Northeastern University
Khoury College of Computer Sciences

DATALAB

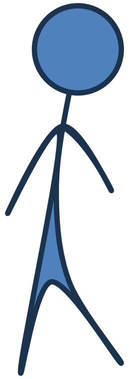
Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries

Neha Makhija
Northeastern University
(Joint work with Wolfgang Gatterbauer)

PODS, Santiago, Chile - June 12, 2024

https://northeastern-datalab.github.io/unified-reverse-data-management/

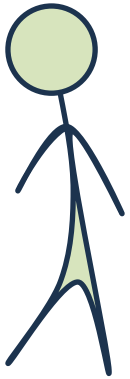
Minimal Factorization



I have a long Boolean formula!

$$r_1s_1t_1+r_1s_1t_2+r_1s_1t_3+r_1s_1t_4+r_1s_2t_3+r_1s_2t_4+r_1s_2t_5+r_2s_1t_1+r_2s_1t_2+r_2s_1t_3+r_2s_2t_4+r_2s_2t_5+r_3s_1t_1+r_3s_1t_2+r_3s_1t_3+r_3s_2t_4+r_3s_2t_5+r_4s_1t_1+r_4s_1t_2+r_4s_1t_3+r_4s_2t_4+r_4s_2t_5+r_5s_1t_1+r_5s_1t_2+\ldots$$

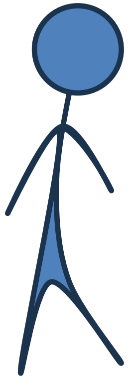
Why don't you just factorize it?



$$(r_1+r_2+r_3+r_4+r_5)(s_1(t_1+t_2+t_3)+ s_2(t_4+t_5))+(r_1+r_2)s_1(t_3+t_4)$$

This expression is equivalent, and I computed it **efficiently**!

Minimal Factorization

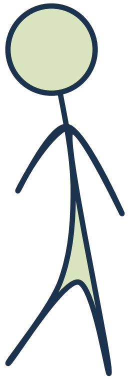


How did you compute that so efficiently?!!

Result. [BU2011] *(informal)*

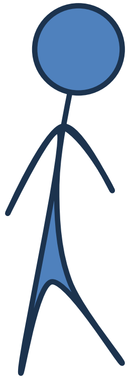
The Minimum Equivalent Expression problem is Σ_2^p-complete

You're right, I used some extra information!
I saw you got this expression from a query result….



David Buchfuhrer and Christopher Umans. 2011. The complexity of Boolean formula minimization. JCSS2011, https://doi.org/10.1016/j.jcss.2010.06.011

Minimal Factorization



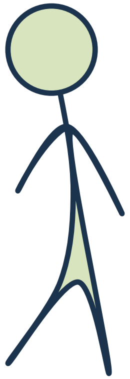
Ah! I see – you computed a Read-Once Expression in PTIME!

Result. [G1997] *(informal)*

A factorization without repeated literals can always be found in PTIME

No, the generated expression was not read-once

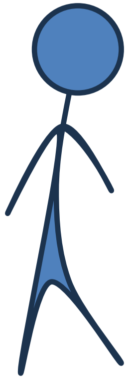
$$(r_1+r_2+r_3+r_4+r_5)(s_1(t_1+t_2+t_3)+ s_2(t_3+t_4))+(r_1+r_2)s_1(t_3+t_4)$$



V.A. Gurvich, On repetition-free Boolean functions, Uspekhi Mat. Nauk. 32 (1977) 183–184 (in Russian, also, On read-once Boolean functions, Russian Math. Surveys 32 (1977) 183–184)

Neha Makhija, Wolfgang Gatterbauer. Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries, PODS 2024. https://northeastern-datalab.github.io/unified-reverse-data-management/

Minimal Factorization



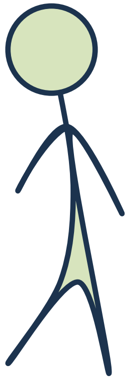
Is it asymptotically optimal like in Factorized Databases?
Is it an approximation?

Result. [OZ2015] *(informal)*

Worst-Case optimal algorithms exist to find factorizations (f-reps) of CQs

No, this is instance optimal – the smallest possible formula!

*We have **new** tractable cases for minimally factorizing **provenance formulas***



Dan Olteanu and Jakub Závodný, Size Bounds for Factorised Representations of Query Results, TODS 2015, https://doi.org/10.1145/2656335

Neha Makhija, Wolfgang Gatterbauer. Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries, PODS 2024. https://northeastern-datalab.github.io/unified-reverse-data-management/

Provenance Formulas

R

x	y	
1	1	r_1
2	1	r_2

S

y	z	
1	1	s_1
1	2	s_2

Provenance Tokens

Input

```
R(x,y),S(y,z)
```

```
select *
from R, S
where R.y = S.y
```

Query

x	y	z	
1	1	1	$r_1 s_1$
1	1	2	$r_1 s_2$
2	1	1	$r_2 s_1$
2	1	2	$r_2 s_2$

Output

$$r_1 s_1 + r_2 s_1 + r_1 s_2 + r_2 s_2$$
$$(r_1 + r_2)(s_1 + s_2)$$

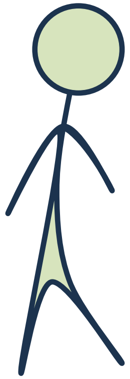
Provenance

Provenance semirings. Green, Karvounarakis, Tannen, PODS 2007. https://doi.org/10.1145/1265530.1265535

Minimal Factorization of Provenance Formulas

Provenance formula is a
- k-partite
- monotone Boolean formula
- that follows join dependencies of the Conjunctive Query

Can we leverage this to make the factorization problem easier?



Restrictions

- ***Self-Join Free Conjunctive Queries***
- ***Input = Provenance DNF***
- ***Minimum formula != Minimum Circuit -> no memoization allowed***

Neha Makhija, Wolfgang Gatterbauer. Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries, PODS 2024. https://northeastern-datalab.github.io/unified-reverse-data-management/

- Problem Setup
- **Motivation**
- Contributions
- Takeaways + Open Questions

Motivation (1/2): Boolean Factorization

- Boolean Factorization is a fundamental problem

- It has led to deep and surprising complexity results

1972 — [Meyer, Stockmeyer]: defined the Minimal Equivalent Expression (MEE) problem

1979 — [Garey Johnson]: Proved MEE to be to be NP-hard

2011 — [Buchfuhrer, Umans]: proved it to be Σ_2^P-complete

(The paper appendix contains an overview of results in this area)

- There are very few known PTIME subcases – Read Once, Read Polarity Once

Motivation (2/2): Probabilistic Inference

Query-level Approaches

Partial Solution

DS[VLDB'04]
Hierarchical Queries

Nilesh N. Dalvi, Dan Suciu. Efficient Query Evaluation on Probabilistic Databases, VLDB 2004, https://doi.org/10.1016/B978-012088469-8.50076-0

Motivation (2/2): Probabilistic Inference

	Query-level Approaches	Data-level Approaches
Partial Solution	DS[VLDB'04] *Hierarchical Queries*	RPT[ICDT'11] *Read-Once Expressions – recovers DS04 as special case*

Nilesh N. Dalvi, Dan Suciu. Efficient Query Evaluation on Probabilistic Databases, VLDB 2004, https://doi.org/10.1016/B978-012088469-8.50076-0
Sudeepa Roy, Vittorio Perduca, and Val Tannen. Faster query answering in probabilistic databases using read-once functions, ICDT 2011, https://doi.org/10.1145/1938551.1938582

Motivation (2/2): Probabilistic Inference

	Query-level Approaches	Data-level Approaches
Partial Solution	DS[VLDB'04] *Hierarchical Queries*	RPT[ICDT'11] *Read-Once Expressions – recovers DS04 as special case*
Complete Solution	GS[VLDB'15] *Dissociations – recovers DS04 as special case*	

Nilesh N. Dalvi, Dan Suciu. Efficient Query Evaluation on Probabilistic Databases, VLDB 2004, https://doi.org/10.1016/B978-012088469-8.50076-0

Sudeepa Roy, Vittorio Perduca, and Val Tannen. Faster query answering in probabilistic databases using read-once functions, ICDT 2011, https://doi.org/10.1145/1938551.1938582

Wolfgang Gatterbauer, Dan Suciu. Dissociation and propagation for approximate lifted inference with standard relational DBMS. VLDBJ. https://doi.org/10.1007/s00778-016-0434-5

Motivation (2/2): Probabilistic Inference

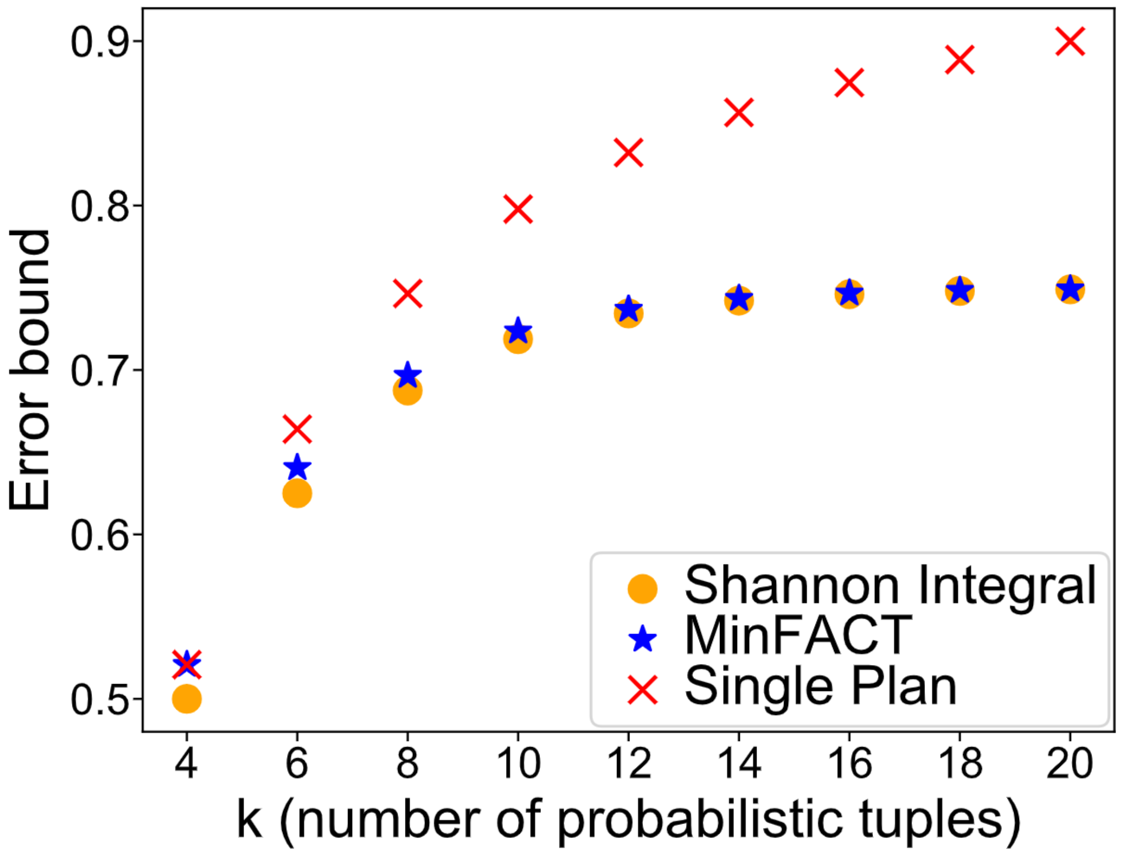
	Query-level Approaches	Data-level Approaches
Partial Solution	DS[VLDB'04] *Hierarchical Queries*	RPT[ICDT'11] *Read-Once Expressions – recovers DS04 as special case*
Complete Solution	GS[VLDB'15] *Dissociations – recovers DS04 as special case*	Our Approach *MinFACT -recovers DS04, RPT11, GS15 as special cases*

