The Rising Lake

An Intuitive Introduction to Functional Aggregate Queries

Wenchao Bai

wbai@seu.edu.cn

December 11, 2025

About This Title

The unknown thing to be known appeared to me as some stretch of earth or hard marl, resisting penetration ... the sea[1] advances insensibly in silence, nothing seems to happen, nothing moves, the water is so far off you hardly hear it ... yet it finally surrounds the resistant substance.

— *Alexander Grothendieck*, *Récoltes et semailles* (1985–1987)

[1]We use "The Rising Lake" instead of "The Rising Sea" as the title since FAQ spans a narrower domain than algebraic geometry.

About This Tutorial

- **Common properties** behind optimization techniques.
- **Common structure** shared by many computational problems.
- **Unified language** able to express semiring problems.

- An intuition for **efficient algorithms** to compute any expression in this language.

Resources

- **Course materials**: Efficient Algorithms[2] (by Prof. Dan Olteanu, UZH)
- **Original paper**: FAQ: Questions Asked Frequently [1] (best paper, PODS'16)

[2] `https://www.ifi.uzh.ch/en/dast/teaching/EA.html`

Outline

Example: Matrix Multiplication

- **Question**: Compute $A \times B \times C$
- **Technique**: Different parenthesizations have different cost:

$$(AB)C \quad \text{vs.} \quad A(BC)$$

- **Insight:** Using **associativity** to choose grouping reduces computation cost.

Example: Summing via MapReduce

- **Question**: Sum over a large dataset distributed across nodes
- **Technique**: Compute partial sums parallelly at different nodes, then aggregate:

$$\sum_{i=1}^{N} x_i = \sum (\text{partial sums})$$

- **Insight:** Using **commutativity** allows arbitrary order of aggregation.

Example: Query Optimization

- **Question**: Compute the following query:

$$\sigma_{x \geq 10}(A \bowtie_x B)$$

- **Technique**: Predicate pushdown:

$$(\sigma_{x \geq 10}(A)) \bowtie_x (\sigma_{x \geq 10}(B))$$

- **Insight**: Using **distributivity** to reduce joining cost.

Takeaway: Common Properties

Associativity, **commutativity**, & **distributivity** are infrastructures for optimization.

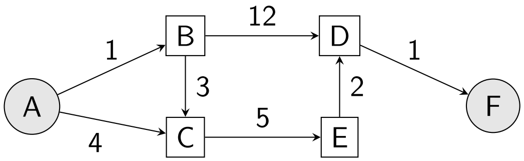
Outline

Key Observation

Computational problems commonly use

- sequences of two binary operations
- applied on a finite set of values from a given domain.

Example: Shortest Distance (SD)



$$W = \begin{pmatrix} 0 & 1 & 4 & \infty & \infty & \infty \\ \infty & 0 & 3 & 12 & \infty & \infty \\ \infty & \infty & 0 & \infty & 5 & \infty \\ \infty & \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\mathtt{SD}(A,F) = \min \left\{ \begin{array}{c} W_{A,x_1} + W_{x_1,F} \\ \dots \\ W_{A,x_1} + \dots + W_{x_{n-1},x_n} + W_{x_n,F} \end{array} \right\} = \min \left\{ \begin{array}{c} 1+12+1 \\ 4+5+2+1 \\ 1+3+5+2+1 \end{array} \right\}.$$

- **Binary operations**: min, $+$
- **Domain**: $(-\infty, \infty]$

Example: Conjunctive Query (CQ)

Orders (O for short)

customer	day	dish
Elise	Monday	burger
Elise	Friday	burger
Steve	Friday	hotdog
Joe	Friday	hotdog

Dish (D for short)

dish	item
burger	patty
burger	onion
burger	bun
hotdog	sausage

Items (I for short)

item	price
patty	6
onion	2
bun	2
sausage	4

$$\mathtt{CQ}(O \bowtie D \bowtie I) = \bigcup_{(v_1,v_2,v_3,v_4,v_5)} O(v_1, v_2, v_3) \cap D(v_3, v_4) \cap I(v_4, v_5)$$

