Path Representations

Wim Martens
University of Bayreuth

LDBC Meeting
@
SIGMOD/PODS'22

Work in progress (paper almost submission ready) with
Matthias Niewerth, Tina Popp, Stijn Vansummeren, Domagoj Vrgoč, Matthias Hofer

Some Challenges in Graph Queries

Some Challenges in Graph Queries

1. The exponential output challenge

Some Challenges in Graph Queries

1. The exponential output challenge

Consider a query like

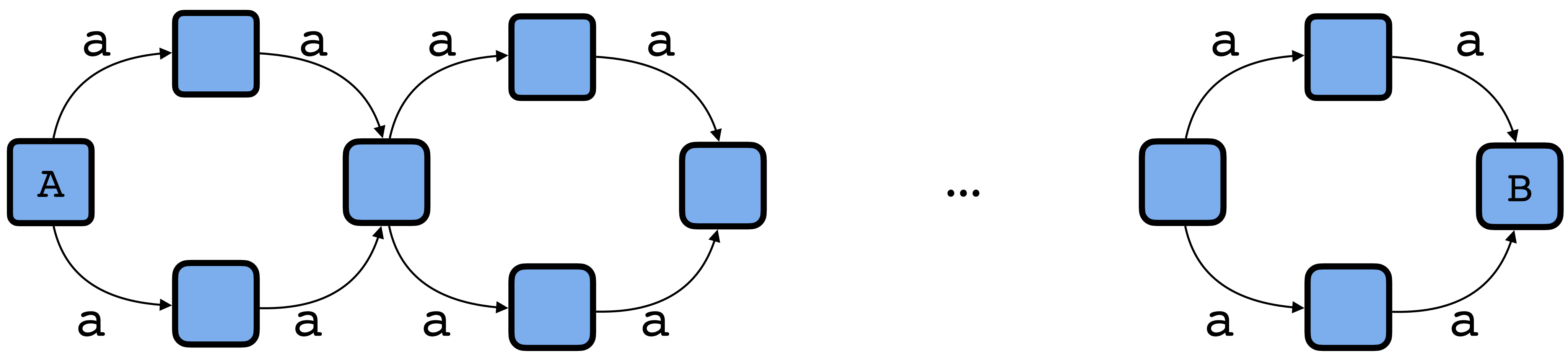
```
MATCH SHORTEST  p = (x:A)-[:a+]->(y:B)
RETURN x, y, p
```

Some Challenges in Graph Queries

1. The exponential output challenge

Consider a query like

```
MATCH SHORTEST  p = (x:A)-[:a+]->(y:B)
RETURN x, y, p
```

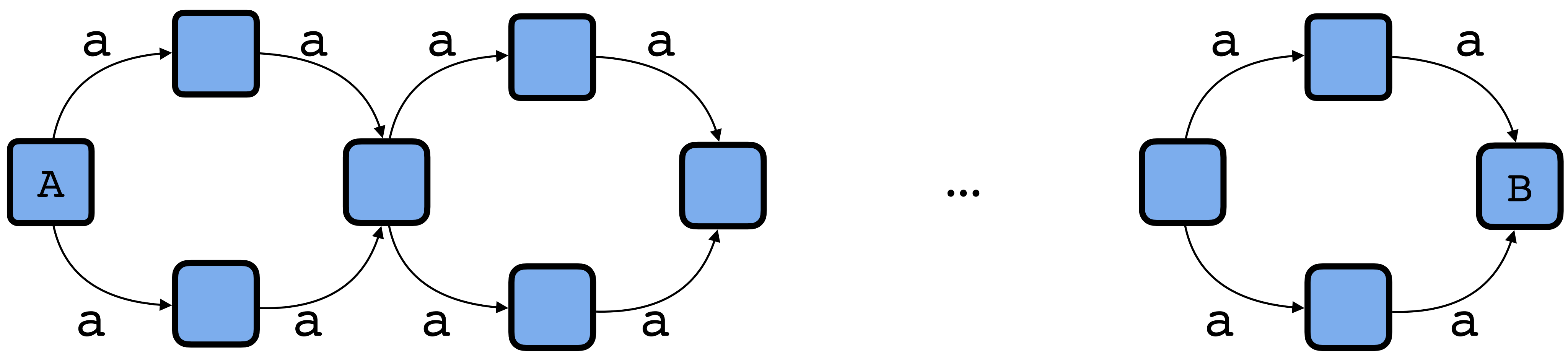


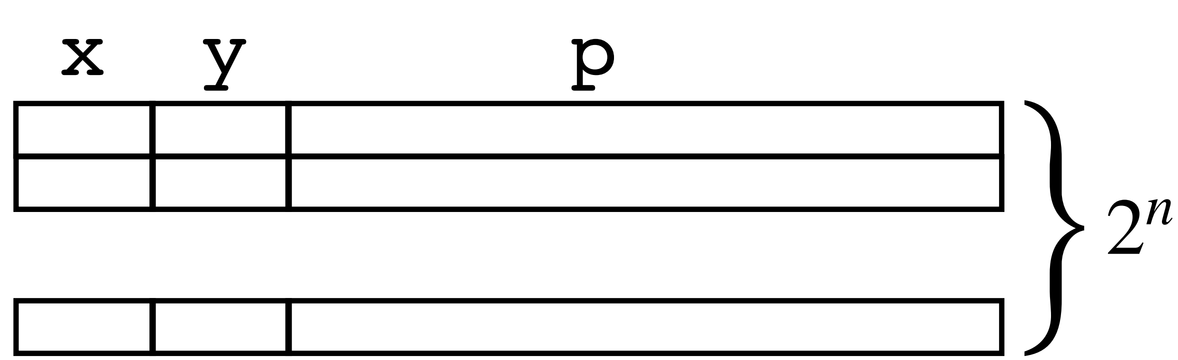
Some Challenges in Graph Queries

1. The exponential output challenge

Consider a query like

```
MATCH SHORTEST  p = (x:A)-[:a+]->(y:B)
RETURN x, y, p
```





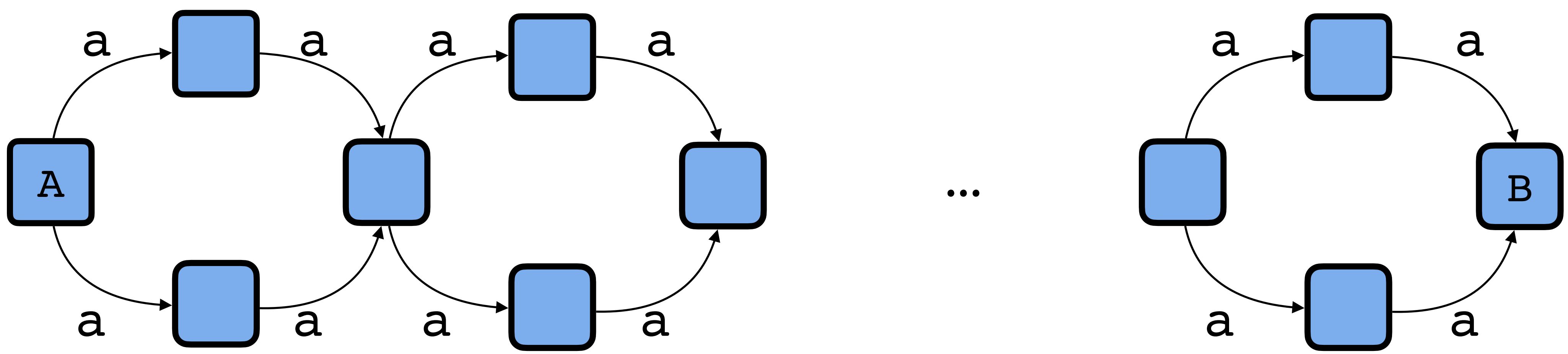
Returns 2^n many paths on a graph with $O(n)$ nodes and edges

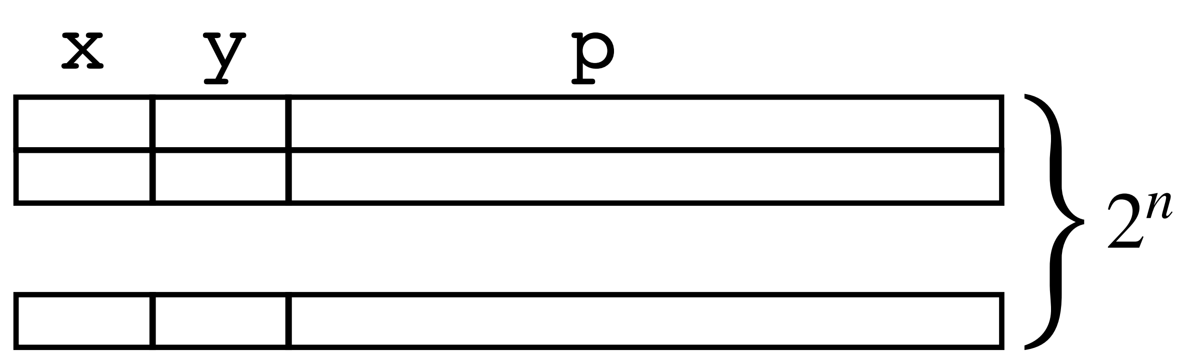
Some Challenges in Graph Queries

1. The exponential output challenge

Consider a query like

```
MATCH SHORTEST  p = (x:A)-[:a+]->(y:B)
RETURN x, y, p
```





Returns 2^n many paths on a graph with $O(n)$ nodes and edges

(This is a lot more than the endpoint pairs from SPARQL and academic research)

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table}$$

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{?}$$

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{?}$$

vs

$$\text{Graph} \xrightarrow{\text{query}} \text{Graph} \xrightarrow{\text{query}}$$

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{?}$$

vs

$$\text{Graph} \xrightarrow{\text{query}} \text{Graph} \xrightarrow{\text{query}}$$

Challenge exists on two levels

- representing the output of entire queries
- representing intermediate results in query plans

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

Some Challenges in Graph Queries

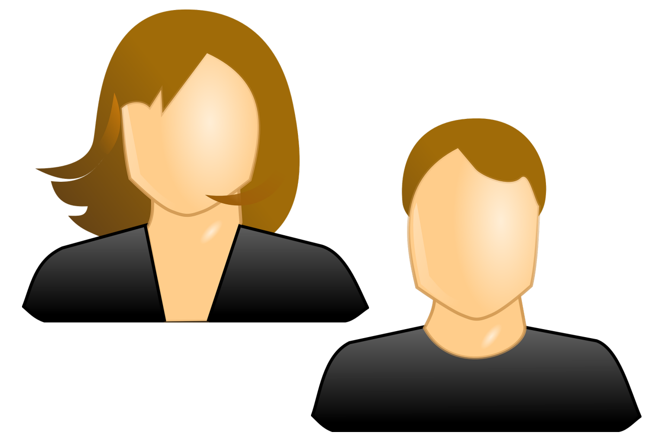
1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table}$$

Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

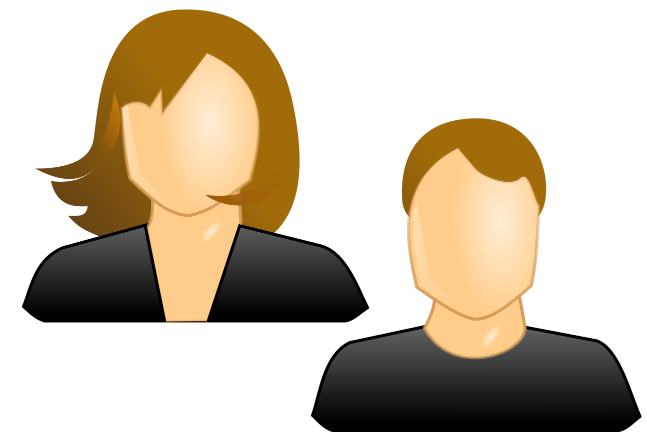
$$\text{Graph} \xrightarrow{\text{query}} \text{Table}$$



Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

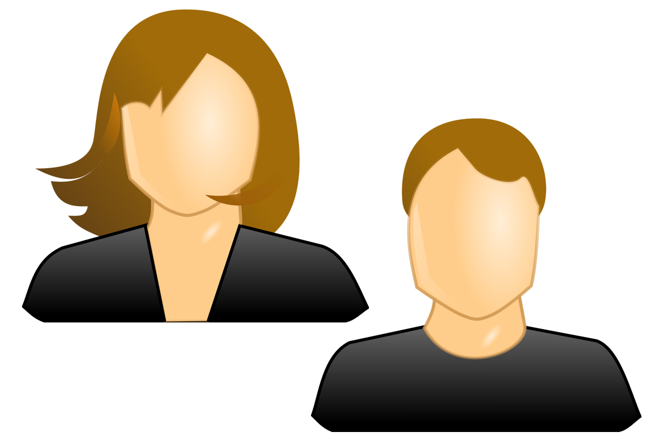
$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{\text{make digestible}} ??$$



Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{\text{make digestible}} \text{??}$$

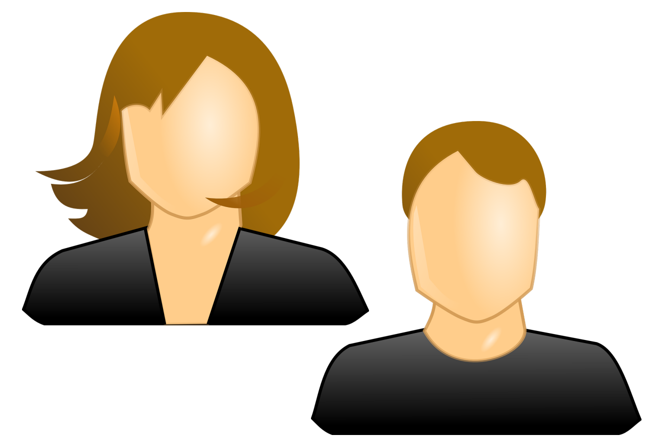


- table can be extremely large and complex
- graph projections look nice but are not lossless

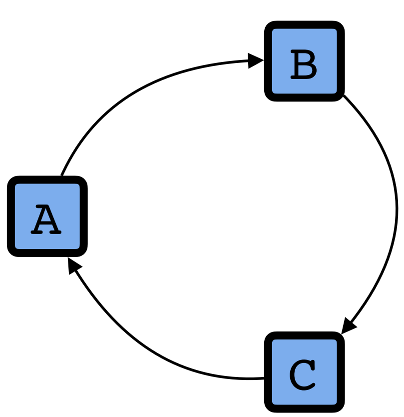
Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

$$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{\text{make digestible}} \text{??}$$



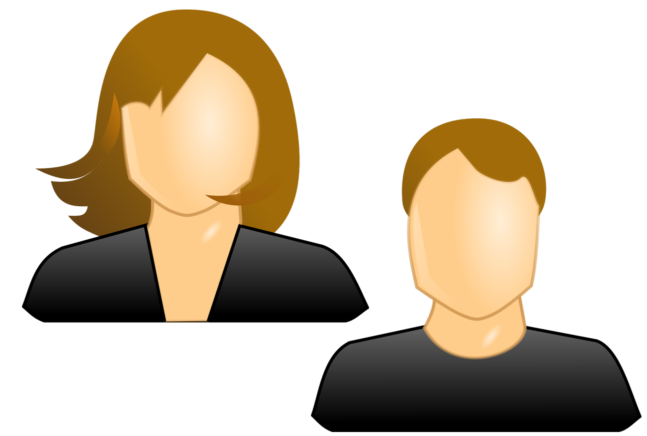
- table can be extremely large and complex
- graph projections look nice but are not lossless



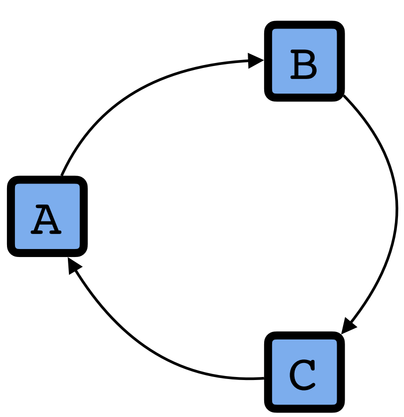
Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{\text{make digestible}} \text{??}$



- table can be extremely large and complex
- graph projections look nice but are not lossless

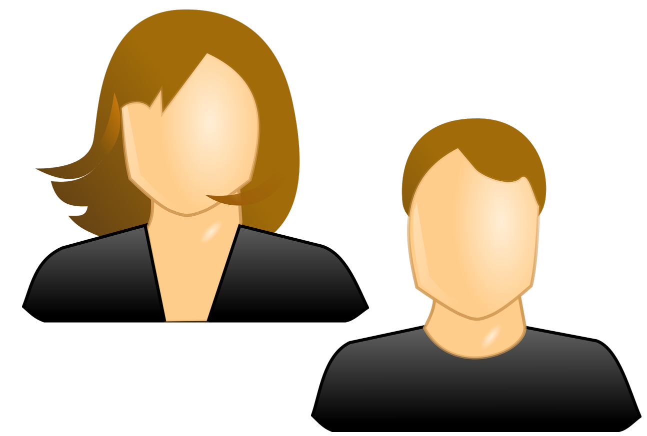


shortest
even length
path
from A
to A

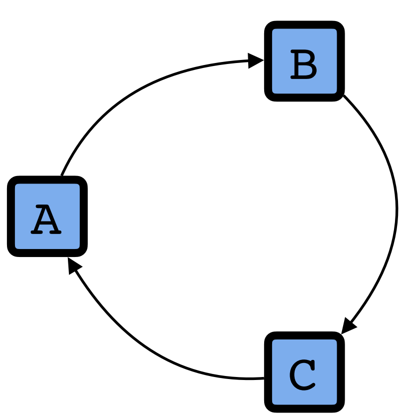
Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

Graph $\xrightarrow{\text{query}}$ Table $\xrightarrow{\text{make digestible}}$??



- table can be extremely large and complex
- graph projections look nice but are not lossless

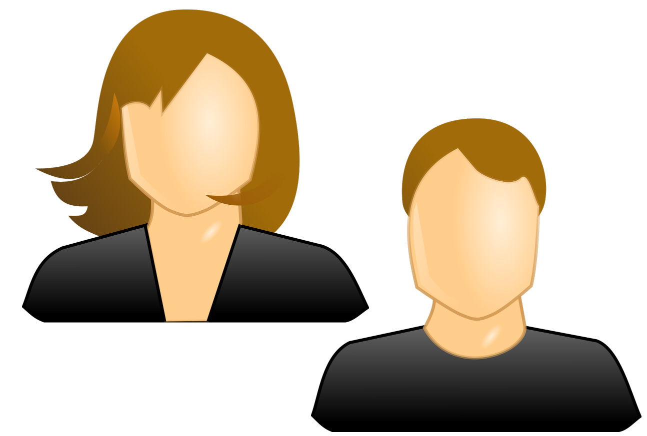


shortest even length path from A to A $\longrightarrow$ Table

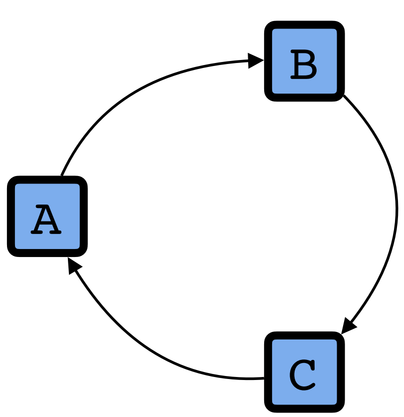
Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

Graph $\xrightarrow{\text{query}}$ Table $\xrightarrow{\text{make digestible}}$??



- table can be extremely large and complex
- graph projections look nice but are not lossless

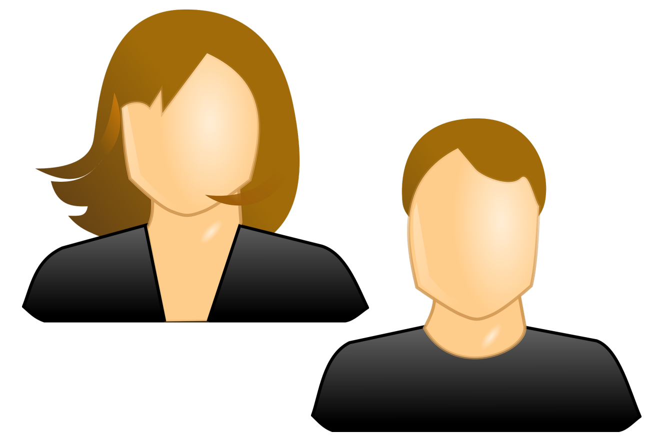


$\xrightarrow{\text{shortest even length path from A to A}}$ Table $\xrightarrow{\text{graph projection}}$

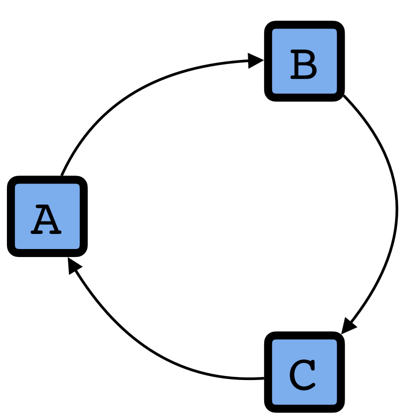
Some Challenges in Graph Queries

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

$\text{Graph} \xrightarrow{\text{query}} \text{Table} \xrightarrow{\text{make digestible}} \text{??}$



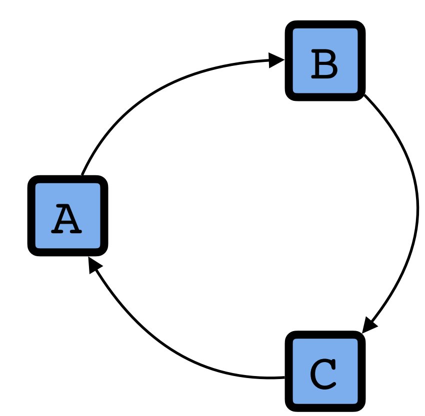
- table can be extremely large and complex
- graph projections look nice but are not lossless



shortest even length path from A to A

Table

graph projection



What Do We Want to Do?

What Do We Want to Do?

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

Present an idea that may help here

What Do We Want to Do?

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

Present an idea that may help here

- Focus on 1. and 2.
- We've done a lot of thinking but it's still work in progress
  - First paper is close to ready
- I think it's very promising
  - We'll definitely keep working on it

What Do We Want to Do?

Store intermediate results of queries as graphs

- Can be exponentially more succinct than the table
- Never larger than the table
- Without losing information (as opposed to graph projection)

What Do We Want to Do?

Store intermediate results of queries as graphs

- Can be exponentially more succinct than the table
- Never larger than the table
- Without losing information (as opposed to graph projection)

Main idea:

What Do We Want to Do?

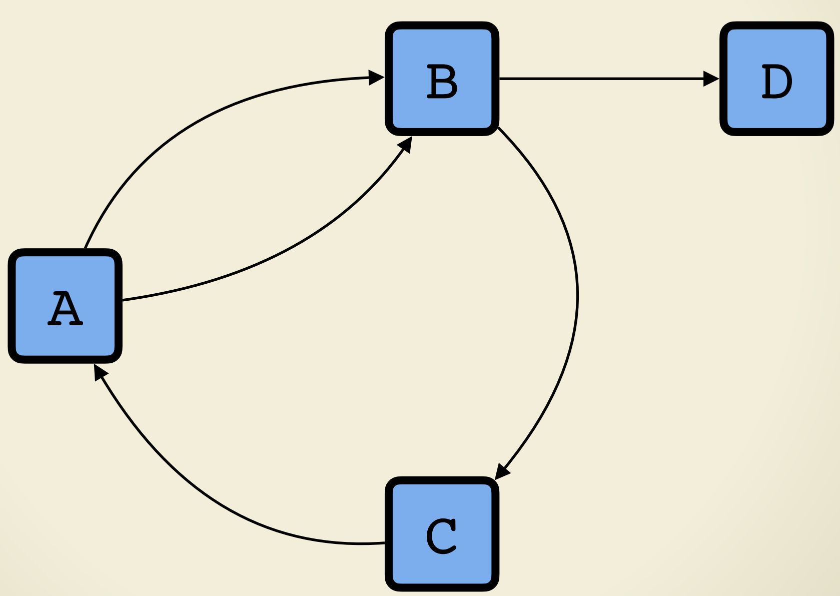
Store intermediate results of queries as graphs

- Can be exponentially more succinct than the table
- Never larger than the table
- Without losing information (as opposed to graph projection)

Main idea:

Query + Graph

```
p = (x:A)-[:a+]->(y:B)
```



What Do We Want to Do?