- *always PTIME (thus complete)*
- *Exact when Possible, else approximate*

Nilesh N. Dalvi, Dan Suciu. Efficient Query Evaluation on Probabilistic Databases, VLDB 2004, https://doi.org/10.1016/B978-012088469-8.50076-0
Sudeepa Roy, Vittorio Perduca, and Val Tannen. Faster query answering in probabilistic databases using read-once functions, ICDT 2011, https://doi.org/10.1145/1938551.1938582
Wolfgang Gatterbauer, Dan Suciu. Dissociation and propagation for approximate lifted inference with standard relational DBMS. VLDBJ. https://doi.org/10.1007/s00778-016-0434-5
Neha Makhija, Wolfgang Gatterbauer. Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries, PODS 2024. https://northeastern-datalab.github.io/unified-reverse-data-management/

Motivation (2/2): Probabilistic Inference

Experimentally, smaller factorized expressions lead to better probabilistic inference



- Problem Setup
- Motivation
- Contributions
  - #1: Connections between Factorizations ↔ Minimal Query Plans
  - #2: Two "Unified" Algorithms
  - #3: New Tractability Results
- Takeaways + Open Questions

C1: Minimal Factorizations ↔ Query Plans

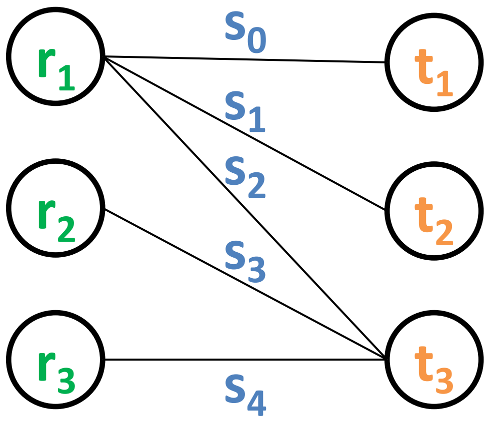
`Q():-R(x),S(x,y),T(y)`

Intuition #1: Evaluating data with different query plans corresponds to different "provenance factorizations"

R	x
r_1	1
r_2	2
	3

S	x	y
s_1	1	1
s_2	1	2
s_3	1	3
s_4	2	3
s_5	3	3

T	y
t_1	1
t_2	2
t_3	3



$\pi_{-x,y} \bowtie \langle R(x), S(x,y), T(y) \rangle$ → $r_1s_1t_1 + r_1s_2t_2 + r_1s_3t_3 + r_2s_4t_3 + r_3s_5t_3$

Length = 15

$\pi_{-x} \bowtie \langle R(x), \pi_{-y} \bowtie \langle S(x,y), T(y) \rangle \rangle$ → $r_1(s_1t_1 + s_2t_2 + s_3t_3) + r_2(s_4t_3) + r_3(s_5t_3)$

Length = 13

$\pi_{-y} \bowtie \langle \pi_{-x} \bowtie \langle S(x,y), R(x) \rangle, T(y) \rangle$ → $(r_1s_1)t_1 + (r_1s_2)t_2 + (r_1s_3 + r_2s_4 + r_3s_5)t_3$

Length = 13

C1: Minimal Factorizations ↔ Query Plans

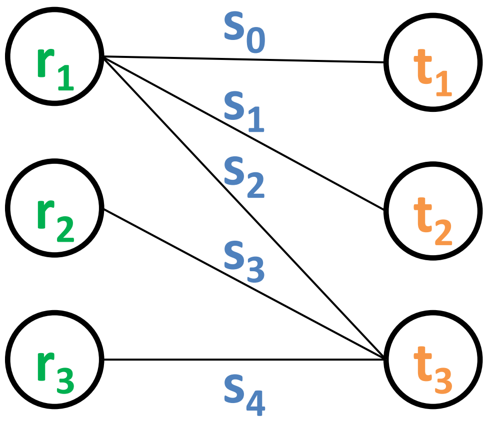
`Q():-R(x),S(x,y),T(y)`

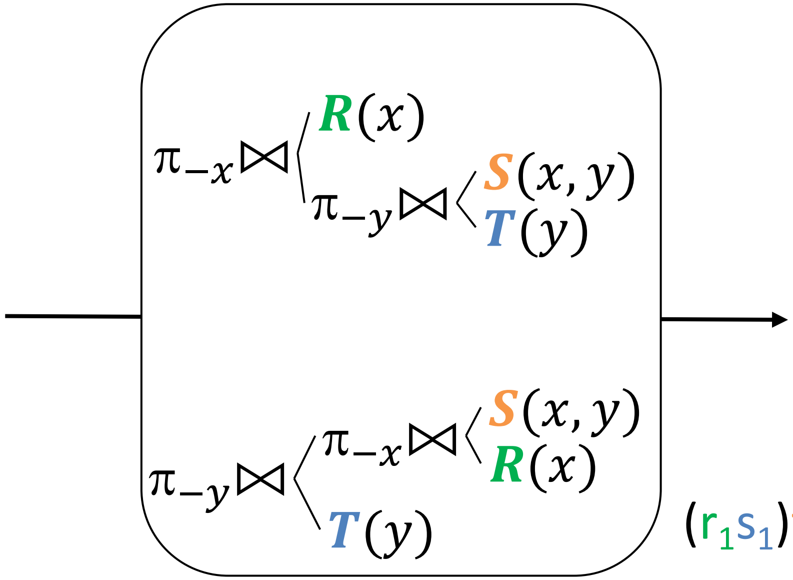
Intuition #2: A single factorization can leverage multiple query plans

R	x
r_1	1
r_2	2
	3

S	x	y
s_1	1	1
s_2	1	2
s_3	1	3
s_4	2	3
s_5	3	3

T	y
t_1	1
t_2	2
t_3	3





$r_1(s_1t_1 + s_2t_2 + s_3t_3) + (r_2 s_4 + r_3 s_5)t_3$

Length = 12

Smallest factorization
but not read-once!

$(r_1s_1)t_1 + (r_1s_2)t_2 + r_1(s_3t_3) + r_2(s_4t_3) + r_3(s_5t_3)$

Length = 15

Intuition #3: Assign plans to DNF terms

C1: Minimal Factorizations ↔ Query Plans

Factorized formula φ

↑

Several intermediate constructions

Not in this talk ☹

↓

Assignment of Query Plans to Query answers

Theorem. *(informal)*

The minimal factorization for sjf CQ provenance can always be recovered an assignment of Query Plan to each term in the provenance DNF