- **Binary operations**: $\cup$, $\cap$
- **Domain**: set of tuples

Common Structure Shared by These Problems

Binary operators $\oplus$ and $\otimes$ over set $\mathbf{D}$ form a commutative semiring $(\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$:

- $\oplus$ is associative: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- $\oplus$ is commutative: $a \oplus b = b \oplus a$
- $\mathbf{0}$ is the additive identity: $a \oplus \mathbf{0} = a$
- $\otimes$ is associative: $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
- $\otimes$ is commutative: $a \otimes b = b \otimes a$
- $\mathbf{1}$ is the multiplicative identity: $a \otimes \mathbf{1} = a$
- $\otimes$ distributes over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
- $\mathbf{0}$ is the multiplicative annihilator: $a \otimes \mathbf{0} = \mathbf{0}$

Additional condition for ring:

- each element a has an additive inverse $-a$: $a \oplus (-a) = \mathbf{0}$

Shortest Distance (SD) as Semiring

SD forms a min-sum semiring: $((-\infty,\infty], \min, +, \infty, 0)$:

- $\oplus = \min$ is associative: $\min(a, \min(b,c)) = \min(\min(a,b), c)$
- $\oplus = \min$ is commutative: $\min(a,b) = \min(b,a)$
- $\mathbf{0} = \infty$ is the additive identity: $\min(a,\infty) = a$
- $\otimes = +$ is associative: $a + (b+c) = (a+b) + c$
- $\otimes = +$ is commutative: $a + b = b + a$
- $\mathbf{1} = 0$ is the multiplicative identity: $a + 0 = a$
- $\otimes = +$ distributes over $\oplus = \min$: $a + \min(b,c) = \min(a+b, a+c)$
- $\mathbf{0} = \infty$ is the multiplicative annihilator: $a + \infty = \infty$

SD cannot form a ring since,

- additive inverse does not exist: $\forall a \neq \infty, \nexists x \in (-\infty,\infty]$, such that $\min(a,x) = \infty$

Conjunctive Query (CQ) as Semiring

`CQ` forms a union-intersection semiring: $(2^{\mathcal{U}}, \cup, \cap, \emptyset, \mathcal{U})$:

- $\mathcal{U}$ is the cartesian product over all attributes' domains.
- $\oplus = \cup$ is associative: $A \cup (B \cup C) = (A \cup B) \cup C$
- $\oplus = \cup$ is commutative: $A \cup B = B \cup A$
- $\mathbf{0} = \emptyset$ is the additive identity: $A \cup \emptyset = A$
- $\otimes = \cap$ is associative: $A \cap (B \cap C) = (A \cap B) \cap C$
- $\otimes = \cap$ is commutative: $A \cap B = B \cap A$
- $\mathbf{1} = \mathcal{U}$ is the multiplicative identity: $A \cap \mathcal{U} = A$
- $\otimes = \cap$ distributes over $\oplus = \cup$: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $\mathbf{0} = \emptyset$ is the multiplicative annihilator: $A \cap \emptyset = \emptyset$

`CQ` cannot form a ring since,

- additive inverse does not exist: $\forall A \neq \emptyset, \nexists X \in 2^{\mathcal{U}}$, such that $A \cup X = \emptyset$

Sample Problems and Their Semirings[3]

Category	**Problem**	**Type**	**Domain**	$\oplus$	$\otimes$	**0**	**1**
Path Queries	Shortest Distance	Min–Sum	$(-\infty, \infty]$	min	$+$	∞	0
	Connectivity	Boolean	$\{\mathtt{F}, \mathtt{T}\}$	$\vee$	$\wedge$	$\mathtt{F}$	$\mathtt{T}$
	Largest Capacity	Max–Min	$[-\infty, \infty]$	max	min	$-\infty$	∞
	Maximum Reliability	Max–Product	$[0, 1]$	max	$\times$	0	1
Satisfiability	Map Coloring	Boolean	$\{\mathtt{F}, \mathtt{T}\}$	$\vee$	$\wedge$	$\mathtt{F}$	$\mathtt{T}$
Database Queries	Conjunctive Queries	Union–Intersection	$2^{\mathcal{U}}$	$\cup$	$\cap$	$\emptyset$	$\mathcal{U}$
	Factorised Agg-Joins	Sum–Product	$\mathbb{Z}$	$+$	$\times$	0	1
...	...	...	...	...	...	...	...