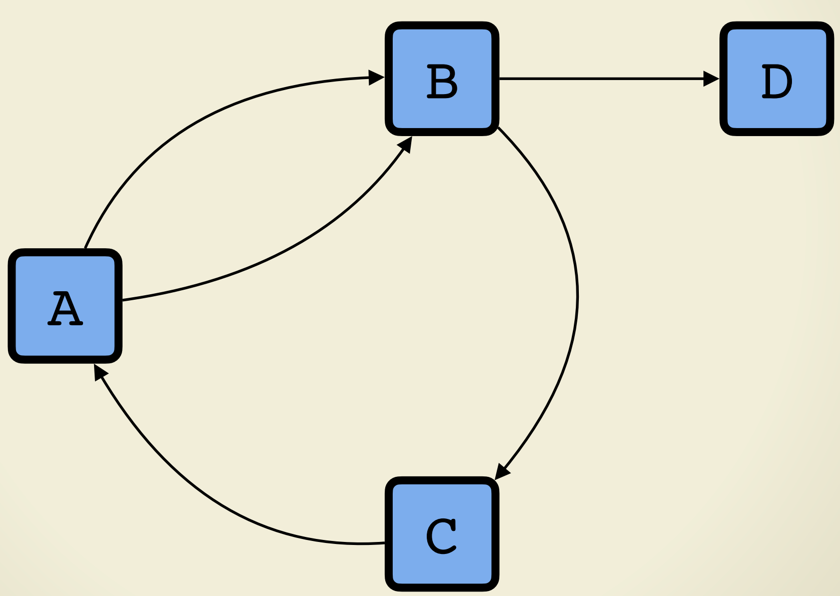
Store intermediate results of queries as graphs

- Can be exponentially more succinct than the table
- Never larger than the table
- Without losing information (as opposed to graph projection)

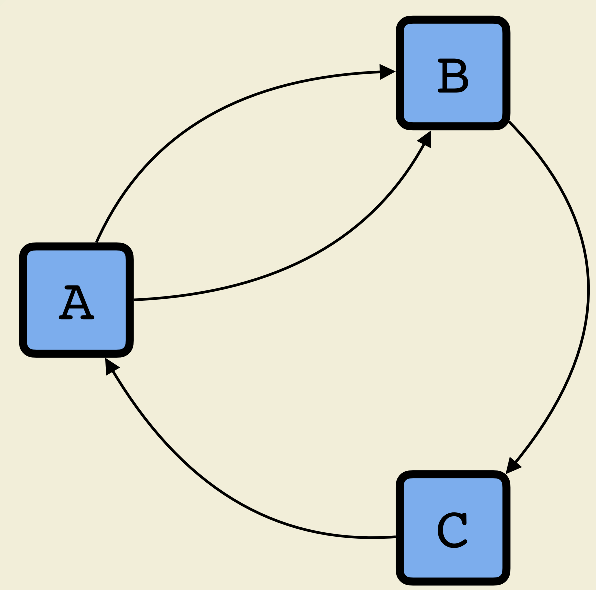
Main idea:

Query + Graph

```
p = (x:A)-[:a+]->(y:B)
```



Representation of Paths in Output



"All paths from A to B in this graph"

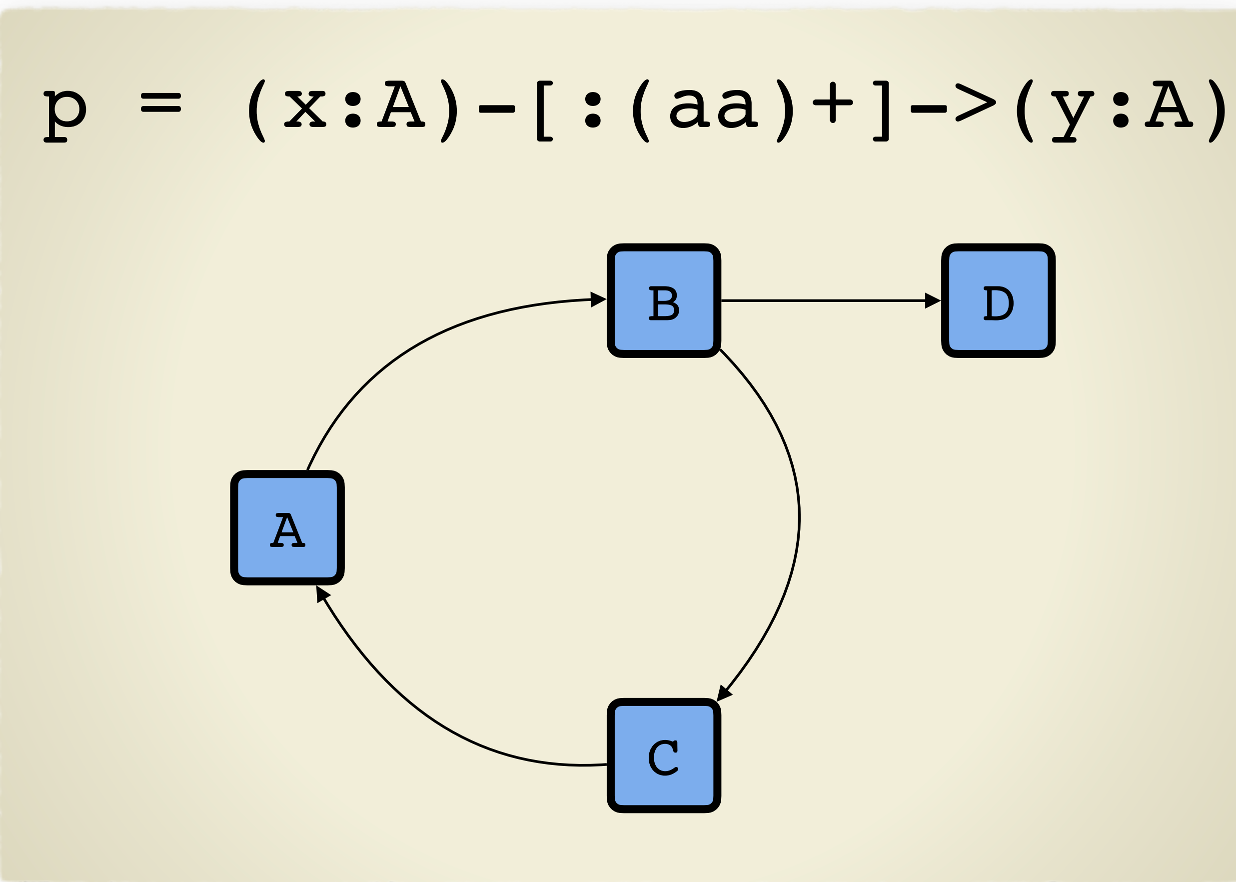
What Do We Want to Do?

Store intermediate results of queries as graphs

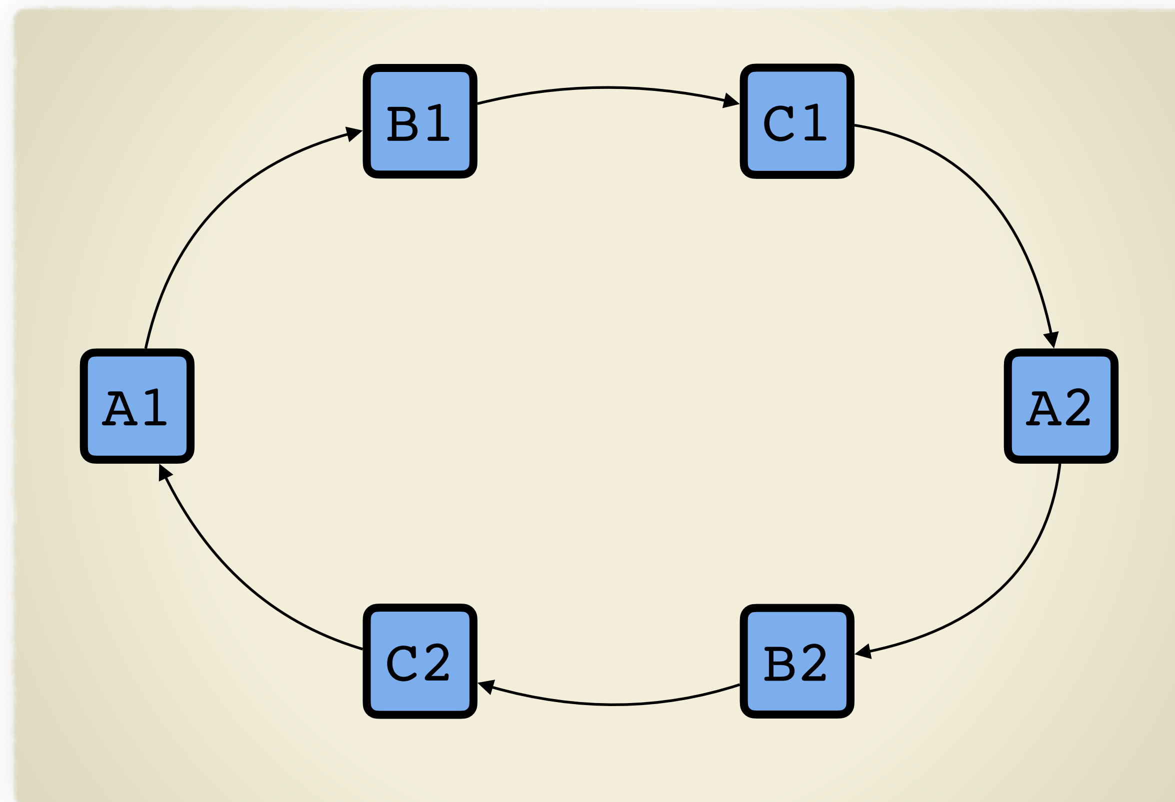
- Can be exponentially more succinct than the table
- Never larger than the table
- Without losing information (as opposed to graph projection)

Main idea:

Query + Graph



Representation of Paths in Output



"All paths from A1 to A1 in this graph"

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \to (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \to (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

if e connects u to v in R,
then $\gamma(e)$ should connect $\gamma(u)$ to $\gamma(v)$ in G

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \rightarrow (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \rightarrow (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

if e connects u to v in R,
then $\gamma(e)$ should connect $\gamma(u)$ to $\gamma(v)$ in G

start nodes

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \to (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

if e connects u to v in R,
then $\gamma(e)$ should connect $\gamma(u)$ to $\gamma(v)$ in G

target nodes

start nodes

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \to (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

start nodes

target nodes

if e connects u to v in R,
then $\gamma(e)$ should connect $\gamma(u)$ to $\gamma(v)$ in G

R represents
"all paths from some node in S to some node in T"

Path Representations

Let $G = (N_G, E_G, \eta, \lambda)$ be a graph, where

- $\eta : E_G \to (N_G \times N_G)$ maps edge ids to pairs of node ids
- λ maps each edge to a label

Definition

A path representation over graph G is a tuple

$$R = (N, E, \eta, \gamma, S, T),$$

where

- (N, E, η) is an unlabeled graph
- $\gamma : (N \cup E) \to (N_G \cup E_G)$ is a total homomorphism
- $S \subseteq N$ and $T \subseteq N$

if e connects u to v in R,
then $\gamma(e)$ should connect $\gamma(u)$ to $\gamma(v)$ in G

target nodes

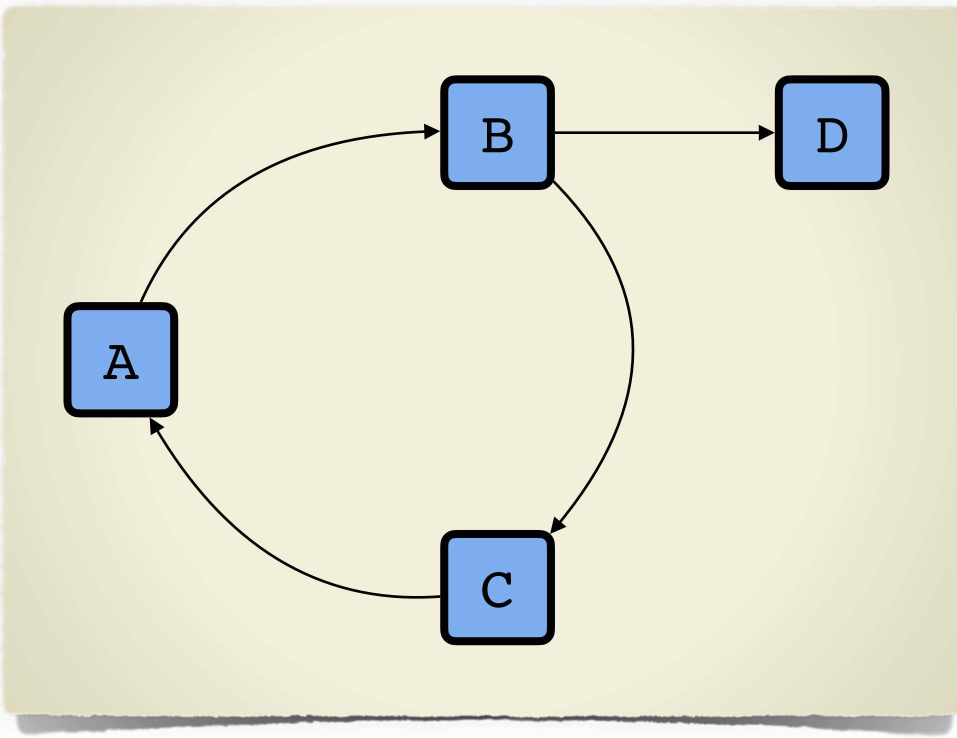
start nodes

R represents
"all paths from some node in S to some node in T"