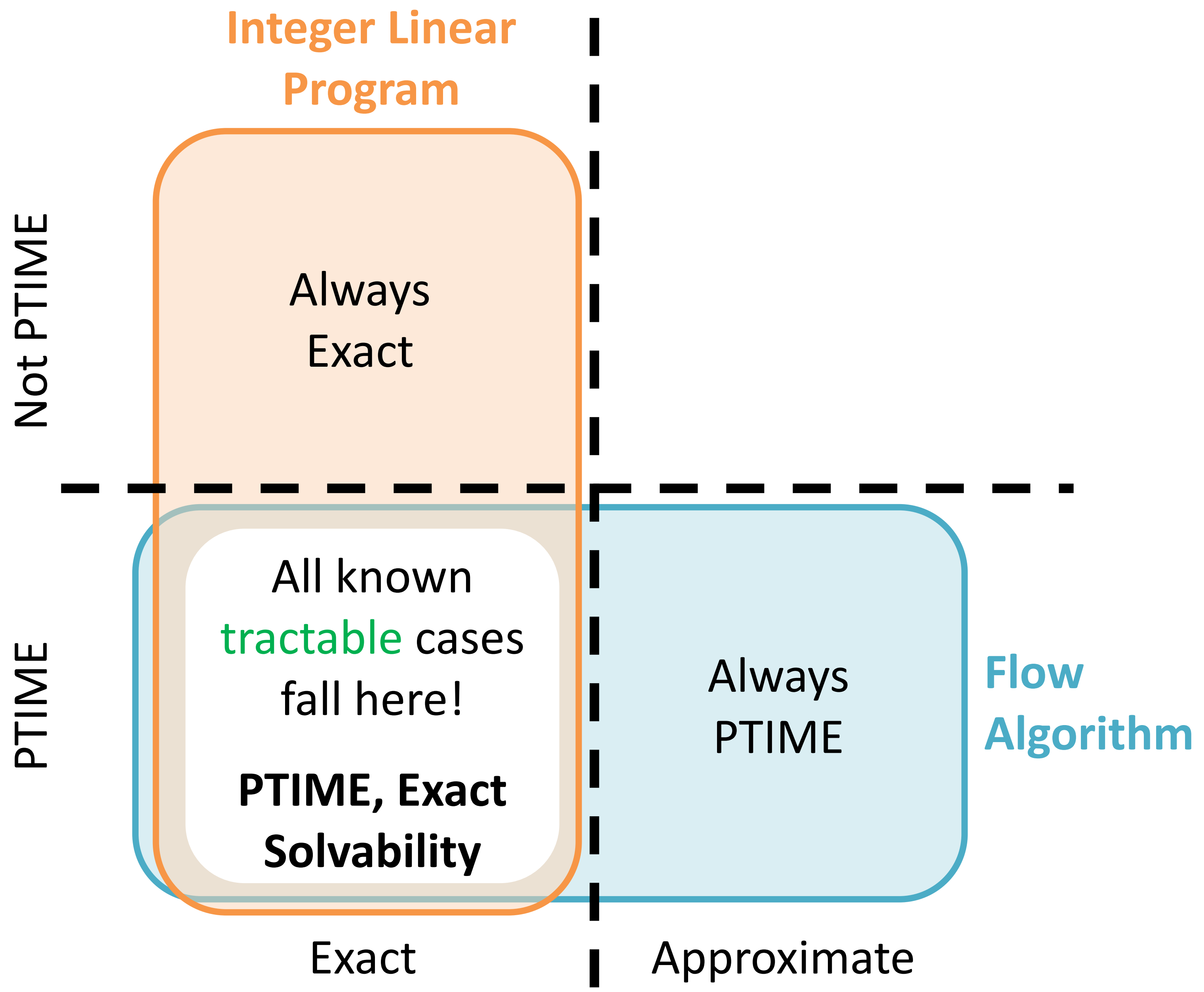
Theorem. *(informal)*

We don't need to look at all Query Plans – just **minimal** ones

*minimal query plans = concept from probabilistic databases (query dissociation)

- Problem Setup
- Motivation
- Contributions
  - #1: Connections between Factorizations ↔ Minimal Query Plans
  - #2: Two “Unified” Algorithms
  - #3: New Tractability Results
- Takeaways + Open Questions

C2: Two Unified Algorithms



C2: Two Unified Algorithms: ILP Intuition

Q():-R(x),S(x,y),T(y)

For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint

$$qp_1[w_1] + qp_2[w_1] \geq 1$$

0-1 Integer Variables

For "chosen" query plan, all relevant prefix paths are "chosen" too

→ Prefix Constraint

$$p_1[qp_1[w_1]] \geq qp_1[w_1]$$

C2: Two Unified Algorithms: ILP Intuition

Q():-R(x),S(x,y),T(y)

Objective: Maximize repeated use of prefixes!

Each prefix has a “cost” – we want to minimize the weighted sum of prefix costs

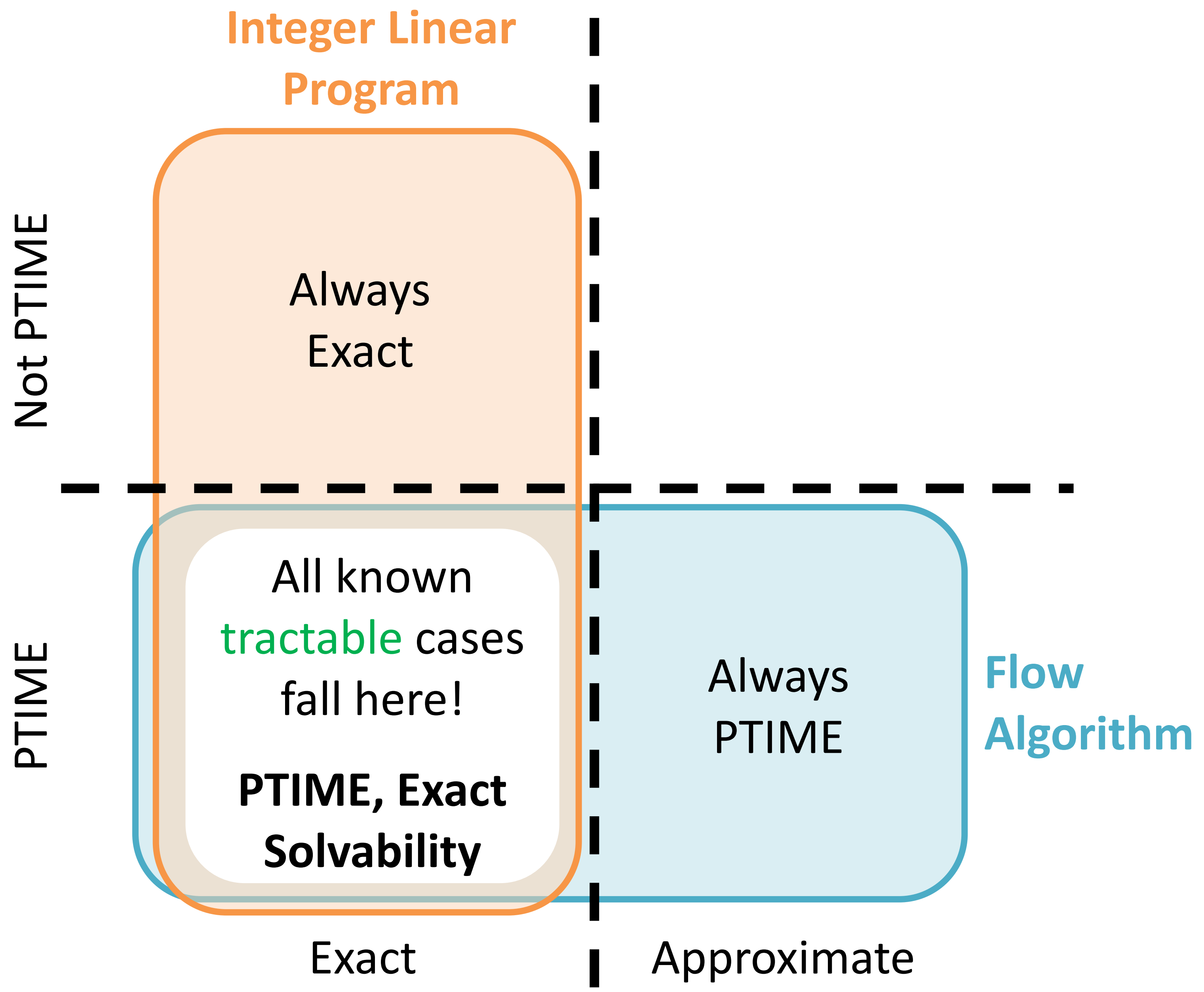
$$\sum_{i,j,k} c(p_i) \times \boxed{p_i[qp_j[w_k]]}$$

We have all the ingredients for an ILP!

Theorem. *(informal)*

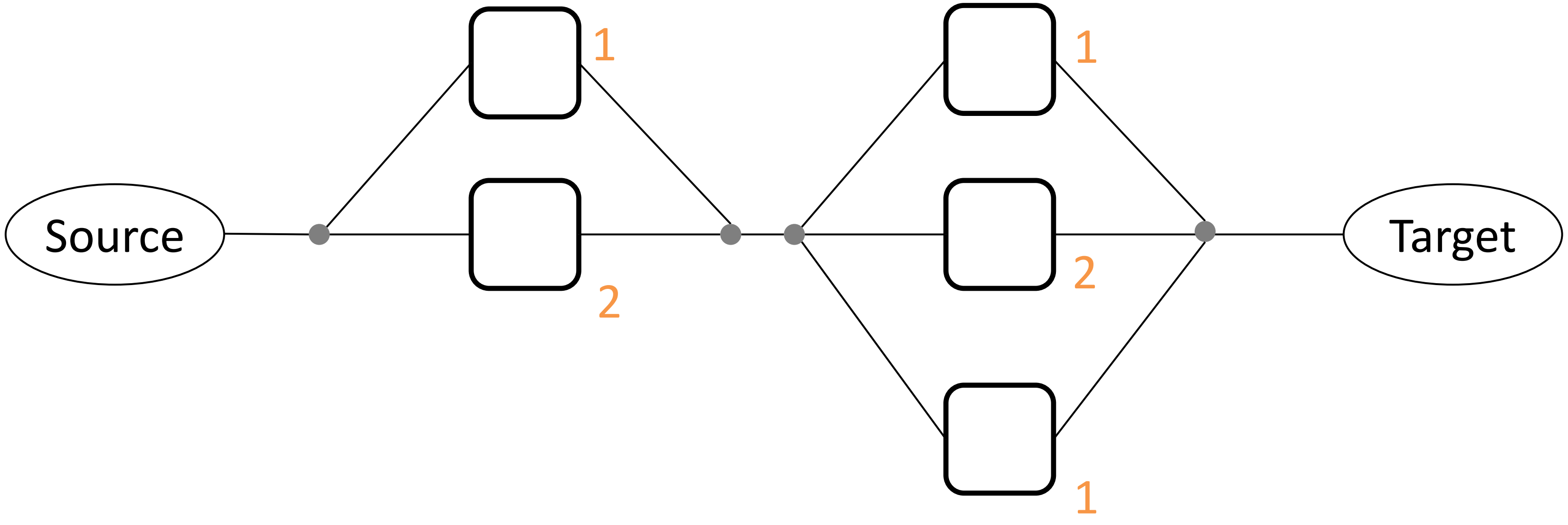
For all known PTIME cases of MinFACT, the objective value of the LP relaxation is identical to the ILP.