[3]See topic 2 (Commutative Semirings) for more detail: `https://www.ifi.uzh.ch/en/dast/teaching/EA.html`

Takeaway: The Power of Semirings

Why are Semirings Relevant in Computer Science?

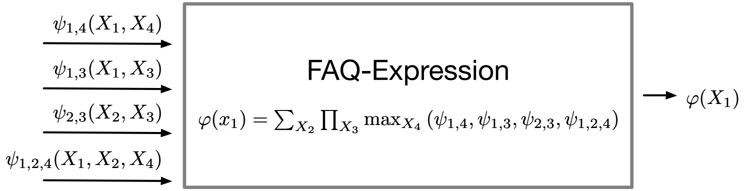
- They enable generic problem solving
  - by changing the semiring, the algorithm remains the same
- They reduce computational complexity
  - thanks to the **distributivity** law
- Permutability is an important property behind optimization techniques.
  - thanks to the **associativity** and **commutativity** laws

Different semirings give different semantics of

- the same problem
- the same algorithm
- the same complexity
- the same implementation

Outline

Functional Aggregate Query: The Input (1/2)

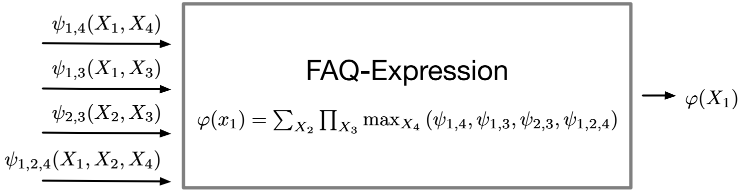


$\psi_{1,4}(X_1, X_4)$
$\psi_{1,3}(X_1, X_3)$
$\psi_{2,3}(X_2, X_3)$
$\psi_{1,2,4}(X_1, X_2, X_4)$
FAQ-Expression
$\varphi(x_1) = \sum_{X_2} \prod_{X_3} \max_{X_4} (\psi_{1,4}, \psi_{1,3}, \psi_{2,3}, \psi_{1,2,4})$
$\varphi(X_1)$

- Variables: $\mathcal{V} = \{X_1, \ldots, X_n\}$
  - $F \subseteq \mathcal{V}$: free variables (input variables)[4], *e.g.*, X_1 is a free variable of $\varphi(X_1)$.
  - $\mathcal{V} \setminus F$: bound variables, *e.g.*, $\{X_2, X_3, X_4\}$ are bound variables of $\varphi(X_1)$.
  - *E.g.*, in the query $\texttt{SD}(A, B)$ "the shortest dist. between A and B", $F = \{A, B\}$.

[4] *w.l.o.g.*, $F = \mathbf{X}_{[f]} = \{X_1, \ldots, X_f\}$, *i.e.*, the first f variables.

Functional Aggregate Query: The Input (2/2)

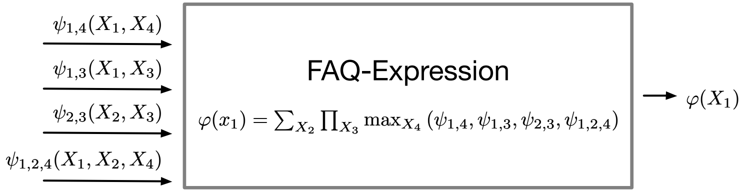


$$\varphi(x_1) = \sum_{X_2} \prod_{X_3} \max_{X_4} (\psi_{1,4}, \psi_{1,3}, \psi_{2,3}, \psi_{1,2,4})$$

- Variables: $\mathcal{V} = \{X_1, \ldots, X_n\}$
- Multi-Hypergraph: $\mathcal{H} = (\mathcal{V}, \mathcal{E})$
  - $\mathcal{V}$: set of vertices (variables)
  - $\mathcal{E} \subseteq 2^{[n]}$: $\forall S \in \mathcal{E}$, we have a factor ψ_S. All factors have the same range **D**.