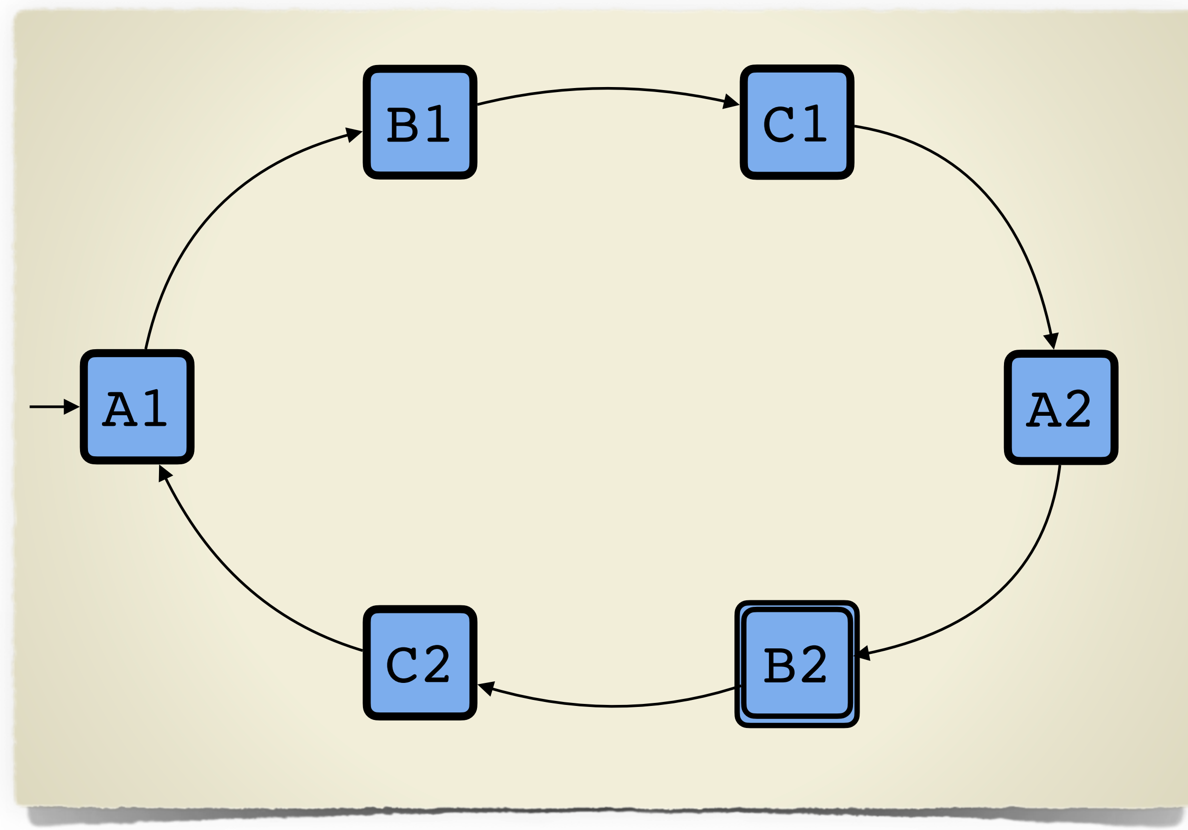
This is a lossless representation
of a set or multiset of paths in G

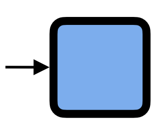
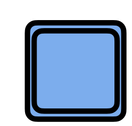
Path Representations: Examples

The set of even length paths from A to B in



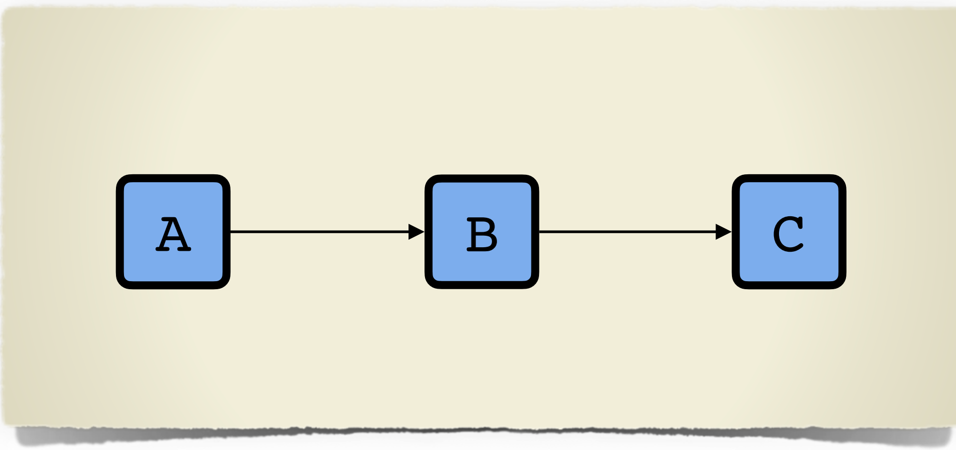
Path Representation



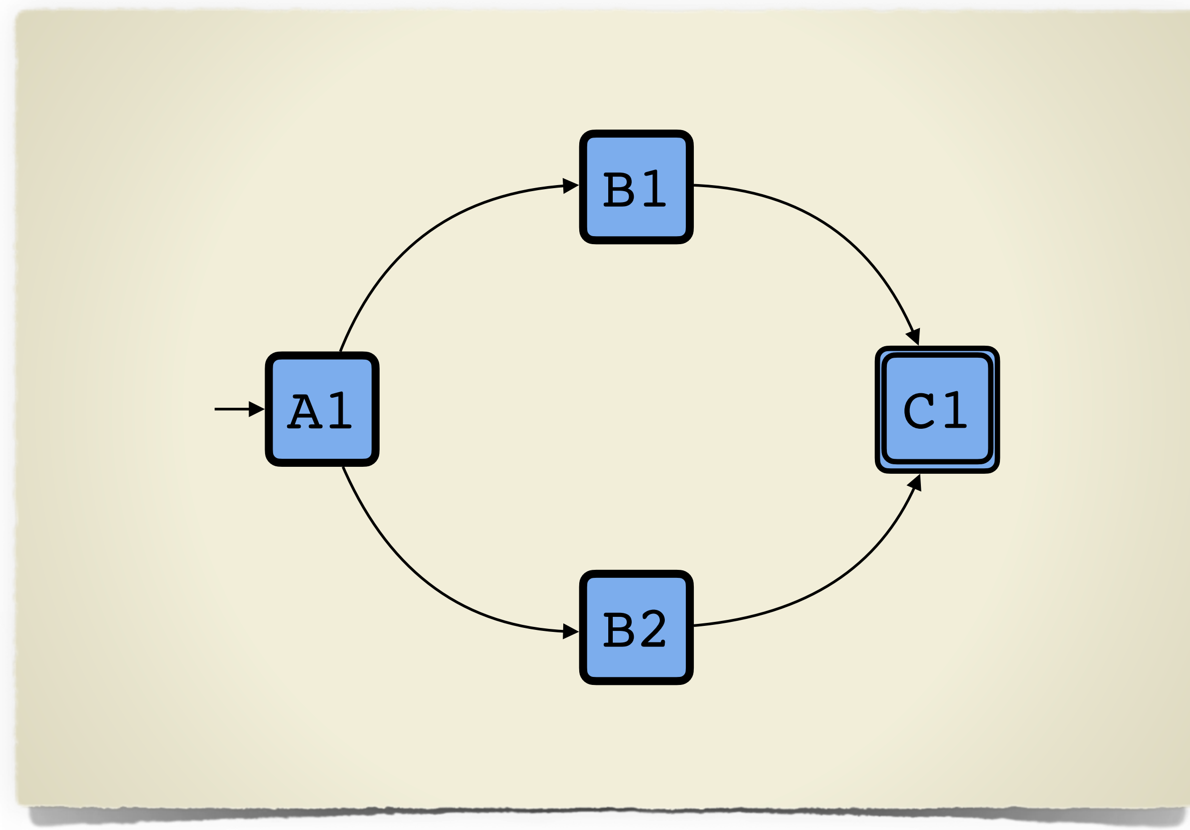
- Each Ai is mapped to A, etc.
- Start nodes: 
- Target nodes: 

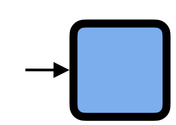
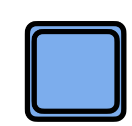
Path Representations: Examples

The path from A to C twice



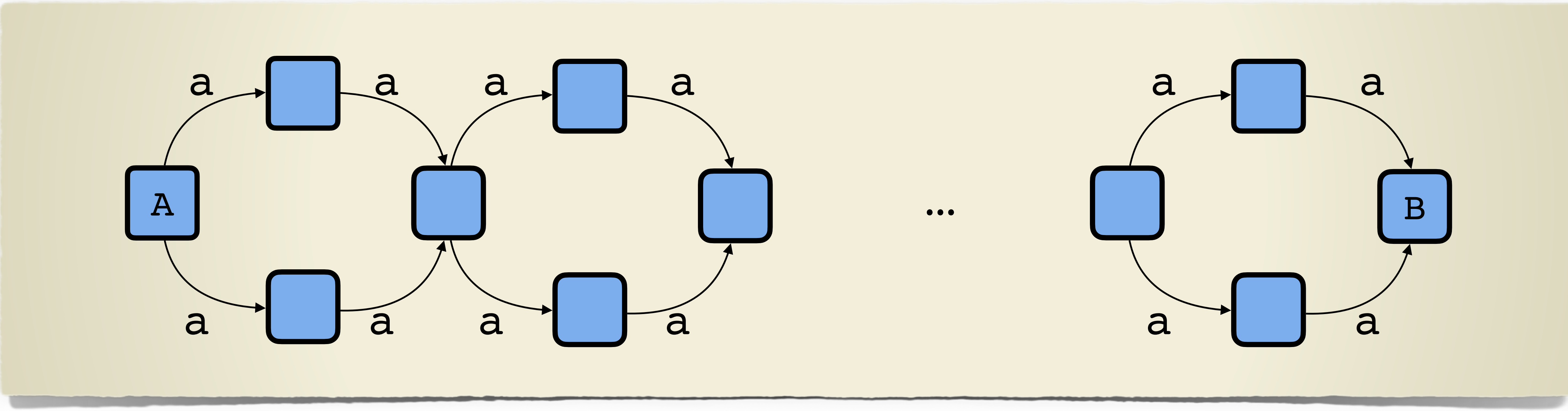
Path Representation



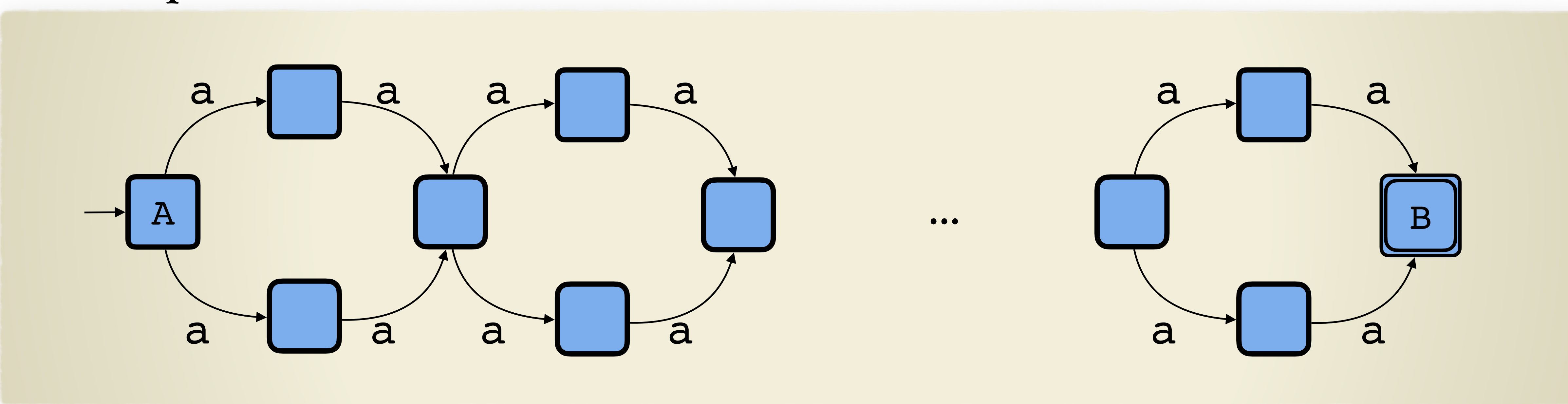
- Each Ai is mapped to A, etc.
- Start nodes: 
- Target nodes: 