C2: Two Unified Algorithms



C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

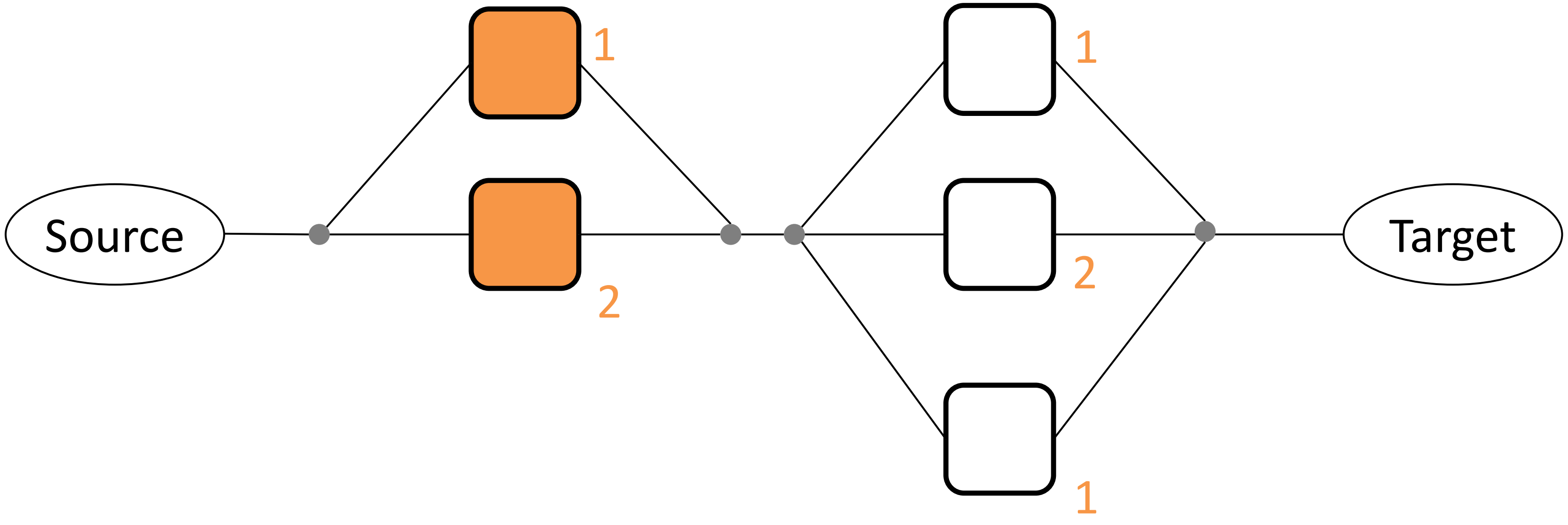
We model the problem as a MINCUT problem



MINCUT = Min cost **nodes** whose removal disconnects source and target

C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem



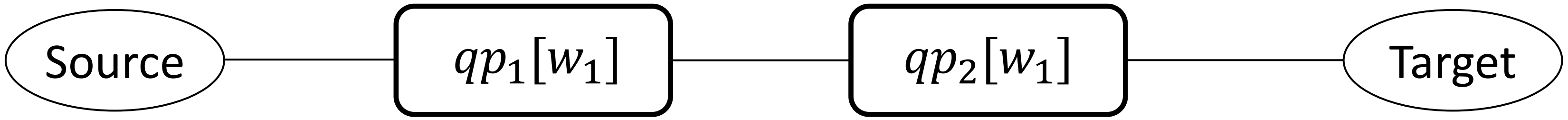
MINCUT = Min cost **nodes** whose removal disconnects source and target

C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem

For every DNF term in the provenance, at least one query plan must be chosen

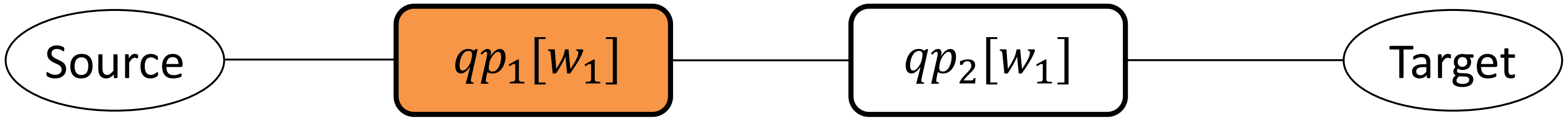
→ Unique Query Plan Constraint



C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem
For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint

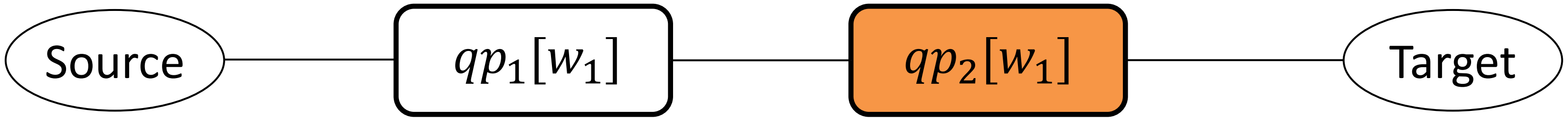


C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem

For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint

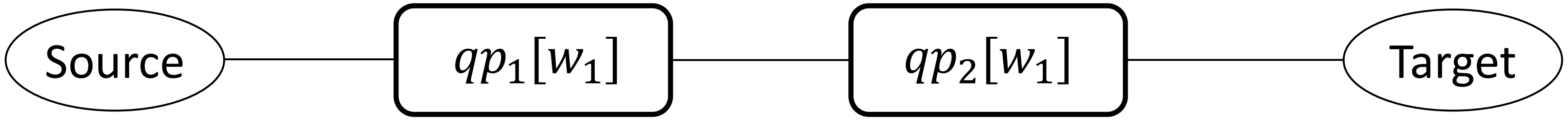


C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem

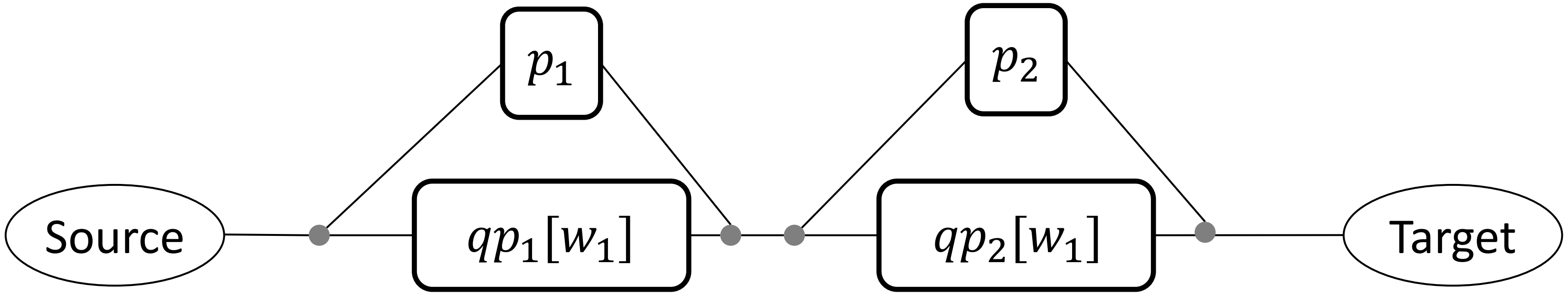
For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint



For "chosen" query plan, all relevant prefix paths are "chosen" too

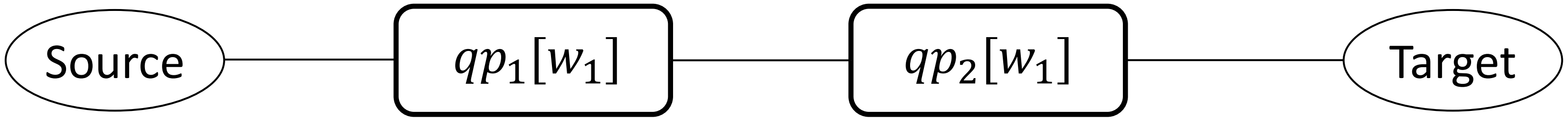
→ Prefix Constraint



C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

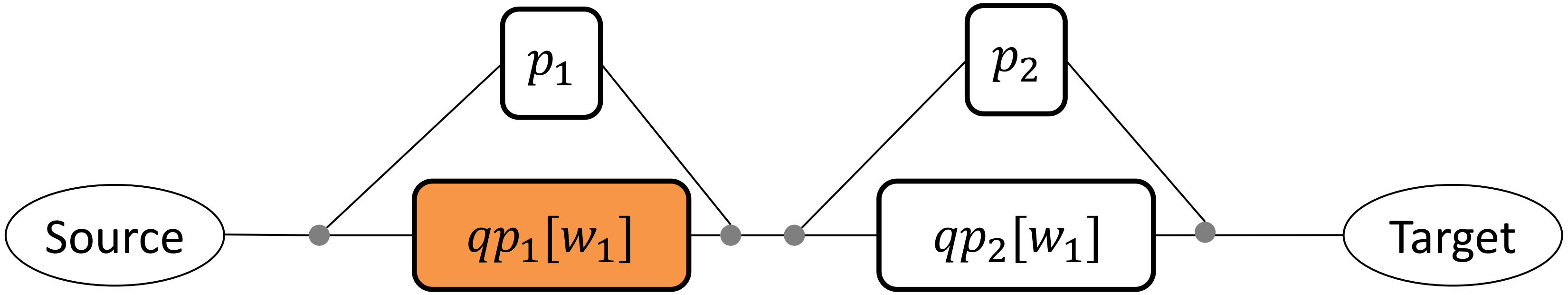
We model the problem as a MINCUT problem
For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint



For “chosen” query plan, all relevant prefix paths are “chosen” too

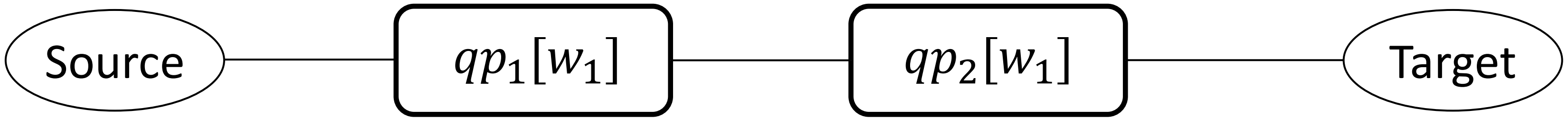
→ Prefix Constraint



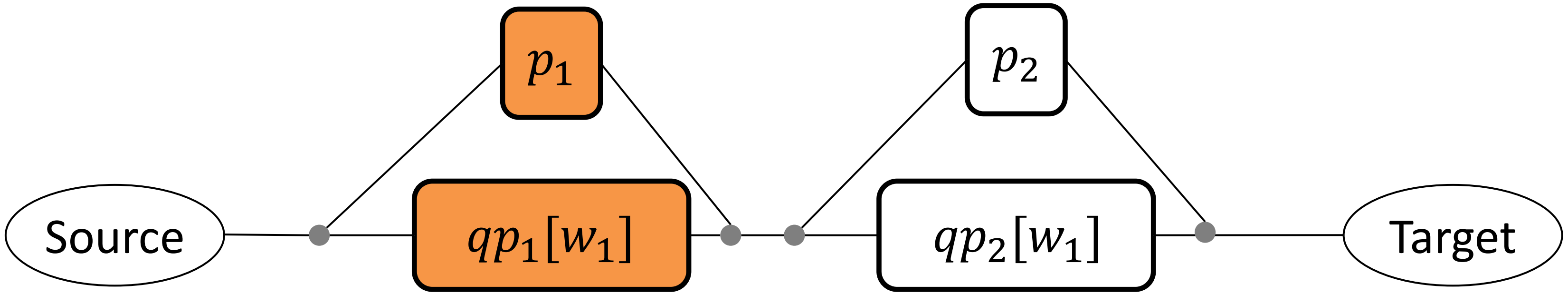
C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