$$\psi_S : \prod_{i \in S} \mathrm{Dom}(X_i) \to \mathbf{D}$$

Functional Aggregate Query: The Output



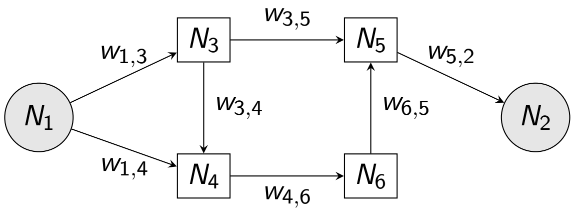
$\psi_{1,4}(X_1, X_4)$
$\psi_{1,3}(X_1, X_3)$
$\psi_{2,3}(X_2, X_3)$
$\psi_{1,2,4}(X_1, X_2, X_4)$
FAQ-Expression
$\varphi(x_1) = \sum_{X_2} \prod_{X_3} \max_{X_4} (\psi_{1,4}, \psi_{1,3}, \psi_{2,3}, \psi_{1,2,4})$
$\varphi(X_1)$

- Compute the function $\varphi : \prod_{i \in F} \text{Dom}(X_i) \to \mathbf{D}$.
- φ is defined by the **FAQ-Expression**:

$$\varphi(\mathbf{x}_{[F]}) = \bigoplus_{x_{f+1} \in \text{Dom}(X_{f+1})}^{(f+1)} \cdots \bigoplus_{x_n \in \text{Dom}(X_n)}^{(n)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S)$$

- For each $\oplus^{(i)}$, either $(\mathbf{D}, \oplus^{(i)}, \otimes, \mathbf{0}, \mathbf{1})$ is a commutative semiring, or $\oplus^{(i)} = \otimes$.

Path Query as FAQ (1/5)



- Variables $\mathcal{V}$: $\{X_1, \ldots, X_6\}$, where $\forall X_i \in \mathcal{V}$, $\texttt{Dom}(X_i) = V(G) = \{N_1, \ldots, N_6\}$.
- Free variables F: $\{X_1, X_2\}$ are assigned as the source and target vertices.
- Hyperedges $\mathcal{E}$: vertex pair $E(G) = V(G)^2$.
- Factors ψ_S: function $\mathcal{E} \to \mathbf{D}$, where $S \in \mathcal{E}$.

Path Query as FAQ (2/5)

$$\begin{aligned}
\varphi(\mathbf{x}_{[2]}) &= \bigoplus_{x_3,x_4,x_5,x_6 \in V(G)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \\
&= \psi(N_1, N_2) \oplus \left(\bigoplus_{x_3 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, N_2) \right) && \text{// 1 and 2 hops} \\
&= \dots && \text{// 3 and 4 hops} \\
&= \oplus \left(\bigoplus_{x_3,\dots,x_6 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, x_4) \otimes \dots \otimes \psi(x_6, N_2) \right) && \text{// 5 hops}
\end{aligned}$$

- **Shortest distance**: $\oplus = \min$, $\otimes = +$, ψ returns edge weights, $\mathbf{D} = \mathbb{R} \cup \{\infty\}$

Path Query as FAQ (3/5)

$$\varphi(\mathbf{x}_{[2]}) = \bigoplus_{x_3,x_4,x_5,x_6 \in V(G)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S)$$

$$= \psi(N_1, N_2) \oplus \left(\bigoplus_{x_3 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, N_2) \right) \qquad \text{// 1 and 2 hops}$$

$$= \dots \qquad \text{// 3 and 4 hops}$$

$$= \oplus \left(\bigoplus_{x_3,\dots,x_6 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, x_4) \otimes \dots \otimes \psi(x_6, N_2) \right) \qquad \text{// 5 hops}$$

- **Largest capacity**: $\oplus = \max$, $\otimes = \min$, ψ returns edge weights, $\mathbf{D} = \mathbb{R} \cup \{-\infty, \infty\}$