Path Representations: Examples

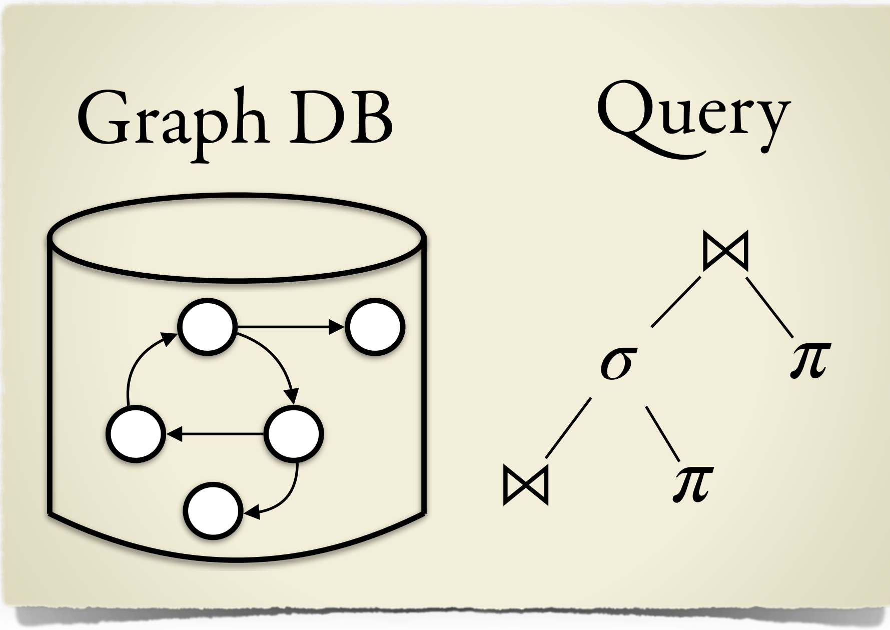
The 2^n paths from A to B



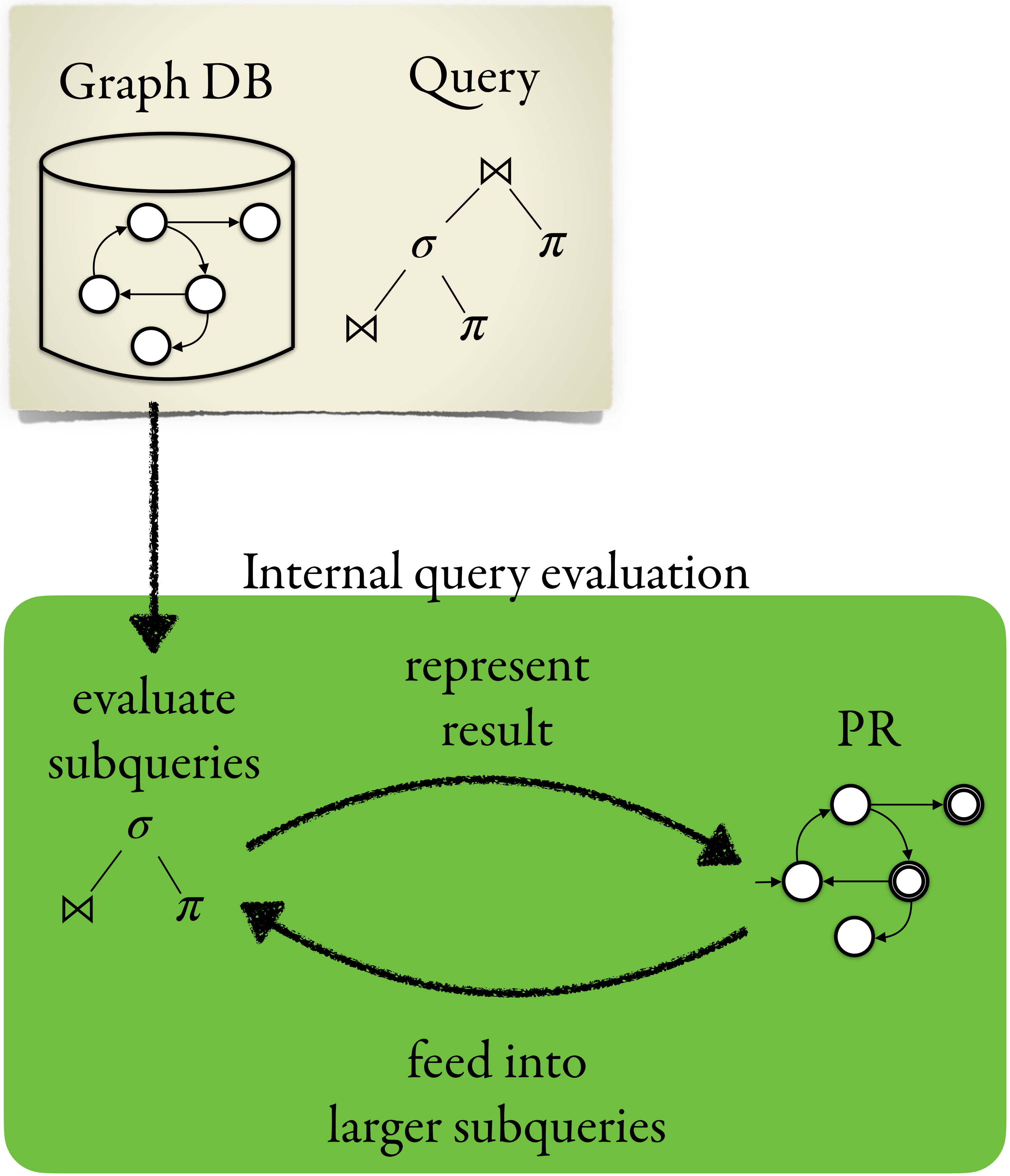
Path Representation



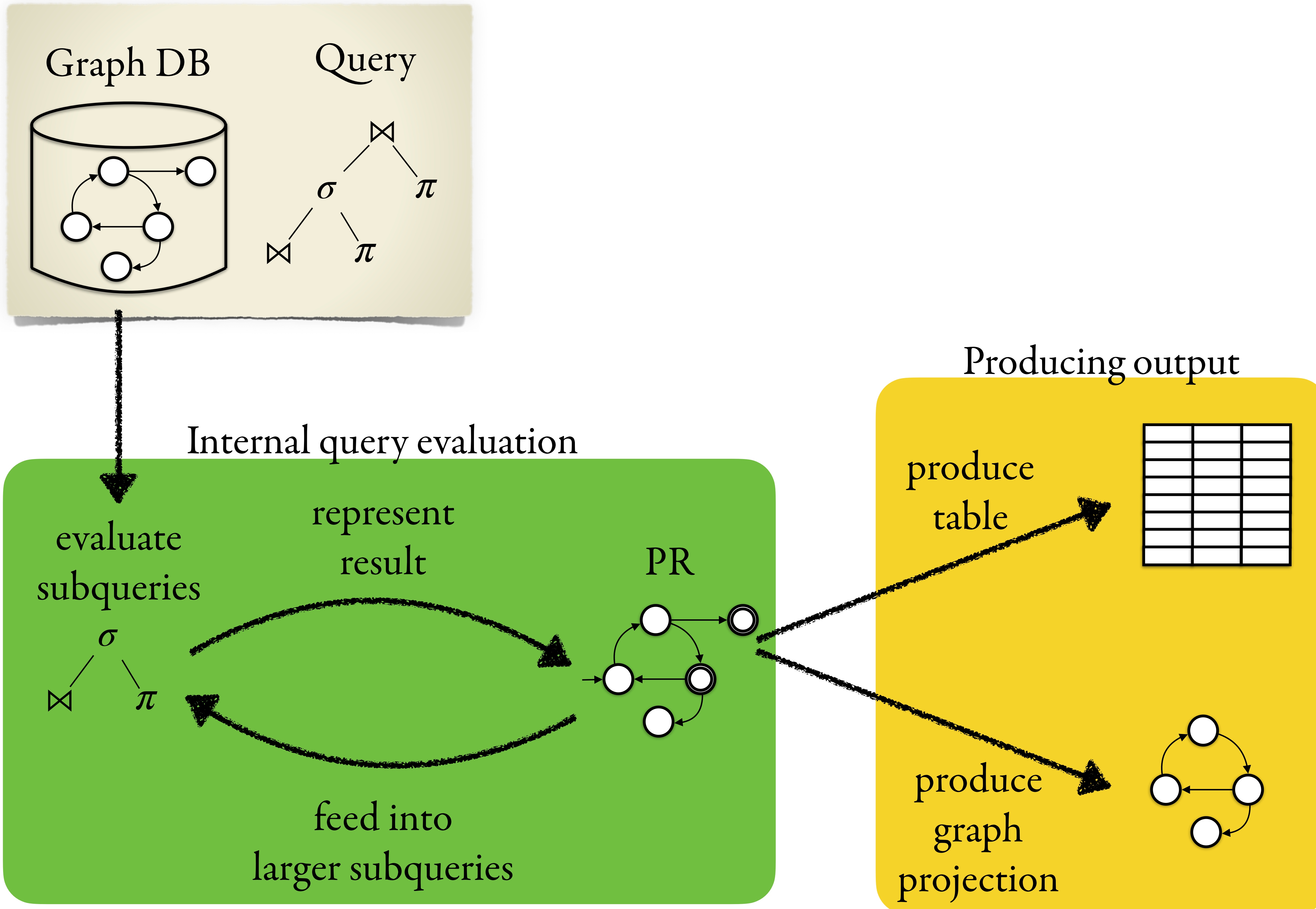
Path Representations: Envisioned Use



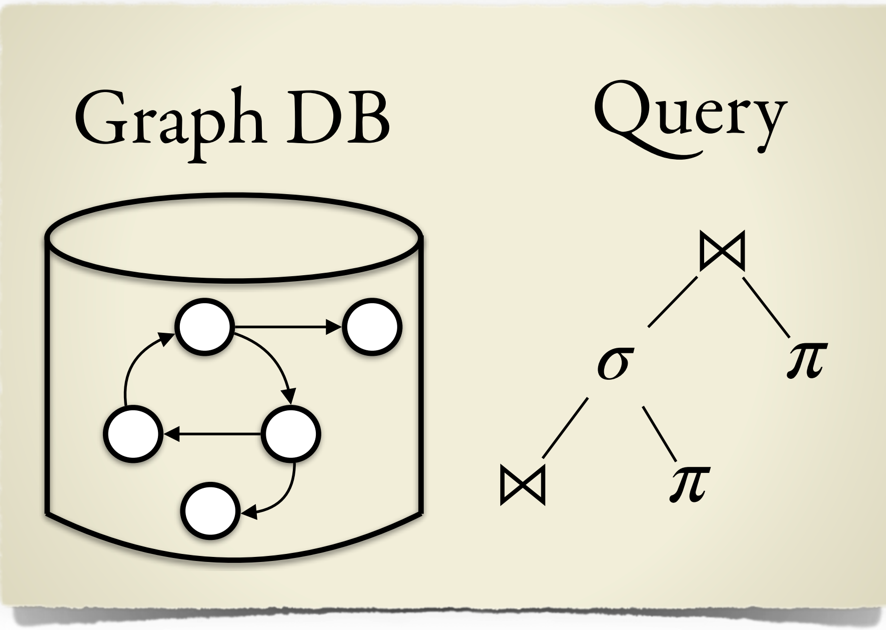
Path Representations: Envisioned Use

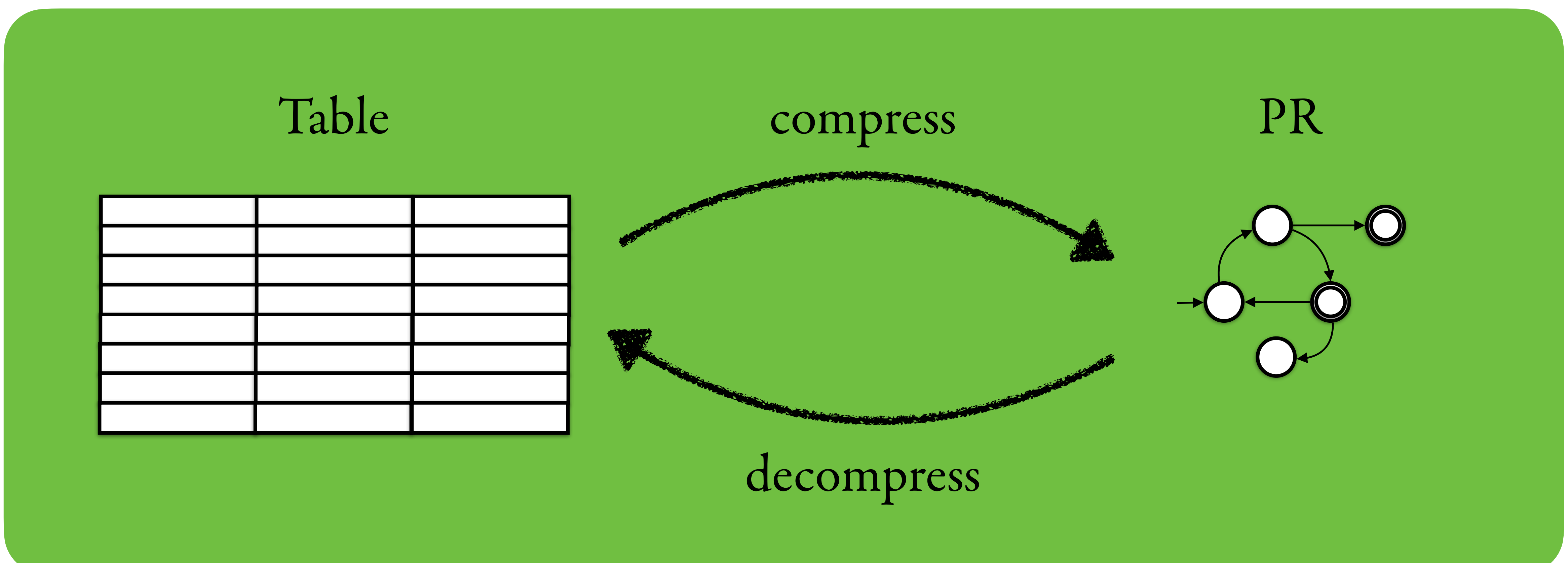


Path Representations: Envisioned Use

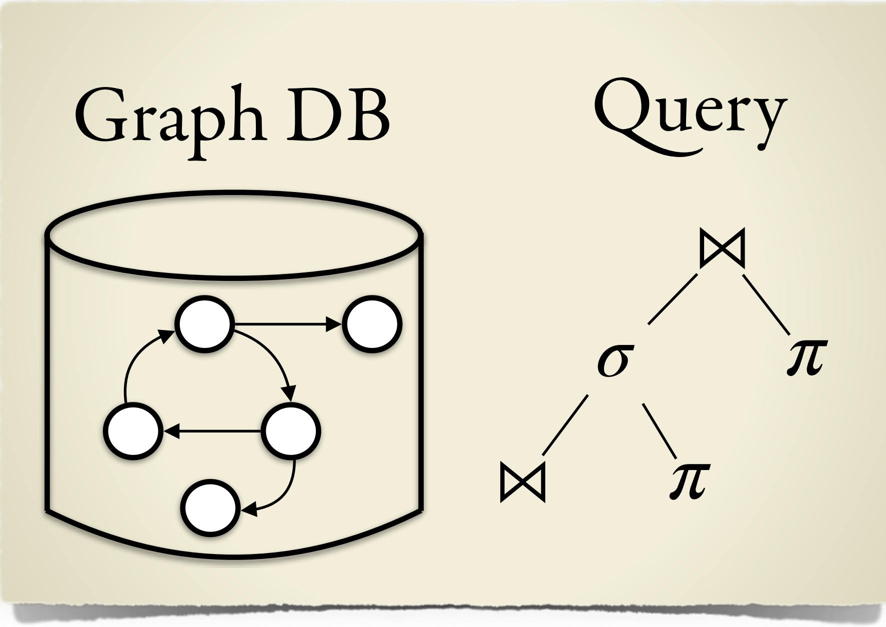


Path Representations: Envisioned Use



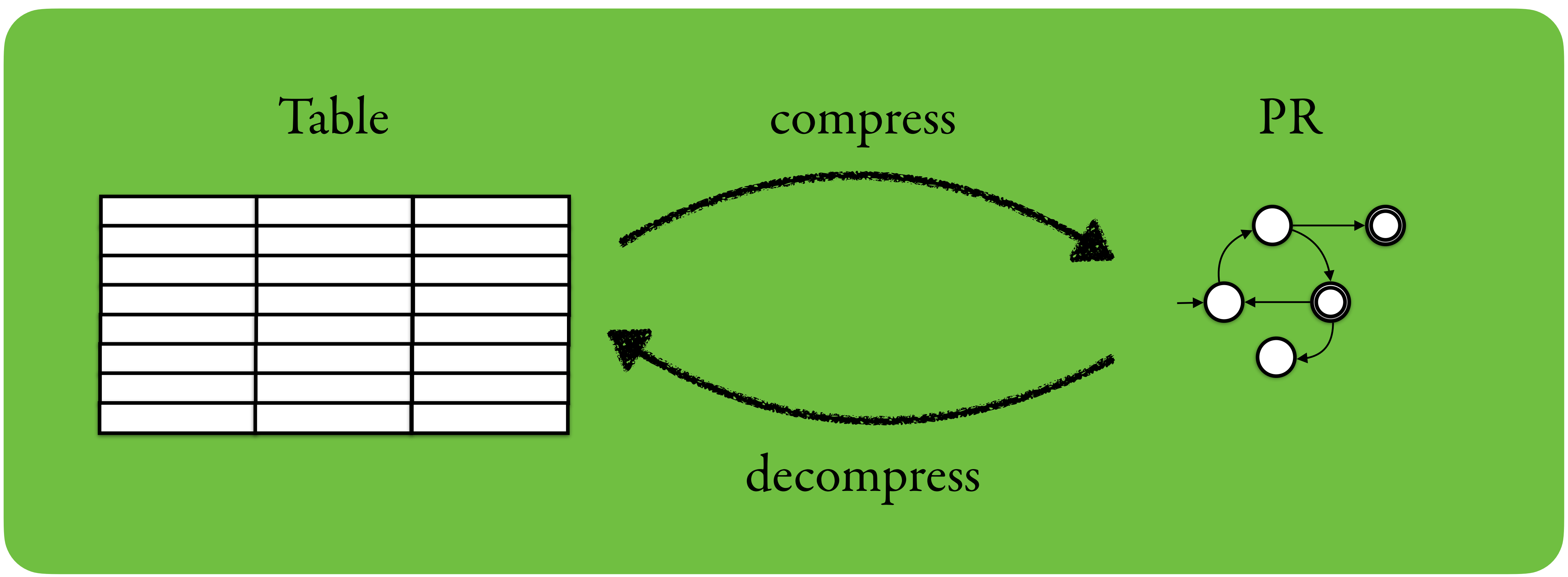


Path Representations: Envisioned Use



What we investigate(d)

Size of representation
Losslessness / Expressivity
Complexity of computing a PR
Complexity of applying upstream operators
Complexity of producing output



Path Representations: Properties

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in *G*
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in *G*
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in *G*
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)
  - Doing multiset minimization is NP-complete

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)
  - Doing multiset minimization is NP-complete
    - (But remember that it is exponentially succinct)

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)
  - Doing multiset minimization is NP-complete
    - (But remember that it is exponentially succinct)

So I'm wondering...

Could PRs be a viable option
for representing (the paths in)
intermediate results for graph queries?

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in G
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)
  - Doing multiset minimization is NP-complete
    - (But remember that it is exponentially succinct)

So I'm wondering...

Could PRs be a viable option
for representing (the paths in)
intermediate results for graph queries?

Helps the exponential output challenge

Path Representations: Properties

Representing Multisets of Paths

- PRs can represent any finite multiset of paths in *G*
  - This is never larger than the table representing this multiset
  - But can be exponentially smaller
- You can compute the table back from the PR
  - This costs about linear time in the size of the table

Optimization

- PRs can be optimized (make representation small)
  - Testing multiset equivalence is in polynomial time (!)
  - Doing multiset minimization is NP-complete
    - (But remember that it is exponentially succinct)

So I'm wondering...

Could PRs be a viable option
for representing (the paths in)
intermediate results for graph queries?

Helps the exponential output challenge
Helps the composability challenge?

PRs for Query Evaluation

Regular Path Queries

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute

- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute

- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute

- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

In our draft paper, we study PRs for RPQs under different evaluation modes:

- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute
- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

→ even works if the output has infinitely many paths

In our draft paper, we study PRs for RPQs under different evaluation modes:
- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute
- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

→ even works if the output has infinitely many paths

In our draft paper, we study PRs for RPQs under different evaluation modes:
- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

The above complexities need to be tweaked for different evaluation modes

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute

- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

→ even works if the output has infinitely many paths

In our draft paper, we study PRs for RPQs under different evaluation modes:

- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

The above complexities need to be tweaked for different evaluation modes

all paths ✔

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute

- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

→ even works if the output has infinitely many paths

In our draft paper, we study PRs for RPQs under different evaluation modes:

- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

The above complexities need to be tweaked for different evaluation modes

all paths ✓

shortest, lexicographically shortest ⇝ similar

PRs for Query Evaluation

Regular Path Queries

Given an RPQ, we can compute
- a PR for the set of paths in its output in linear time (as opposed to exponential time for tables)
- a graph projection of the output in linear time (as opposed to exponential time for tables)

→ even works if the output has infinitely many paths

In our draft paper, we study PRs for RPQs under different evaluation modes:
- all paths
- all shortest paths
- "lexicographically shortest paths"
- simple paths
- trails

The above complexities need to be tweaked for different evaluation modes

all paths ✓

shortest, lexicographically shortest ⇝ similar

simple paths, trails ⇝ more expensive,
but PRs are still exp more succinct than tables

PRs for Query Evaluation

Regular Path Queries

From such a PR, we can

- count the number of paths in polynomial time
- uniformly sample a path of length n in polynomial time

Beyond RPQs?

Unions of Regular Path Queries

Beyond RPQs?

Unions of Regular Path Queries

- These are easy to deal with
  - Essentially, one just needs a good multiset semantics for PRs to deal with unions
  - That's why we already incorporated multiset semantics from the start

Beyond RPQs?

Unions of Regular Path Queries

- These are easy to deal with
  - Essentially, one just needs a good multiset semantics for PRs to deal with unions
  - That's why we already incorporated multiset semantics from the start

Conjunctive RPQs

Beyond RPQs?

Unions of Regular Path Queries

- These are easy to deal with
  - Essentially, one just needs a good multiset semantics for PRs to deal with unions
  - That's why we already incorporated multiset semantics from the start

Conjunctive RPQs

- We looked at conjunctions of RPQs (plus projection)
- PRs open up interesting aspects of query optimization

Beyond RPQs?

Unions of Regular Path Queries

- These are easy to deal with
  - Essentially, one just needs a good multiset semantics for PRs to deal with unions
  - That's why we already incorporated multiset semantics from the start

Conjunctive RPQs

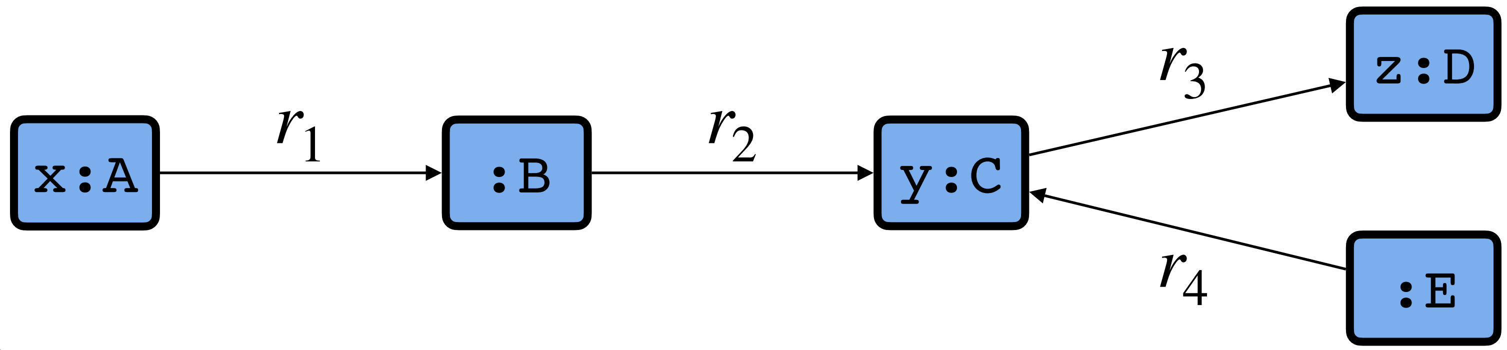
- We looked at conjunctions of RPQs (plus projection)
- PRs open up interesting aspects of query optimization