We model the problem as a MINCUT problem
For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint



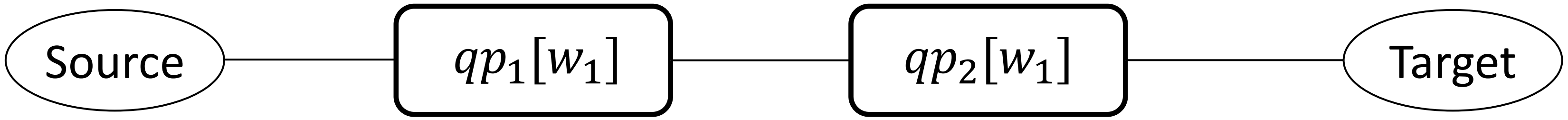
For “chosen” query plan, all relevant prefix paths are “chosen” too
→ Prefix Constraint



C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`

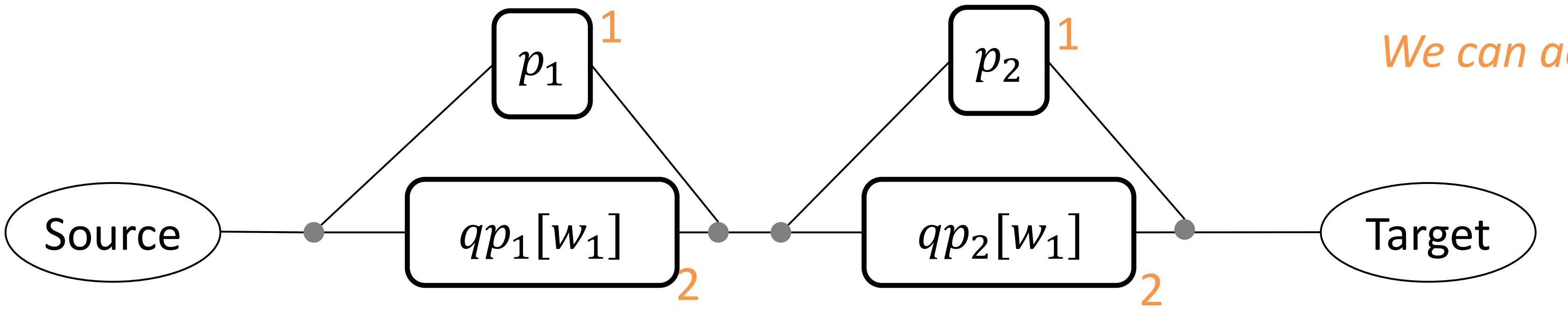
We model the problem as a MINCUT problem
For every DNF term in the provenance, at least one query plan must be chosen

→ Unique Query Plan Constraint



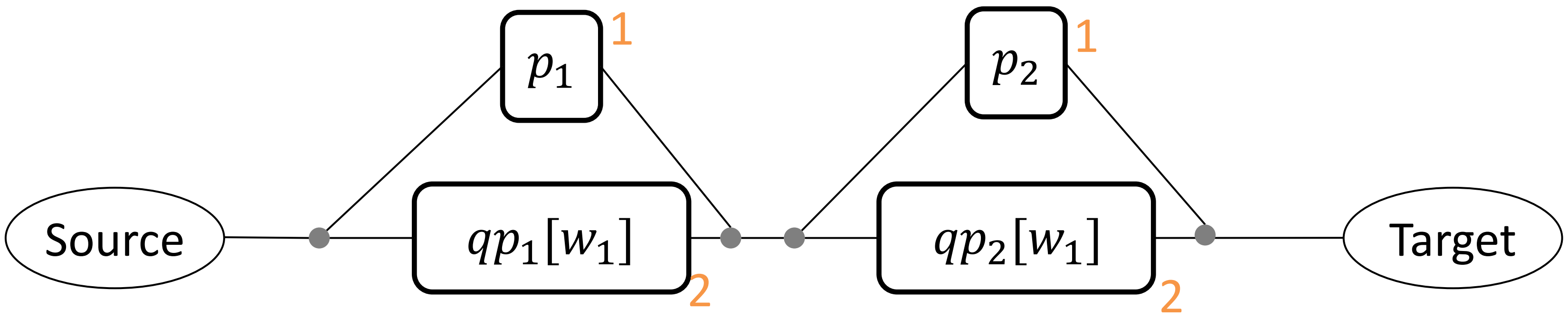
For "chosen" query plan, all relevant prefix paths are "chosen" too

→ Prefix Constraint



We can add costs as well!

C2: Two Unified Algorithms: Flow Intuition `Q():-R(x),S(x,y),T(y)`



The paths model ***all*** constraints.

→ But extra paths could imply additional constraints

→ Causing over-approximation

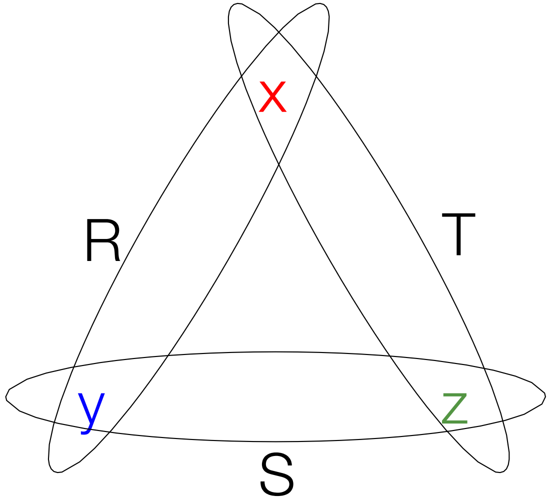
→ Ordering the nodes differently can lead to different results!

Theorem. *(informal)*

For all known PTIME cases of MinFACT, there is an ordering of VEOs such that MINCUT = MinFACT

- Problem Setup
- Motivation
- Contributions
  - #1: Connections between Factorizations ↔ Minimal Query Plans
  - #2: Two "Unified" Algorithms
  - #3: New Tractability Results
- Takeaways + Open Questions

When is Minimal Factorization Tractable?

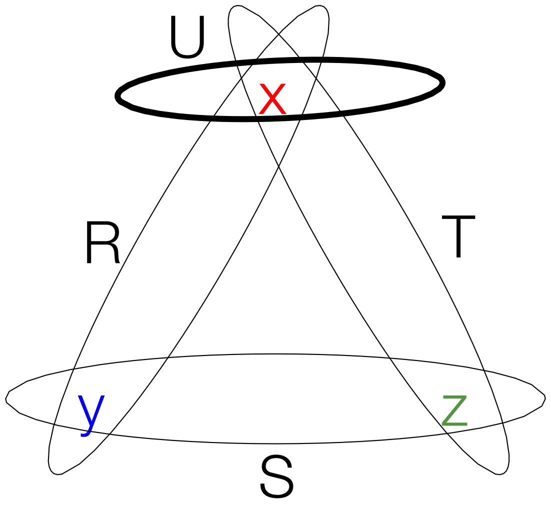


$Q_\Delta\text{:-}R(x,y)S(y,z)T(z,x)$

3 minimal Query Plans

Contains cycle

NP-Complete!



$Q_{\Delta U}\text{:-}\boldsymbol{U(x)}R(x,y)S(y,z)T(z,x)$

3 minimal Query Plans

Contains cycle

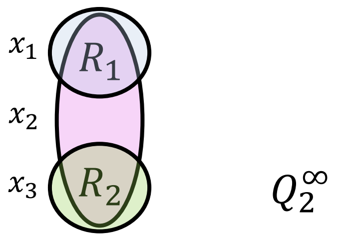
PTIME!

When is Minimal Factorization Tractable?

If query has $\mathbf{1}$ minimal query plan

→ It is hierarchical, and all provenance is read-once

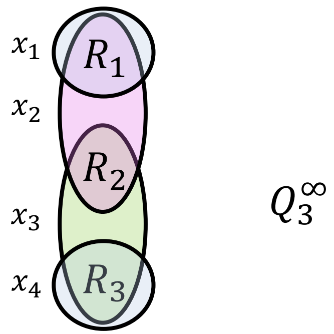
→ MinFACT is PTIME (previously known)



If query has $\leq \mathbf{2}$ minimal query plans

→ We **prove it's PTIME**!

→ Proof: ILP Constraint Matrix is Totally Unimodular

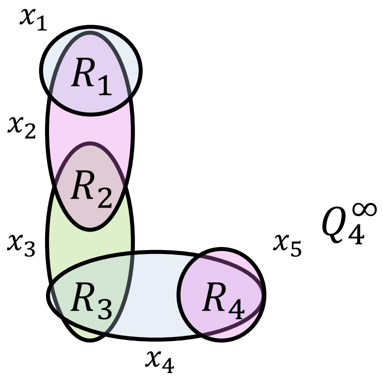


If query has $\mathbf{3} +$ minimal query plans

→ **Open**

→ We **prove two PTIME** queries with 3 and 5 plans

→ Proof: Detailed analysis of all possible extra constraints in flow graph



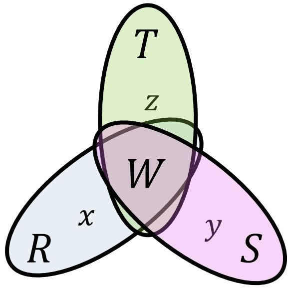
When is Minimal Factorization NP-Complete?