Path Query as FAQ (4/5)

$$\begin{aligned}
\varphi(\mathbf{x}_{[2]}) &= \bigoplus_{x_3,x_4,x_5,x_6 \in V(G)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \\
&= \psi(N_1, N_2) \oplus \left(\bigoplus_{x_3 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, N_2) \right) && \text{// 1 and 2 hops} \\
&= \dots && \text{// 3 and 4 hops} \\
&= \oplus \left(\bigoplus_{x_3,\dots,x_6 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, x_4) \otimes \dots \otimes \psi(x_6, N_2) \right) && \text{// 5 hops}
\end{aligned}$$

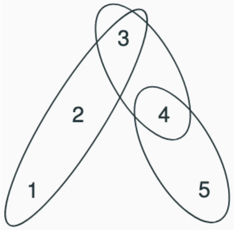
- **Connectivity**: $\oplus = \vee$, $\otimes = \wedge$, ψ returns edge existance, $\mathbf{D} = \{\mathtt{F}, \mathtt{T}\}$

Path Query as FAQ (5/5)

$$\begin{aligned}
\varphi(\mathbf{x}_{[2]}) &= \bigoplus_{x_3,x_4,x_5,x_6 \in V(G)} \bigotimes_{S \in \mathcal{E}} \psi_S(\mathbf{x}_S) \\
&= \psi(N_1, N_2) \oplus \left(\bigoplus_{x_3 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, N_2) \right) && \text{// 1 and 2 hops} \\
&= \dots && \text{// 3 and 4 hops} \\
&= \oplus \left(\bigoplus_{x_3,\dots,x_6 \in V(G)} \psi(N_1, x_3) \otimes \psi(x_3, x_4) \otimes \dots \otimes \psi(x_6, N_2) \right) && \text{// 5 hops}
\end{aligned}$$

- **Shortest path**: $\oplus = \cup$, $\otimes =$ **concat**, ψ returns edge itself or $\emptyset$, $\mathbf{D} = E(G) \cup \{\emptyset\}$

DB Query as FAQ (1/3)



$$\underbrace{\texttt{Orders(customer, day, dish)}}_{\psi_{123}(x_1,x_2,x_3)}$$

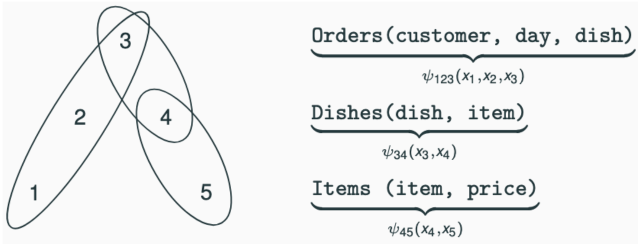
$$\underbrace{\texttt{Dishes(dish, item)}}_{\psi_{34}(x_3,x_4)}$$

$$\underbrace{\texttt{Items (item, price)}}_{\psi_{45}(x_4,x_5)}$$

- Q1: `SELECT * FROM Orders NATURAL JOIN Dish NATURAL JOIN Items;`
- **FAQ** over union-intersection semiring, where ψ maps tuple to $\{\emptyset, \{\text{tuple}\}\}$:

$$\varphi() = \bigcup_{x_1,x_2,x_3,x_4,x_5} \psi_{1,2,3}(x_1,x_2,x_3) \cap \psi_{3,4}(x_3,x_4) \cap \psi_{4,5}(x_4,x_5)$$

DB Query as FAQ (2/3)



$$\underbrace{\texttt{Orders(customer, day, dish)}}_{\psi_{123}(x_1,x_2,x_3)}$$

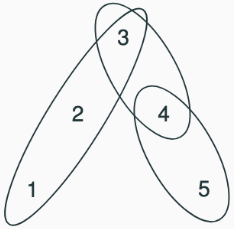
$$\underbrace{\texttt{Dishes(dish, item)}}_{\psi_{34}(x_3,x_4)}$$

$$\underbrace{\texttt{Items (item, price)}}_{\psi_{45}(x_4,x_5)}$$

- Q2: `SELECT customer,COUNT(*) from Q1 GROUP BY customer;`
- **FAQ** over sum-product semiring, where ψ maps tuple to $\{0,1\}$:

$$\varphi(x_1) = \sum_{x_2,x_3,x_4,x_5} \psi_{1,2,3}(x_1,x_2,x_3)\cdot\psi_{3,4}(x_3,x_4)\cdot\psi_{4,5}(x_4,x_5)$$

