, X_f)$ compatible with top-down traversal of join tree
- Factors are sorted following this variable order
- For each value x_1 for X_1, we seek a value x_2 for X_2, and so on
  - If factors sorted, then all values for X_i are in a contiguous block in factors, given the values for $X_1, \ldots, X_{i-1}$
  - Since there are no dangling tuples, each value x_i participates in at least one output tuple
- We output a complete assignment $\mathbf{x}_{[f]}$ and backtrack

Why constant delay?

- For each variable X_i, we need constant time to locate its next value, given values for variables $X_1, \ldots, X_{i-1}$
- The time to output a tuple is independent of the sizes of the factors

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\cdots$	$\cdots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\cdots$	$\cdots$
b_1	c_N
b_2	c_1
$\cdots$	$\cdots$
b_2	c_N
b_3	c_1

$\psi_{345\to 23}$

X_3
c_1
c_2
$\cdots$
c_N

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_1
	$\cdots$	$\cdots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\cdots$
	c_N

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_1
	$\cdots$	$\cdots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\cdots$
	c_N

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_1
	$\cdots$	$\cdots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\cdots$
	c_N

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\dots$
	c_N

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345\to 23}(x_3)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\dots$	$\dots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\dots$	$\dots$
b_1	c_N
b_2	c_1
$\dots$	$\dots$
b_2	c_N
b_3	c_1

$\psi_{345\to 23}$

X_3
c_1
c_2
$\dots$
c_N

Output: (a_1, b_1, c_1)

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}

X_1	X_2
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
$\dots$	$\dots$
a_N	b_1
a_N	b_2
a_N	b_3

ψ_{23}

X_2	X_3
b_1	c_1
$\dots$	$\dots$
b_1	c_N
b_2	c_1
$\dots$	$\dots$
b_2	c_N
b_3	c_1

$\psi_{345\to 23}$

X_3
c_1
c_2
$\dots$
c_N

Output: $(a_1, b_1, c_1), \dots, (a_1, b_1, c_N)$

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\dots$
	c_N

Output: $(a_1, b_1, c_1), \dots, (a_1, b_1, c_N)$

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) — \psi_{23}(x_2, x_3) — \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\dots$
	c_N

Output: $(a_1, b_1, c_1), \dots, (a_1, b_1, c_N), (a_1, b_2, c_1)$

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\dots$
	c_N

Output: $(a_1, b_1, c_1), \dots, (a_1, b_1, c_N), (a_1, b_2, c_1), \dots, (a_1, b_2, c_N)$

Strategy 1: Enumeration Example

Input: join tree and factors as follows, free variables are $\{X_1, X_2, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\; \psi_{23}(x_2, x_3) \;—\; \psi_{345\to 23}(x_3)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

$\psi_{345\to 23}$	X_3
	c_1
	c_2
	$\dots$
	c_N

Output: $(a_1, b_1, c_1), \dots, (a_1, b_1, c_N), (a_1, b_2, c_1), \dots, (a_1, b_2, c_N), \dots$

Strategy 2: Linear Preprocessing and Linear Enumeration Delay

We discuss this strategy for α-acyclic CQs that are not free-connex

Preprocessing Step

- Apply a full reducer to remove the dangling tuples
- Sort factors following an order of the free variables compatible with top-down traversal of join tree
- This computation can be done in linear(ithmic) time

Enumeration Step

- Iterate over the possible values x_1 for variable X_1
- Restrict the factors for X_1 to those tuples where $X_1 = x_1$
- Fully reduce all other factors to avoid newly dangling tuples
- Do the previous three steps for the next variable
- When a complete variable assignment is found, output it and backtrack
- This computation can be done in linear time per complete assignment

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;\text{———}\; \psi_{23}(x_2, x_3) \;\text{———}\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_1
	$\dots$	$\dots$
	b_2	c_N
	b_3	c_1

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\dots$	$\dots$	$\dots$
	c_N	d_1	e_1
	c_N	d_2	e_1