We're looking into those

Conjunctions of (2)RPQs

Take the query

$\pi_{x,z}$ (

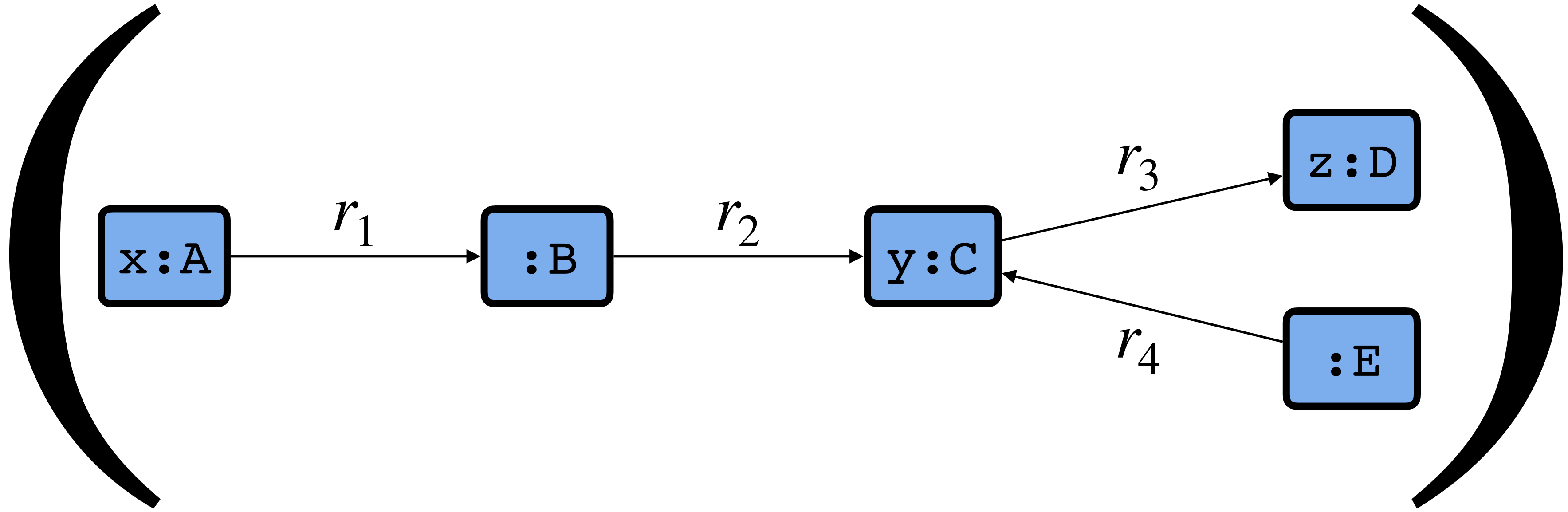


)

Conjunctions of (2)RPQs

Take the query

$\pi_{x,z}$



Lemma

For a given set of nodes U and an RPQ r, you can compute in linear time

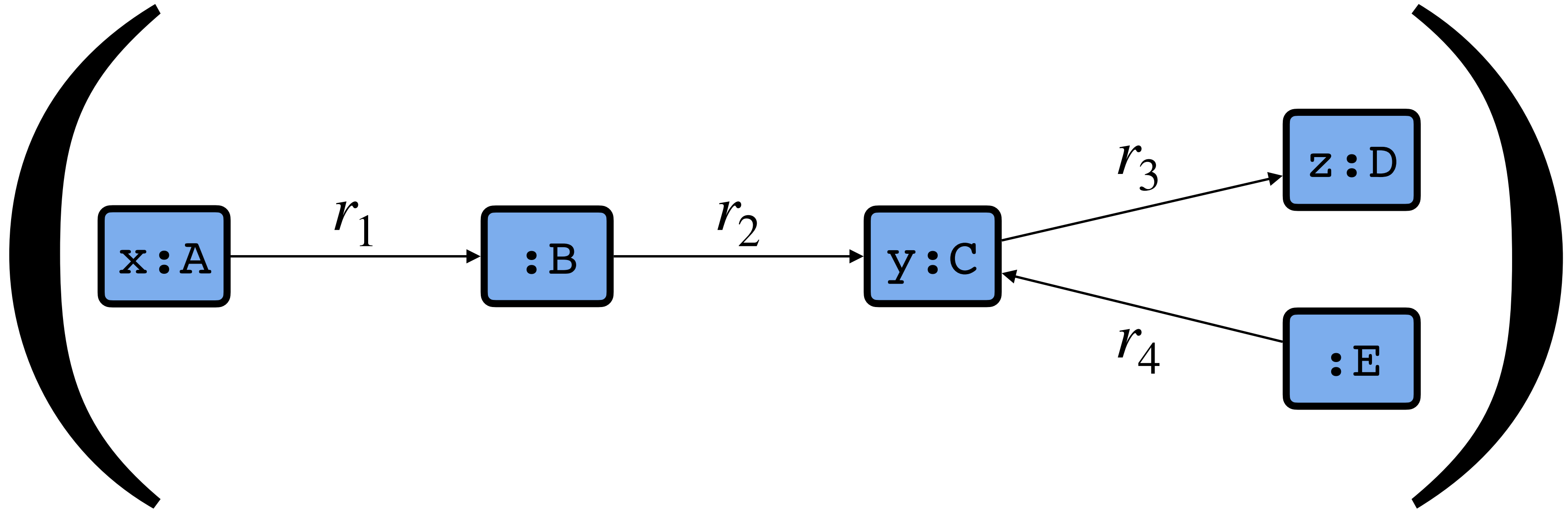
- the set V such that there's a path
  - from some node in U
  - to some node in V
- a PR that contains all these paths

Step 1:
Take the A-nodes of the graph, apply the lemma to get candidates for :B

Conjunctions of (2)RPQs

Take the query

$\pi_{x,z}$



Lemma

For a given set of nodes U and an RPQ r, you can compute in linear time

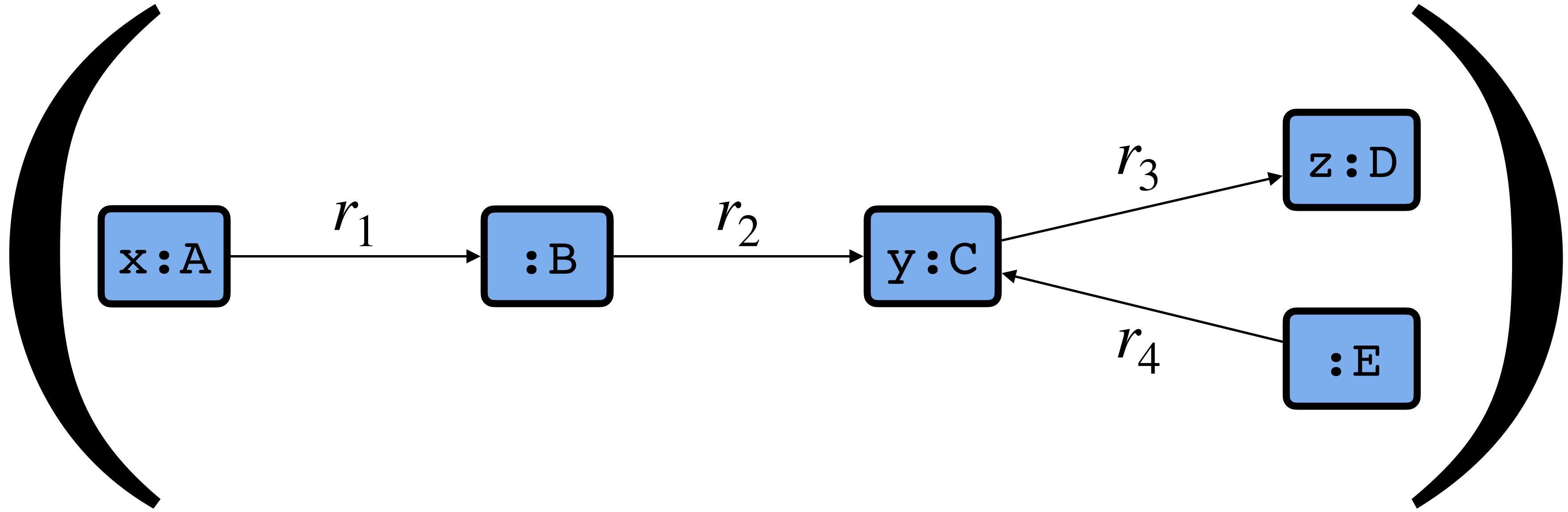
- the set V such that there's a path
  - from some node in U
  - to some node in V
- a PR that contains all these paths

Step 1':
Take the A-nodes of the graph, apply the lemma to get candidates for y:C

(With tables for intermediate results, already this step costs exponential time)

Conjunctions of (2)RPQs

Take the query

$\pi_{x,z}$ 

Lemma

For a given set of nodes U and an RPQ r, you can compute in linear time

- the set V such that there's a path
  - from some node in U
  - to some node in V
- a PR that contains all these paths

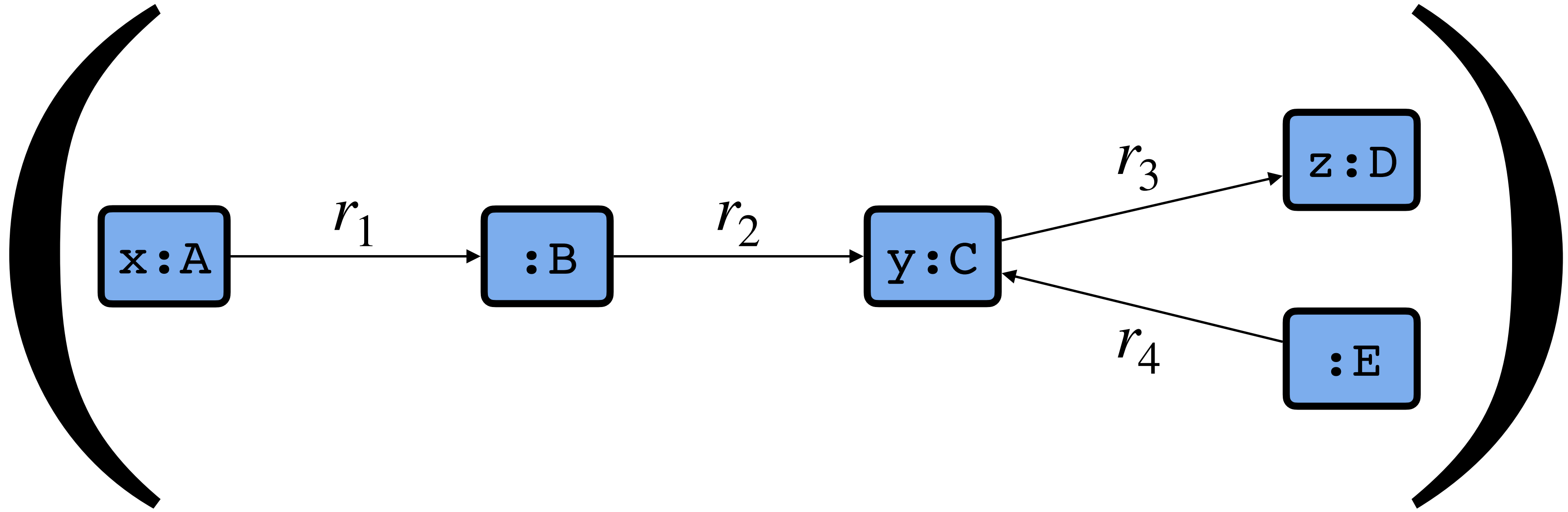
Step 2:
Apply the lemma again to get candidates for z:D and :E