If query has an “*active triad*”*

→ NP-Complete

→ Proof: Reduction from Vertex Cover

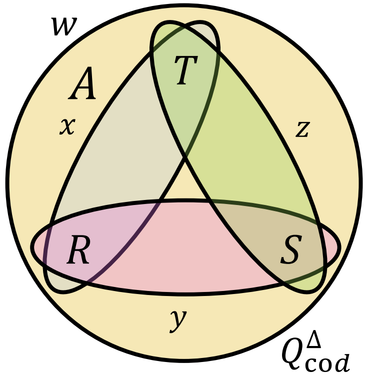
→ Same hardness condition as *Resilience*



If query has a “*co-deactivated triad*”*

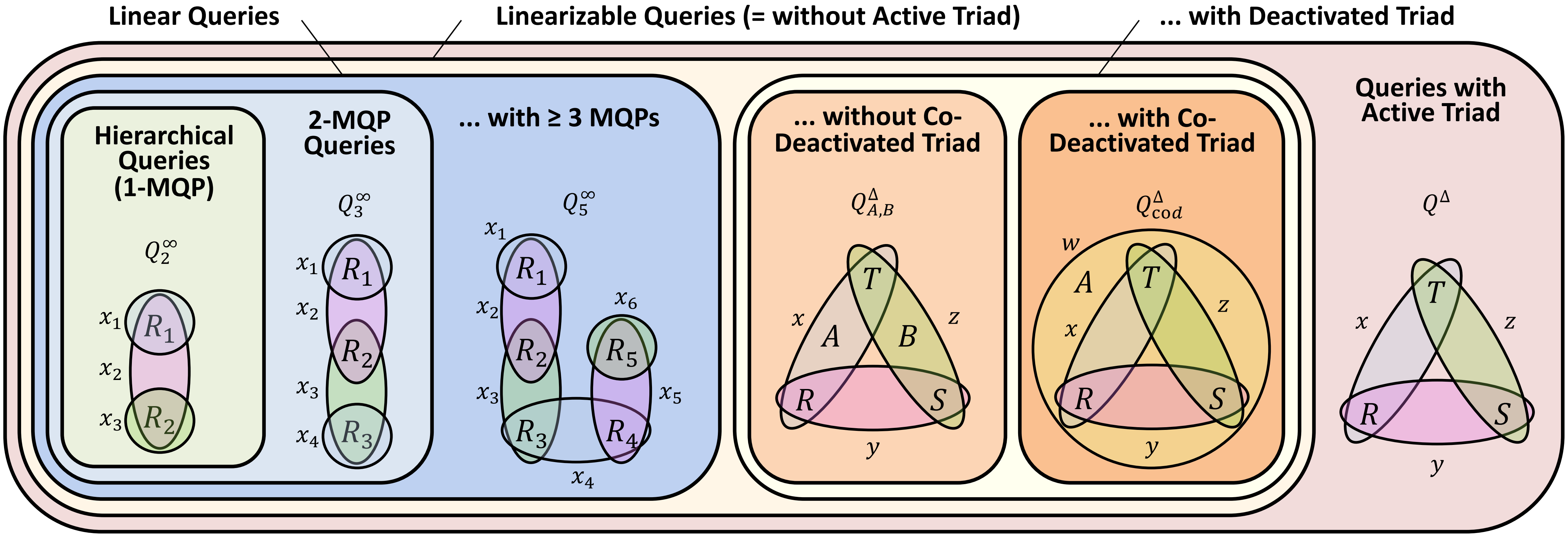
→ NP-Complete

→ Proof: Reduction from Vertex Cover



* = Definitions in paper

Minimal Factorization Complexity Sandwich



RES(Q)	PTIME	PTIME	PTIME	PTIME	PTIME	NP-C
FACT(Q)	PTIME	PTIME	?	?	NP-C	NP-C
Prob(Q)	PTIME	#P-hard	#P-hard	#P-hard	#P-hard	#P-hard

- Problem Setup
- Motivation
- Contributions
  - #1: Connections between Factorizations ↔ Minimal Query Plans
  - #2: Two “Unified” Algorithms
  - #3: New Tractability Results
- Takeaways + Open Questions

Open Problems

- Complete the complexity dichotomy
  - Conjecture: All path queries are tractable
  - Challenge #1: Go beyond existing ILP tractability criteria like Total Unimodularity
  - Challenge #2: Deal with many types of extra paths in flow graphs
- Generalize: self-joins, bag semantics, non-provenance formulas

Take-aways

- New tractable cases for factorization beyond read-once
- Unified algorithms: automatically optimally for all tractable cases
- Deep connections between dissociations + factorizations

Many more details, proofs, experiments, approximations:

- https://northeastern-datalab.github.io/unified-reverse-data-management/

Also see:

- Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations, SIGMOD 2024

Appendix

When is Minimal Factorization NP-Complete?

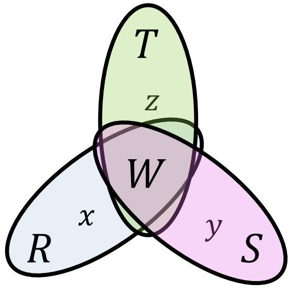
If query has an "*active triad*"

→ NP-Complete

→ Proof: Reduction from Vertex Cover

Triad: Have 3 atoms R,S,T that have an independent path to each other

Active triad: Have 3 independent atoms in triad

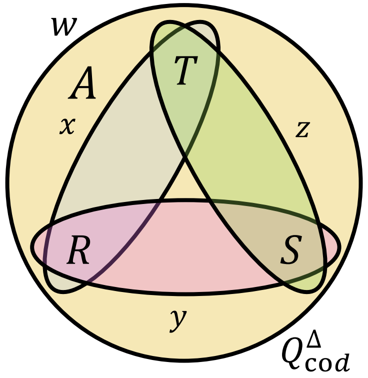


If query has a "*co-deactivated triad*"

→ NP-Complete

→ Proof: Reduction from Vertex Cover

Co-deactivated triad: 3 atoms of triad are dominated by same set of atoms

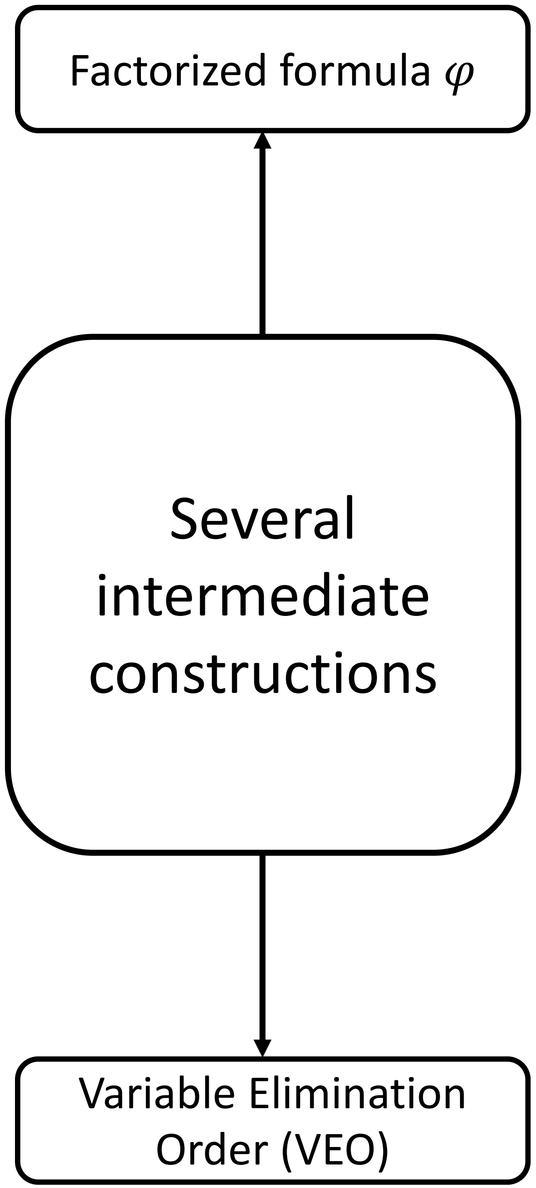


Independent Atom: $\boldsymbol{R}$ is independent if there is no atom $\boldsymbol{S}$ s.t. $\text{var}(\mathbf{S}) \subset \text{var}(\mathbf{R})$

Independent Path: A path from $\boldsymbol{R}$ to $\boldsymbol{S}$ using no variable in $\boldsymbol{T}$ is independent of T