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \text{ ——— } \psi_{23}(x_2, x_3) \text{ ——— } \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \longrightarrow \psi_{23}(x_2, x_3) \longrightarrow \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;\text{———}\; \psi_{23}(x_2, x_3) \;\text{———}\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here a_1

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;\text{———}\; \psi_{23}(x_2, x_3) \;\text{———}\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\dots$	$\dots$	$\dots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here a_1
  - remove dangling tuples in other factors

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\!—\!—\; \psi_{23}(x_2, x_3) \;—\!—\!—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	$\boxed{a_1}$	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here $\boxed{a_1}$
  - remove dangling tuples in other factors
  - iterate over all results of $\Phi_3(x_3) = \psi_{12}(\boxed{a_1}, x_2) \otimes \psi_{23}(x_2, x_3) \otimes \psi_{345}(x_3, x_4, x_5)$

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\dots$	$\dots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\dots$	$\dots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\dots$	$\dots$	$\dots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here a_1
  - remove dangling tuples in other factors
  - iterate over all results of $\Phi_3(x_3) = \psi_{12}(a_1, x_2) \otimes \psi_{23}(x_2, x_3) \otimes \psi_{345}(x_3, x_4, x_5)$

  Output: (a_1, c_1)

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;—\!—\!—\; \psi_{23}(x_2, x_3) \;—\!—\!—\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here a_1
  - remove dangling tuples in other factors
  - iterate over all results of $\Phi_3(x_3) = \psi_{12}(a_1, x_2) \otimes \psi_{23}(x_2, x_3) \otimes \psi_{345}(x_3, x_4, x_5)$

  Output: $(a_1, c_1), (a_1, c_2)$

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \;\text{———}\; \psi_{23}(x_2, x_3) \;\text{———}\; \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	$\boxed{a_1}$	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	$\cdots$	$\cdots$
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	$\cdots$	$\cdots$
	b_1	$\boxed{c_N}$
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	$\cdots$	$\cdots$	$\cdots$
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here $\boxed{a_1}$
  - remove dangling tuples in other factors
  - iterate over all results of $\Phi_3(x_3) = \psi_{12}(\boxed{a_1}, x_2) \otimes \psi_{23}(x_2, x_3) \otimes \psi_{345}(x_3, x_4, x_5)$

  Output: $(a_1, c_1), (a_1, c_2), \ldots, (a_1, c_N)$

Strategy 2: Enumeration Example

Fix again join tree and factors as follows, free variables are now $\{X_1, X_3\}$

$$\psi_{12}(x_1, x_2) \text{ --- } \psi_{23}(x_2, x_3) \text{ --- } \psi_{345}(x_3, x_4, x_5)$$

ψ_{12}	X_1	X_2
	a_1	b_1
	a_1	b_2
	a_2	b_1
	a_2	b_2
	...	...
	a_N	b_1
	a_N	b_2
	a_N	b_3

ψ_{23}	X_2	X_3
	b_1	c_1
	b_2	c_1
	b_3	c_1
	b_1	c_2
	b_2	c_2
	...	...
	b_1	c_N
	b_2	c_N

ψ_{345}	X_3	X_4	X_5
	c_1	d_1	e_1
	c_1	d_2	e_1
	c_2	d_1	e_1
	c_2	d_2	e_1
	...	...	...
	c_N	d_1	e_1
	c_N	d_2	e_1

- Preprocessing: Remove dangling tuples, sort ψ_{23} by the free variable X_3
- Iterate over all results of $\Phi_1(x_1) = \bigoplus_{x_2} \psi_{12}(x_1, x_2)$
  - restrict ψ_{12} to fixed value, here a_1
  - remove dangling tuples in other factors
  - iterate over all results of $\Phi_3(x_3) = \psi_{12}(a_1, x_2) \otimes \psi_{23}(x_2, x_3) \otimes \psi_{345}(x_3, x_4, x_5)$

  Output: $(a_1, c_1), (a_1, c_2), \ldots, (a_1, c_N), \ldots$