Step 3:
Trim everything; using backward reachability

Conjunctions of (2)RPQs

Take the query

$\pi_{x,z}$



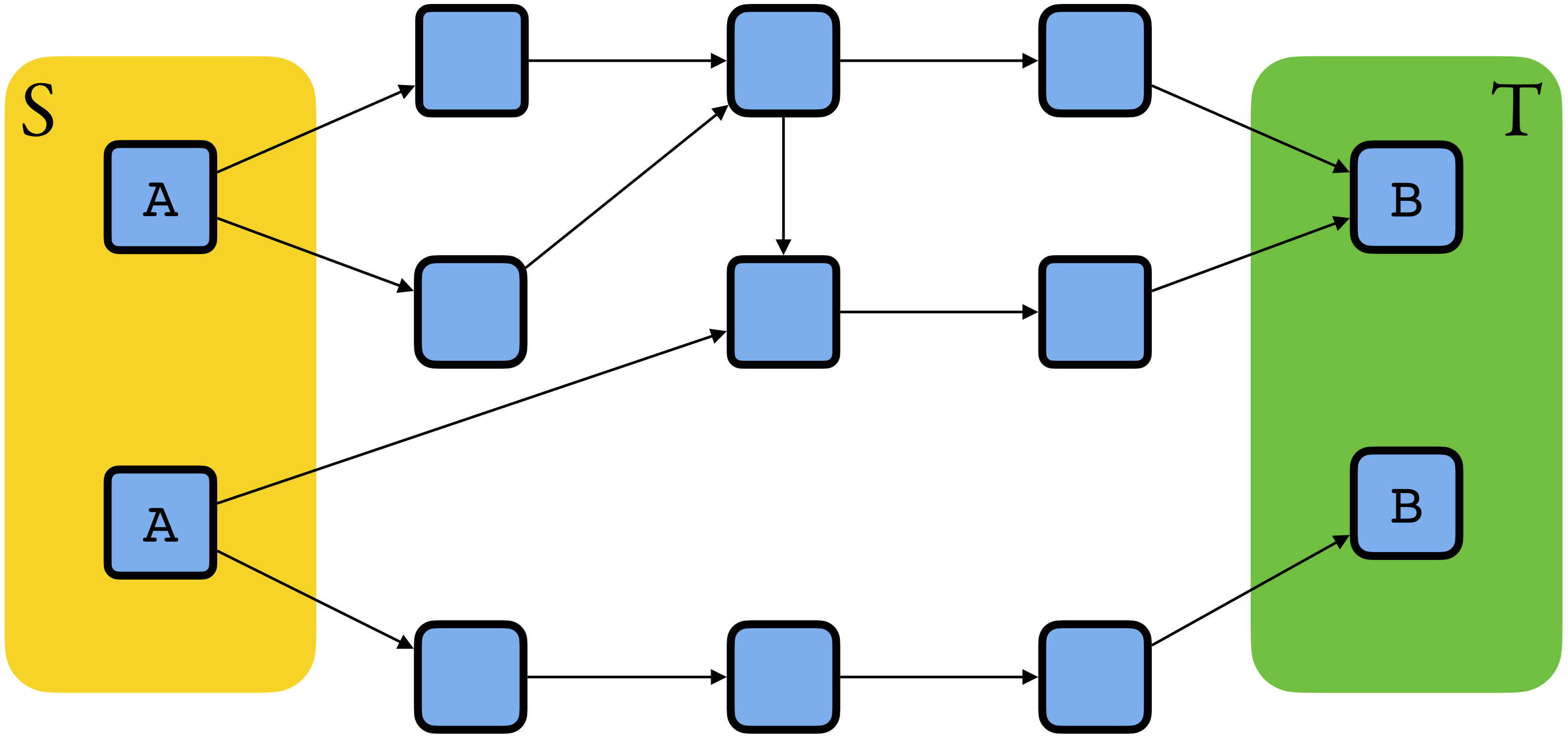
Lemma

For a given PR R of G and a pair of nodes (u, v) of G, we can compute the number of paths from u to v represented by R in linear time

Step 4:
Use counting results to efficiently count cardinalities of endpoint pairs in the result

Conjunctions of (2)RPQs

Insight

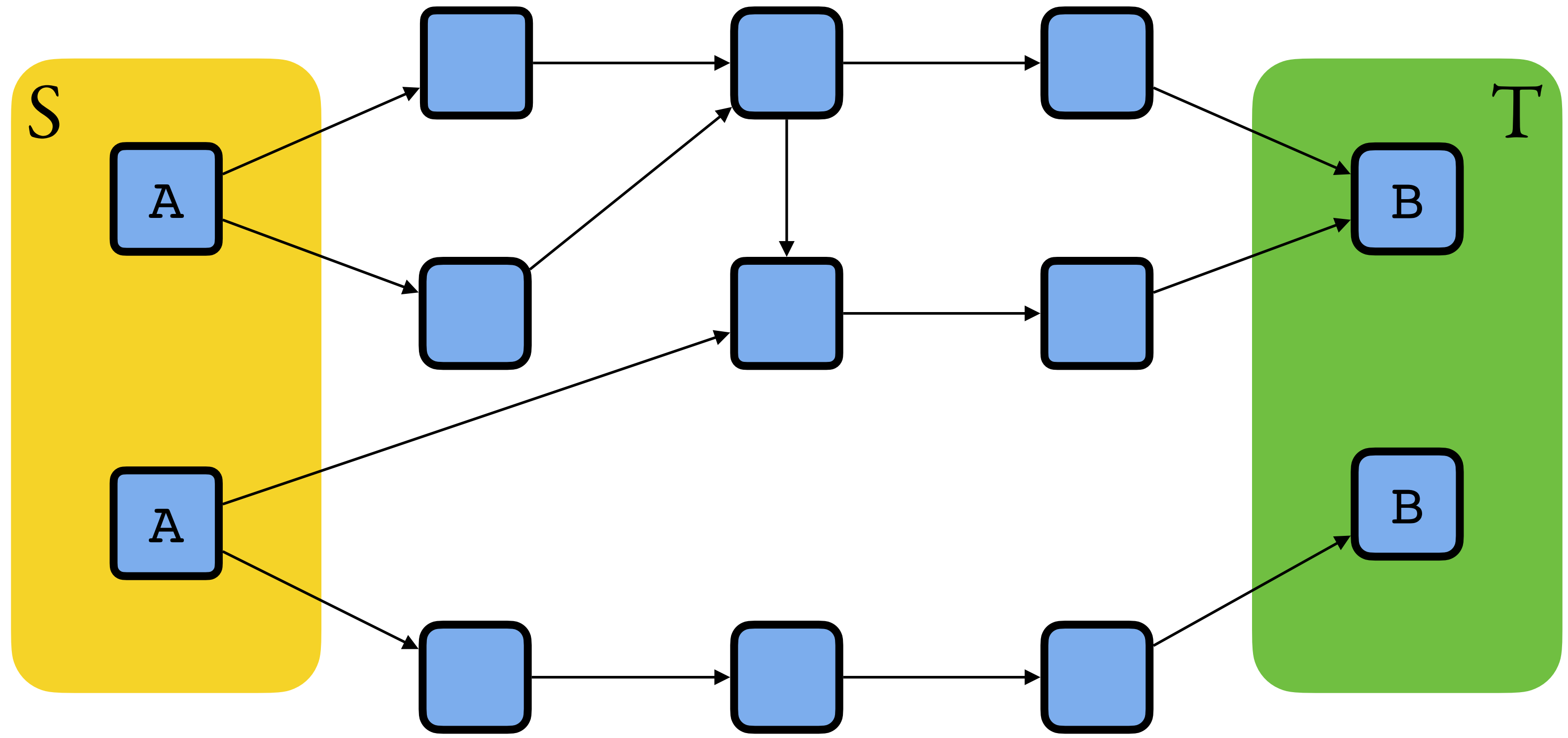


Using PRs, we can represent
"all paths from A-nodes to B-nodes"
in different ways

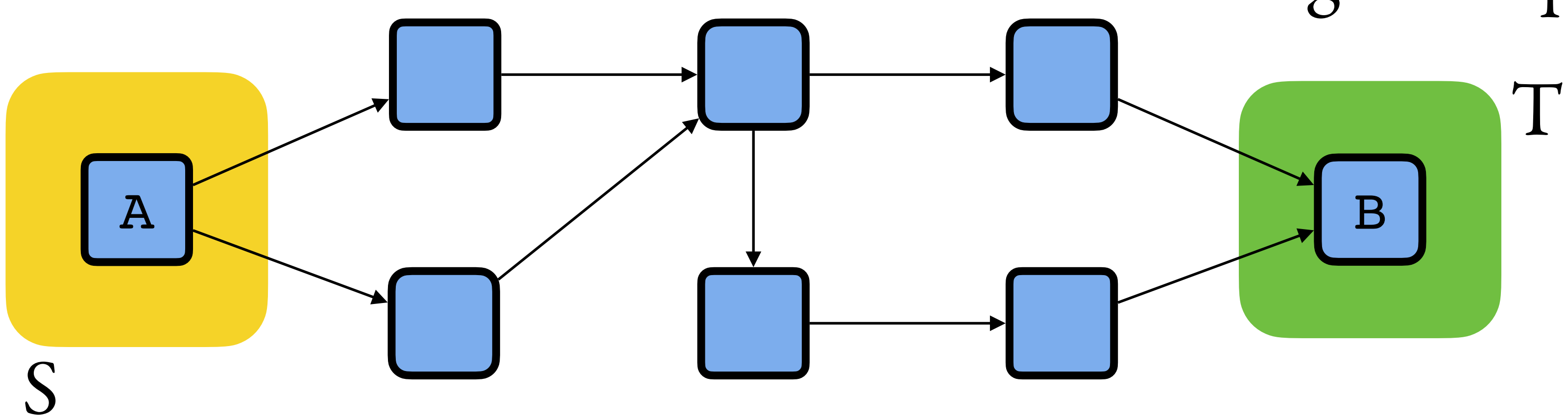
1. As you see it here

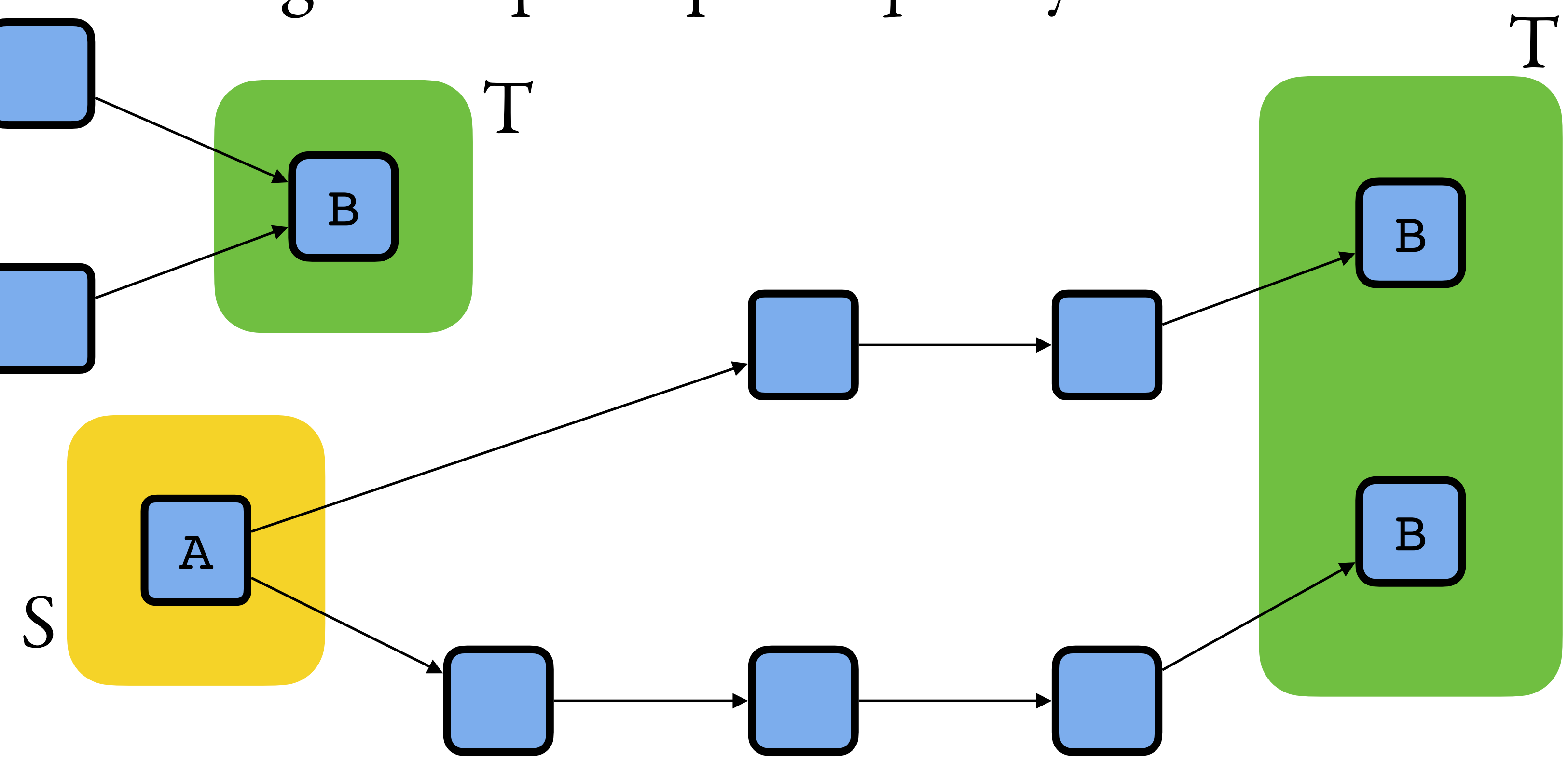
Conjunctions of (2)RPQs

Insight



2. In a way that allows you to get "endpoint pairs" quickly





This gives you a lot of flexibility for CRPQ evaluation

Concluding

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

1. PRs are succinct, so they may help a lot
2. PRs are graphs, so they may help here too
3. We're not HCI experts, so we don't know how PRs help users to digest results (but who knows?)

Concluding

1. The exponential output challenge
2. The composability challenge
3. The "output representation" challenge

1. PRs are succinct, so they may help a lot
2. PRs are graphs, so they may help here too
3. We're not HCI experts, so we don't know how PRs help users to digest results (but who knows?)

Our contribution

We introduce the concept of PRs
that we believe can become quite helpful
for evaluating modern graph DB queries
in which paths are first-class citizens

Thanks!

Questions?

--> happy to chat here

--> feel free to reach out by email