C1: Minimal Factorizations ↔ Query Plans

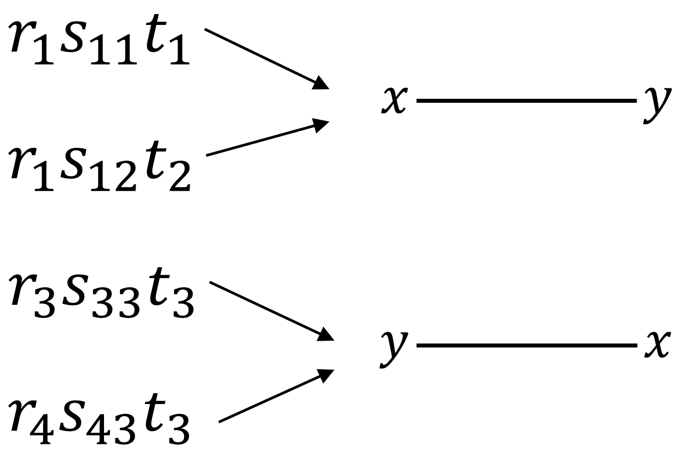
`Q():-R(x),S(x,y),T(y)`



$$r_1(s_{11}t_1 + s_{12}t_2) + \ t_3(r_3s_{33} + r_4s_{43})$$

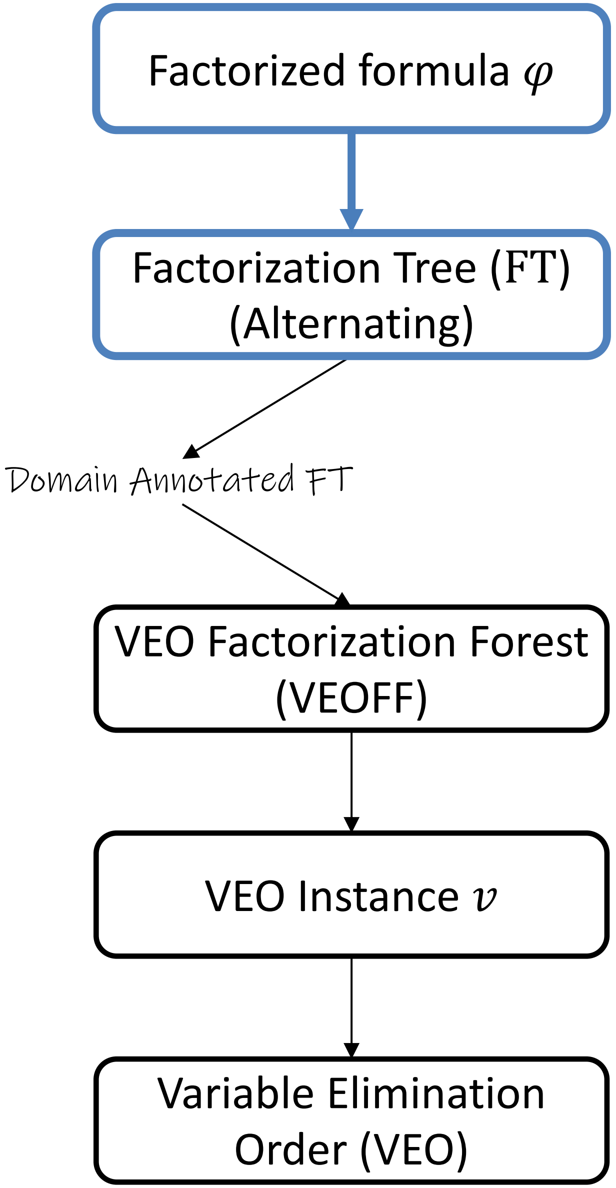
Theorem. *(informal)*

The minimal factorization for CQ provenance can always be recovered an assignment of Variable Orders

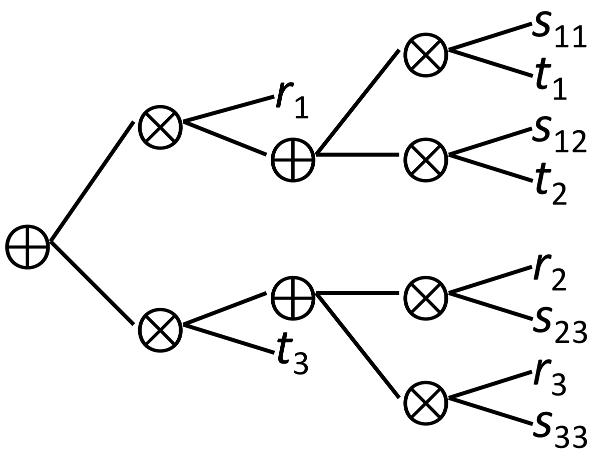


C1: Minimal Factorizations ↔ Query Plans

```
Q():-R(x),S(x,y),T(y)
```

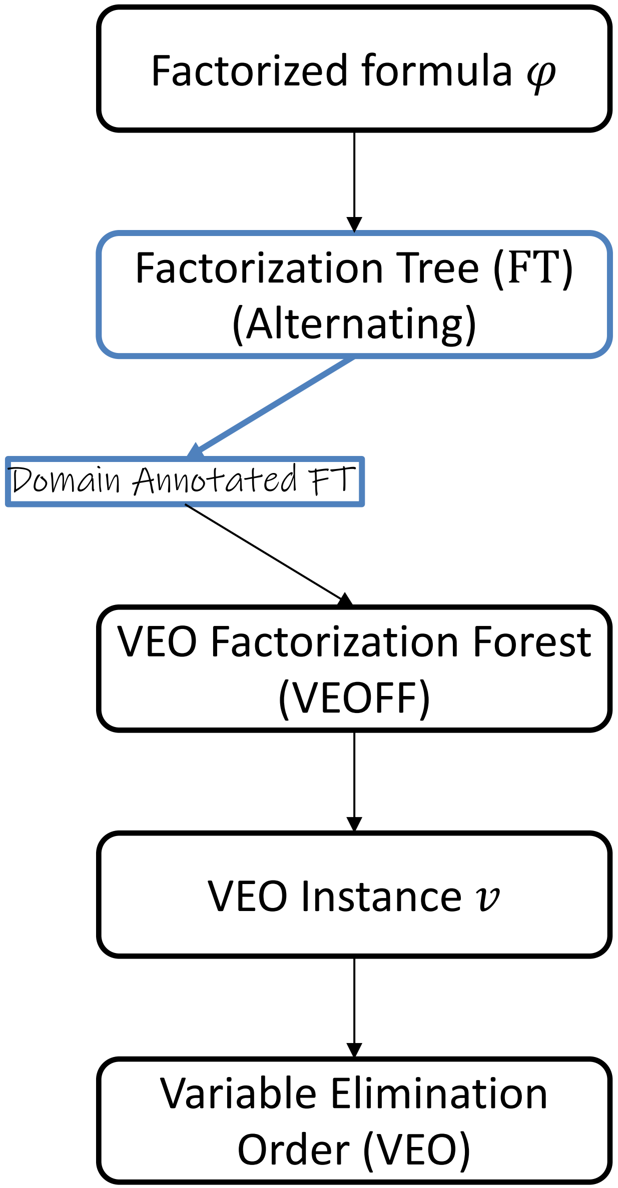


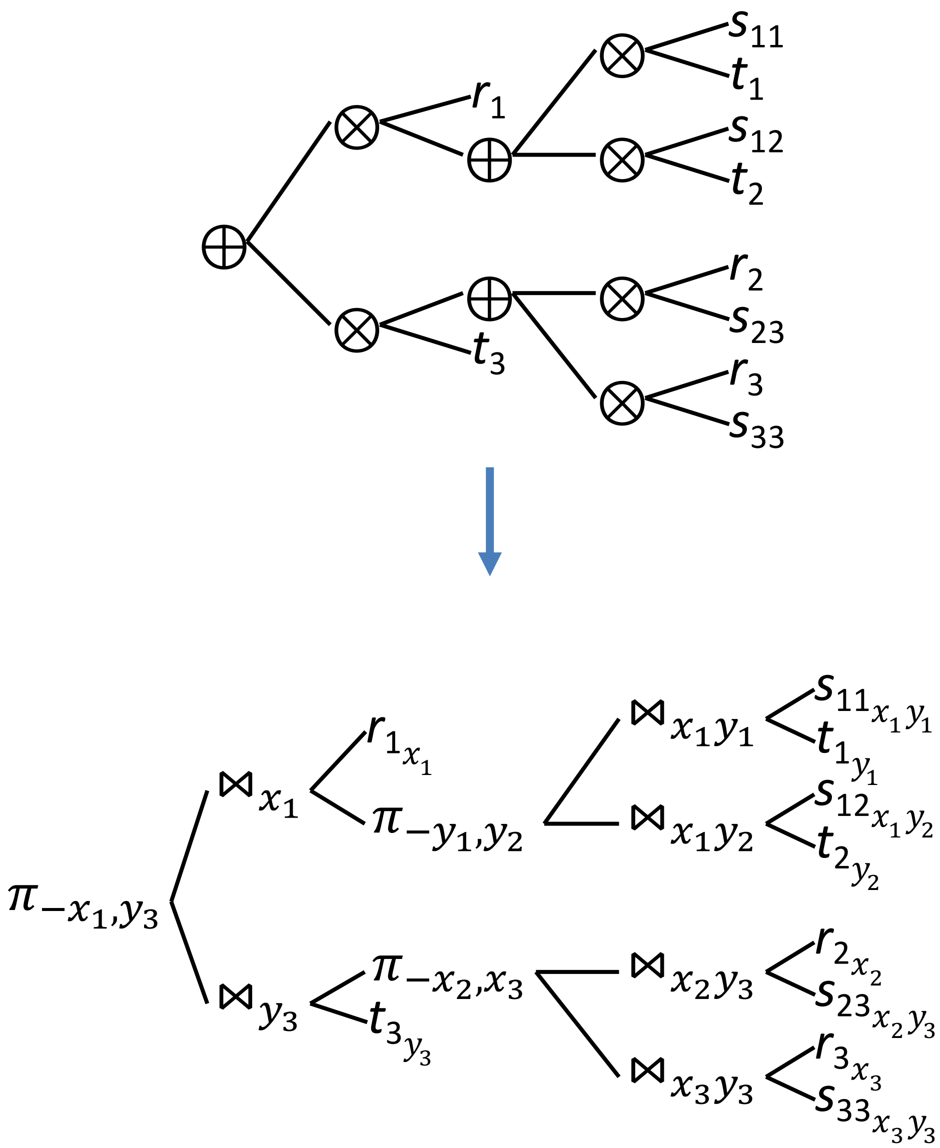
$$r_1(s_{11}t_1 + s_{12}t_2) + t_3(r_3s_{33} + r_4s_{43})$$