DB Query as FAQ (3/3)



$$\underbrace{\texttt{Orders(customer, day, dish)}}_{\psi_{123}(x_1,x_2,x_3)}$$

$$\underbrace{\texttt{Dishes(dish, item)}}_{\psi_{34}(x_3,x_4)}$$

$$\underbrace{\texttt{Items (item, price)}}_{\psi_{45}(x_4,x_5)}$$

- Q3: `SELECT customer,day,SUM(price) from Q1 GROUP BY customer,day;`
- **FAQ** over sum-product semiring, where $\psi_{4,5}$ maps (x_4, x_5) to x_5; others are the same as Q2:

$$\varphi(x_1, x_2) = \sum_{x_3,x_4,x_5} \psi_{1,2,3}(x_1, x_2, x_3) \cdot \psi_{3,4}(x_3, x_4) \cdot \psi_{4,5}(x_4, x_5)$$

Takeaway: A Unified Language

- FAQ is a unified language to express many problems in computer science.
- See appendix for more problems expressible in FAQ over different semirings.

Outline

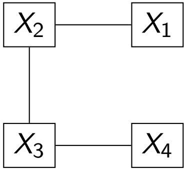
The Nature of FAQ

- A collection of factors.
- A hypergraph to guide the factor assembling.

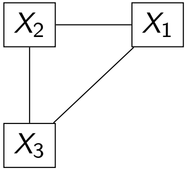
Hypergraphs: The Good and The Bad

Consider the following two FAQs. φ_2 is the same as φ_2 in the case when $X_4 = X_1$.

- **Acyclic FAQ**: $\varphi_1 = \bigoplus_{\mathbf{x}_{[4]}} \psi_{1,2}(x_1, x_2) \otimes \psi_{2,3}(x_2, x_3) \otimes \psi_{3,4}(x_3, x_4)$
- **Cyclic FAQ**: $\varphi_2 = \bigoplus_{\mathbf{x}_{[3]}} \psi_{1,2}(x_1, x_2) \otimes \psi_{2,3}(x_2, x_3) \otimes \psi_{3,1}(x_1, x_3)$

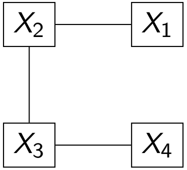


Hypergraph of φ_1.



Hypergraph of φ_2.

Example: The Acyclic FAQ over Boolean Semiring (1/2)



Consider the instance of φ_1 over the Boolean semiring:

$$\varphi = \bigvee_{\mathbf{x}_{[4]} \in \prod_{i \in [4]} \texttt{Dom}(X_i)} \psi_{1,2}(x_1, x_2) \wedge \psi_{2,3}(x_2, x_3) \wedge \psi_{3,4}(x_3, x_4)$$

φ asks whether there's a tuple $(x_1, \ldots, x_4)$ such that all factors $\psi_{i,j}(x_i, x_j) = \texttt{True}$.

Example: The Acyclic FAQ over Boolean Semiring (2/2)

$$\begin{tikzcd}
\boxed{X_2} \arrow[r, dash] \arrow[d, dash] & \boxed{X_1} \\
\boxed{X_3} \arrow[r, dash] & \boxed{X_4}
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{\varphi_{1,2}(X_1, X_2)} \\
\boxed{\varphi_{2,3}(X_2, X_3)} \arrow[u] \\
\boxed{\varphi_{3,4}(X_3, X_4)} \arrow[u]
\end{tikzcd}$$

$$\varphi = \bigvee_{\mathbf{x}_{[4]} \in \prod_{i \in [4]} \mathtt{Dom}(X_i)} \psi_{1,2}(x_1, x_2) \wedge \psi_{2,3}(x_2, x_3) \wedge \psi_{3,4}(x_3, x_4)$$