C1: Minimal Factorizations ↔ Query Plans

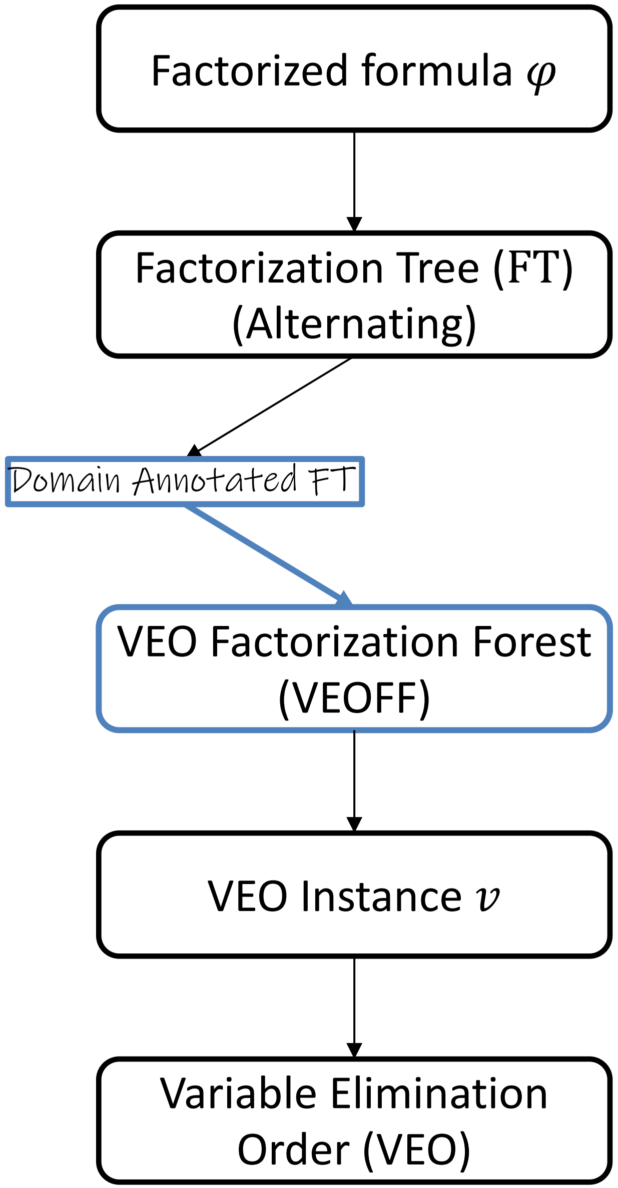
`Q():-R(x),S(x,y),T(y)`

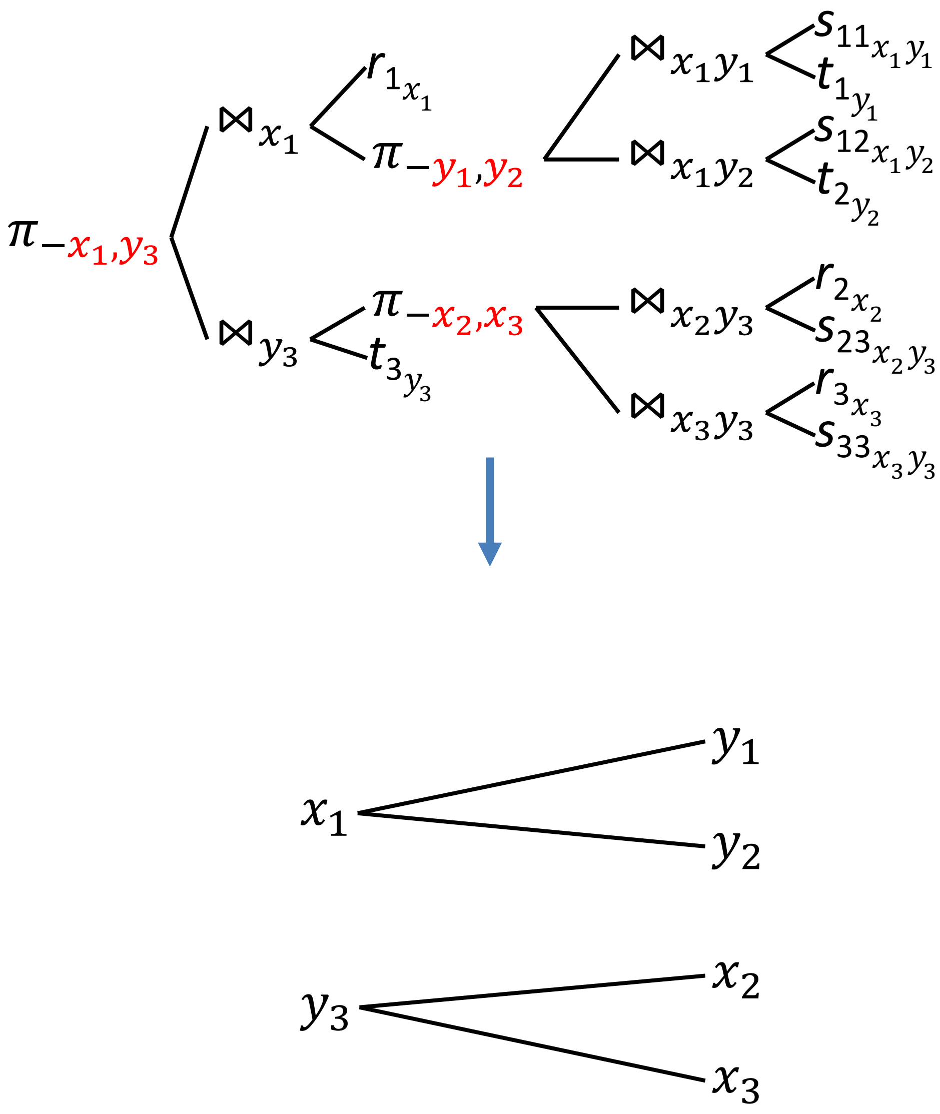




C1: Minimal Factorizations ↔ Query Plans

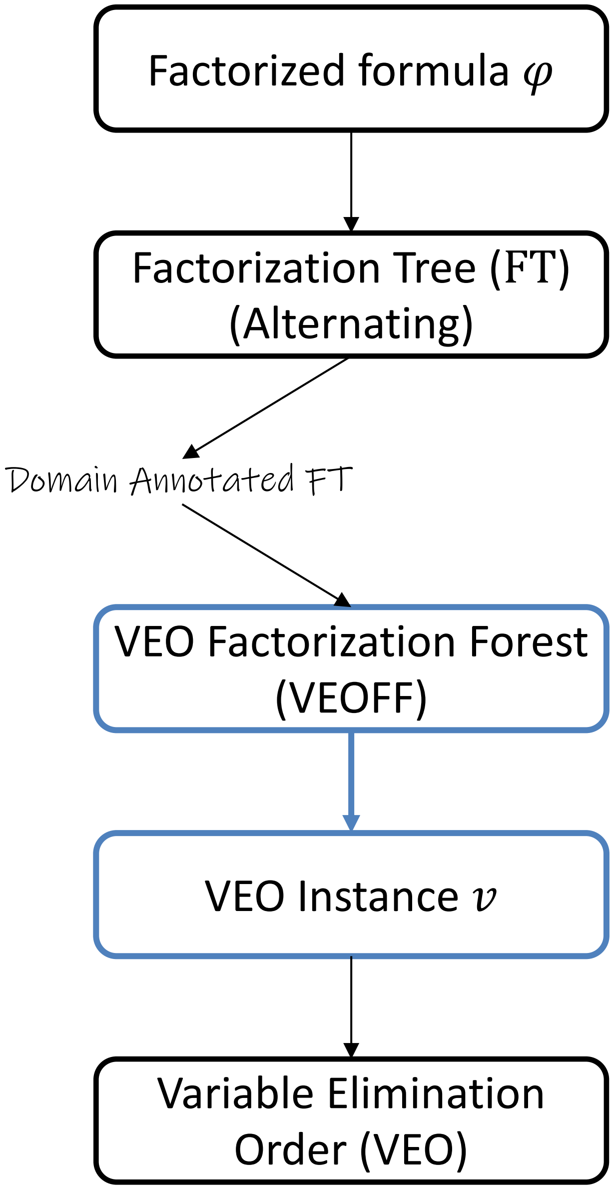
`Q():-R(x),S(x,y),T(y)`

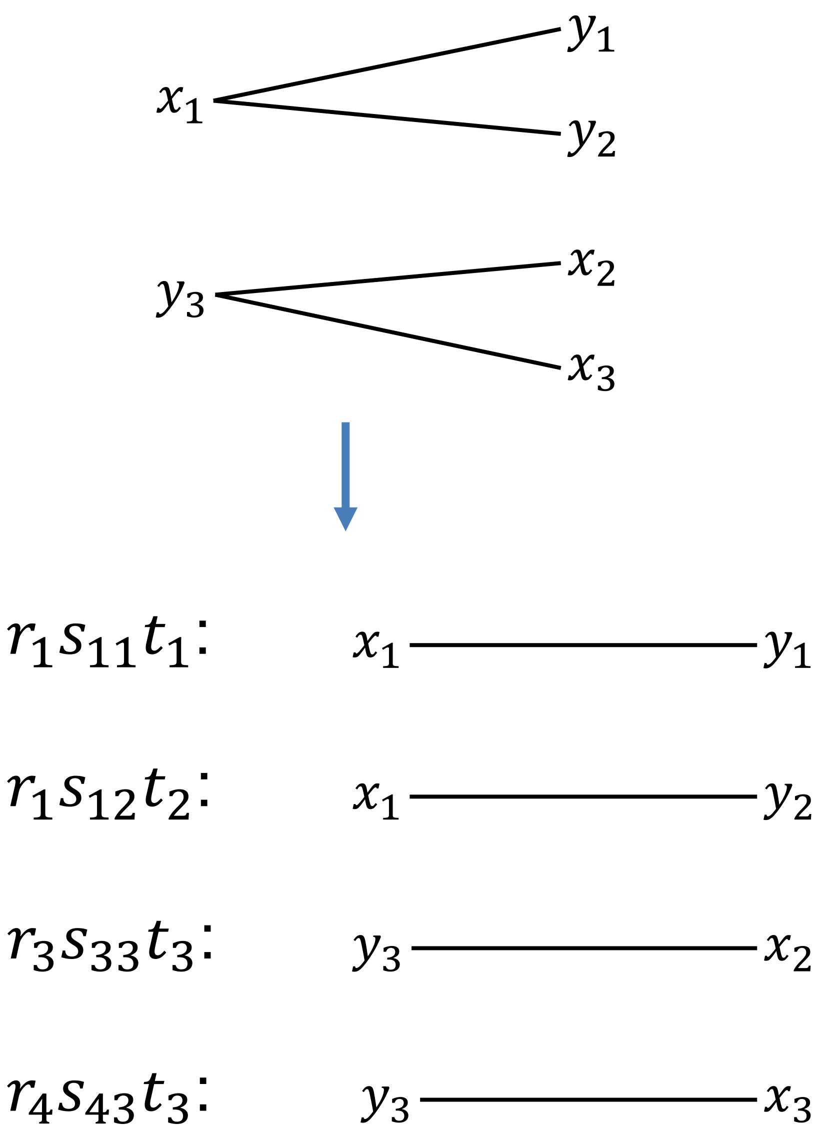




C1: Minimal Factorizations ↔ Query Plans

`Q():-R(x),S(x,y),T(y)`





C1: Minimal Factorizations ↔ Query Plans

Q():-R(x),S(x,y),T(y)