@$\varphi_{3,4}$ Send up x_4-values: $V_{(3,4)\to(2,3)}(x_4) = \bigvee_{x_3} \psi_{3,4}(x_3, x_4)$

@$\varphi_{2,3}$ Send up x_2-values: $V_{(2,3)\to(1,2)}(x_2) = \bigvee_{x_3} \psi_{2,3}(x_2, x_3) \wedge V_{(3,4)\to(2,3)}(x_3)$

@$\varphi_{1,2}$ Sum up: $\varphi() = \bigvee_{x_1} \psi_{1,2}(x_1, x_2) \wedge V_{(2,3)\to(1,2)}(x_2)$

The Power of Acyclicity

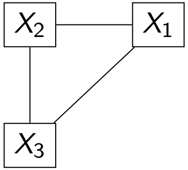
All computation steps are local and their cost upper bounded by the factor sizes

- Typical assumption: $|\psi_{i,j}| \leq N$ for some value N.
- We pass along at most N values between factors.
- Local computation is just filtering local values with incoming values.
- Overall: linear computation time - This is the best in the worst case.

Evaluation strategy know for decades under different names:

- Message passing [4] (in AI literatures)
- Semi-Join reduction [5] (in DB literatures)

The Bad Case: Cyclic FAQs (1/2)



$$\begin{tikzcd}
\boxed{\varphi_{1,2}(X_1, X_2)} \\
\boxed{\varphi_{2,3}(X_2, X_3)} \arrow[u] \\
\boxed{\varphi_{3,1}(X_3, X_1)} \arrow[u]
\end{tikzcd}$$

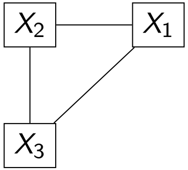
$$\varphi' = \bigvee_{\mathbf{x}_{[3]} \in \prod_{i \in [3]} \mathtt{Dom}(X_i)} \psi_{1,2}(x_1, x_2) \wedge \psi_{2,3}(x_2, x_3) \wedge \psi_{3,1}(x_3, x_1)$$

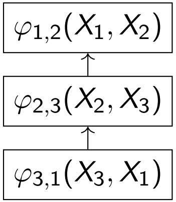
@$\varphi_{3,1}$ Send up (x_1, x_3)-values: $V_{(3,1)\to(2,3)}(x_1, x_3) = \psi_{3,1}(x_3, x_1)$

@$\varphi_{2,3}$ Send up (x_1, x_2)-values: $V_{(2,3)\to(1,2)}(x_1, x_2) = \bigvee_{x_3} \psi_{2,3}(x_2, x_3) \wedge V_{(3,1)\to(2,3)}(x_1, x_3)$

@$\varphi_{1,2}$ Sum up: $\varphi'() = \bigvee_{x_1, x_2} \psi_{1,2}(x_1, x_2) \wedge V_{(2,3)\to(1,2)}(x_1, x_2)$

The Bad Case: Cyclic FAQs (2/2)




$\varphi_{1,2}(X_1, X_2)$
$\varphi_{2,3}(X_2, X_3)$
$\varphi_{3,1}(X_3, X_1)$

$$\varphi' = \bigvee_{\mathbf{x}_{[3]} \in \prod_{i\in[3]} \texttt{Dom}(X_i)} \psi_{1,2}(x_1, x_2) \wedge \psi_{2,3}(x_2, x_3) \wedge \psi_{3,1}(x_3, x_1)$$

$V_{(2,3)\to(1,2)} = \bigvee_{x_3} \psi_{2,3}(x_2, x_3) \wedge V_{(3,1)\to(2,3)}(x_1, x_3)$ introduces $O(N^2)$ cost.

- It's a join instead of a semi-join.

A Roadmap for Further Study

1. Can we distinguish syntatically the acyclic from the cyclic hypergraphs?
   - α-acyclic, β-acyclic, free-connex, *etc.*
2. How to transform cyclic hypergraphs to acyclic ones?
   - Hypertree decomposition.
3. How to measure the goodness of such transformations?
   - Width measures: hypertree width, fractional hypertree width, *etc.*
4. How to design efficient algorithms for FAQs over (commutative) semirings?
   - InsideOut algorithm [1].

Problems Expressible in FAQ over Boolean Semiring

$(\{\mathtt{F}, \mathtt{T}\}, \vee, \wedge, \mathtt{F}, \mathtt{T})$

■ Constraint satisfaction problems (CSP)	FAQ [1]
■ Boolean conjunctive query evaluation (BCQ)	FAQ [1]
■ Conjunctive query evaluation (CQ)[5]	FAQ [1]
■ Join evaluation	FAQ [1]
■ Satisfiability (SAT)	FAQ [1]
■ *k*-colorability	FAQ [1]
■ List recovery problem (coding theory)	FAQ [1]

[5]it's also expressible using the set semiring.

Problems Expressible in FAQ over Set Sum-Product Semiring

$(2^{\mathcal{U}}, \cup, \cap, \emptyset, \mathcal{U})$

■ Conjunctive query evaluation (CQ)[6]	FAQ [1]
■ Join evaluation	FAQ [1]

[6]It's also expressible using the Boolean semiring.

Problems Expressible in FAQ over Natural Sum-Product Semiring

$(\mathbb{N}, +, \times, 0, 1)$

- Complex network analysis FAQ [1]
- Count constraint satisfaction problems (#CSP) FAQ [1]
- Count satisfiability (#SAT) FAQ [1]

Problems Expressible in FAQ over Real Sum-Product Semiring

$(\mathbb{R}, +, \times, 0, 1)$

Permanent	FAQ [1]
Discrete Fourier transform	FAQ [1],AjiMcEl [2]
Hadamard transform	AjiMcEl [2]
Inference in probabilistic graphical models	FAQ [1]
Probability propagation in AI	AjiMcEl [2]
Matrix chain multiplication	FAQ [1],AjiMcEl [2]
Graph homomorphism	FAQ [1]
BCJR decoding (Bahl, Cocke, Jelinek, Raviv)	AjiMcEl [2]
Holant problem	FAQ [1]

Problems Expressible in FAQ over Max-Product Semiring

$([0, \infty), \max, \times, 0, 1)$

- MAP queries in probabilistic graphical models FAQ [1]
- Quantified conjunctive query evaluation (QCQ)[7] FAQ [1]

[7] It's also expressible using the max-product, min-product semirings.

Problems Expressible in FAQ over Min-Sum Semiring

$((-\infty, \infty], \min, +, \infty, 0)$

■ Gallager-Tanner-Wiberg decoding	AjiMcEl [2]
■ Viterbi decoding	AjiMcEl [2]
■ Trellis path problem	AjiMcEl [2]
■ Graph optimization	KohlWils [3]
■ Queuing systems	KohlWils [3]
■ Discrete event systems	KohlWils [3]
■ Optimization for weighted CSPs	KohlWils [3]

Problems Expressible in FAQ over Two Semiring

$([0, \infty), \max, \times, 0, 1), ((0, \infty], \min, \times, \infty, 1)$

- Quantified conjunctive query evaluation (QCQ)[8] FAQ [1]

$(\mathbb{N}, \max, \times, 0, 1), (\mathbb{N}, +, \times, 0, 1)$

- Count conjunctive query evaluation (#CQ) FAQ [1]
- Count quantified conjunctive query evaluation (#QCQ) FAQ [1]

[8] It's also expressible using the max-product semiring

References I

- Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra.
  Faq: Questions asked frequently.
  In Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '16, page 13–28, 2016.

- S.M. Aji and R.J. McEliece.
  The generalized distributive law.
  IEEE Transactions on Information Theory, 46(2):325–343, 2000.

- Jürg Kohlas and Nic Wilson.
  Semiring induced valuation algebras: Exact and approximate local computation algorithms.
  Artif. Intell., 172(11):1360–1399, 2008.

References II

Judea Pearl.
Fusion, propagation, and structuring in belief networks.
In Probabilistic and Causal Inference: The Works of Judea Pearl, pages 139–188. 2022.

Mihalis Yannakakis.
Algorithms for acyclic database schemes.
In VLDB, volume 81, pages 82–94, 1981.