Factorization matters in *large graphs*

Nikolay Yakovets

Ag
Avant**Graph**

Os Open Source	**Mm** Main Memory	**Ra** Recursive Analytics
Vc Vectorized Compiled	**Wco** Worst-case Optimal	**Fz** Factorized
Tm Temporal	**Re** Reachability	**Td** Topo+data

not your grandma's graph engine!

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

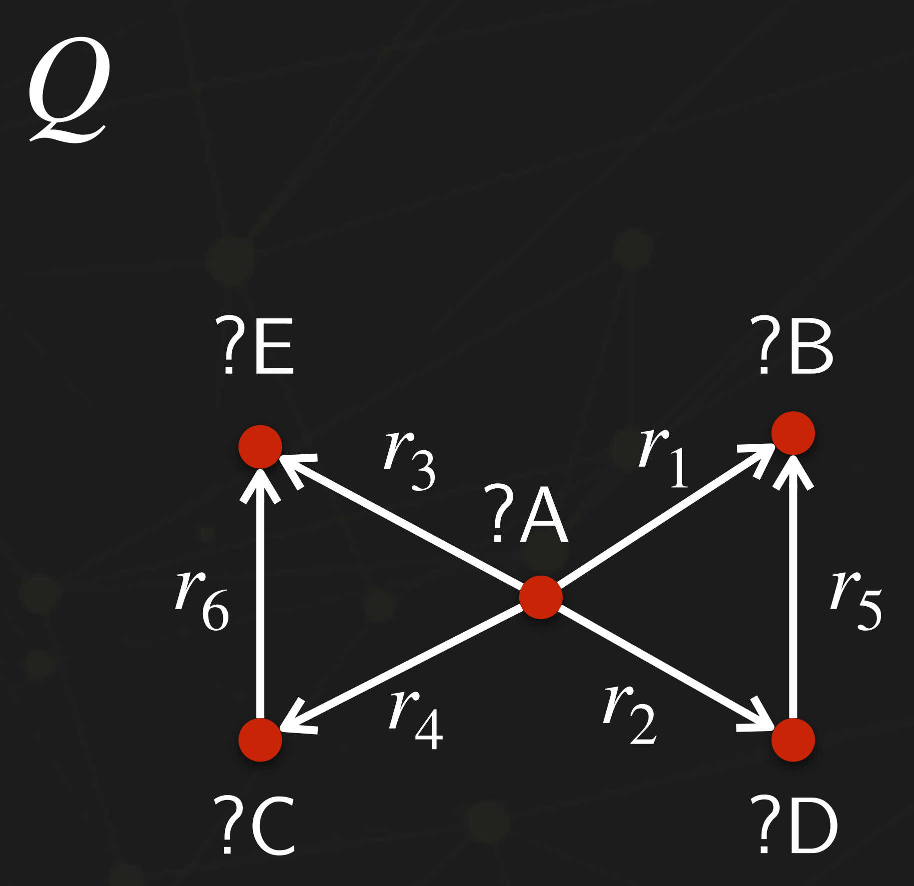
Conjunctive queries (over graphs)

A conjunctive query (CQ) is a "query graph":

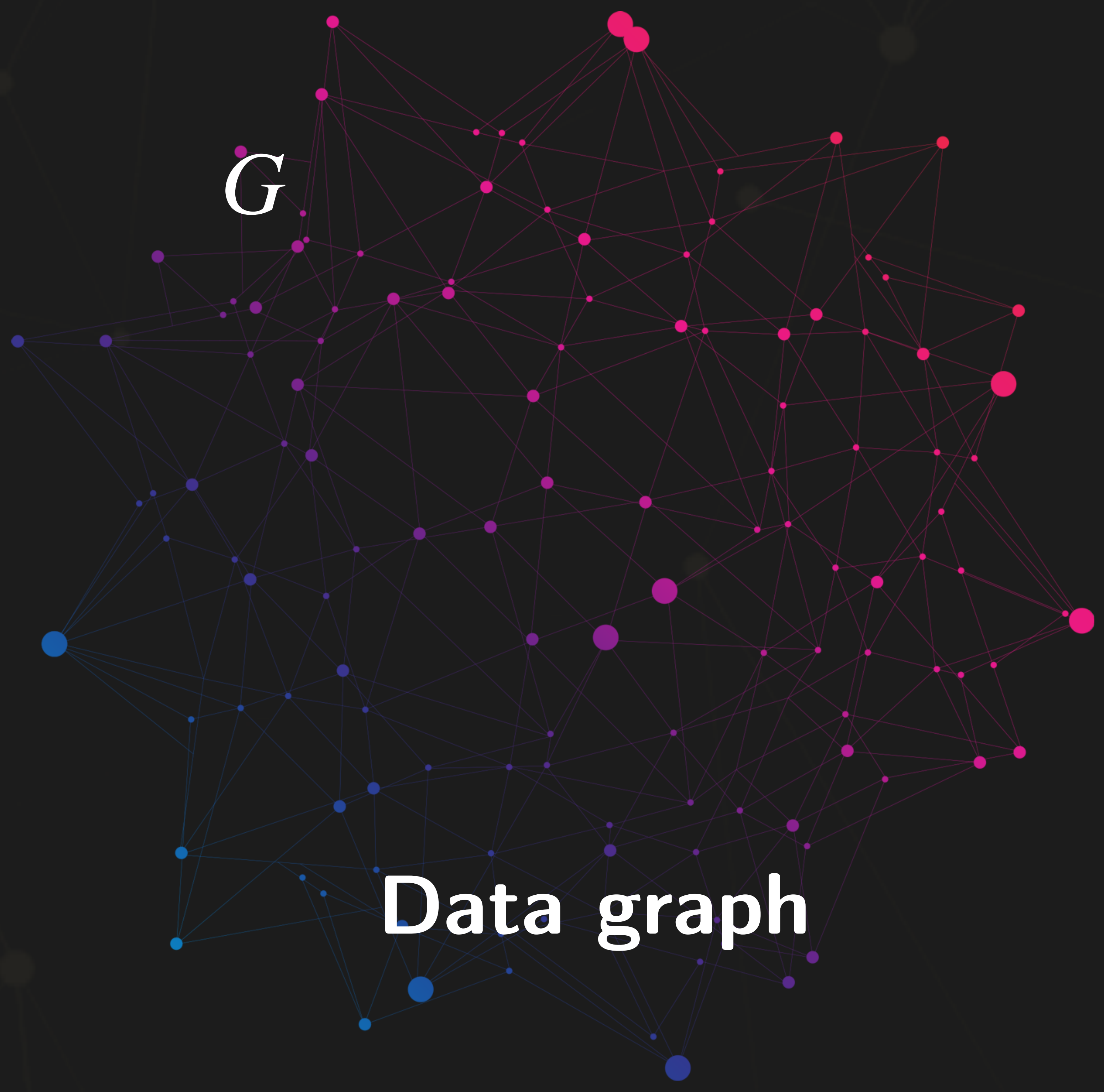
- Its "nodes" are the query's binding variables; and "edges" between nodes are constituent queries

The answer of a CQ:

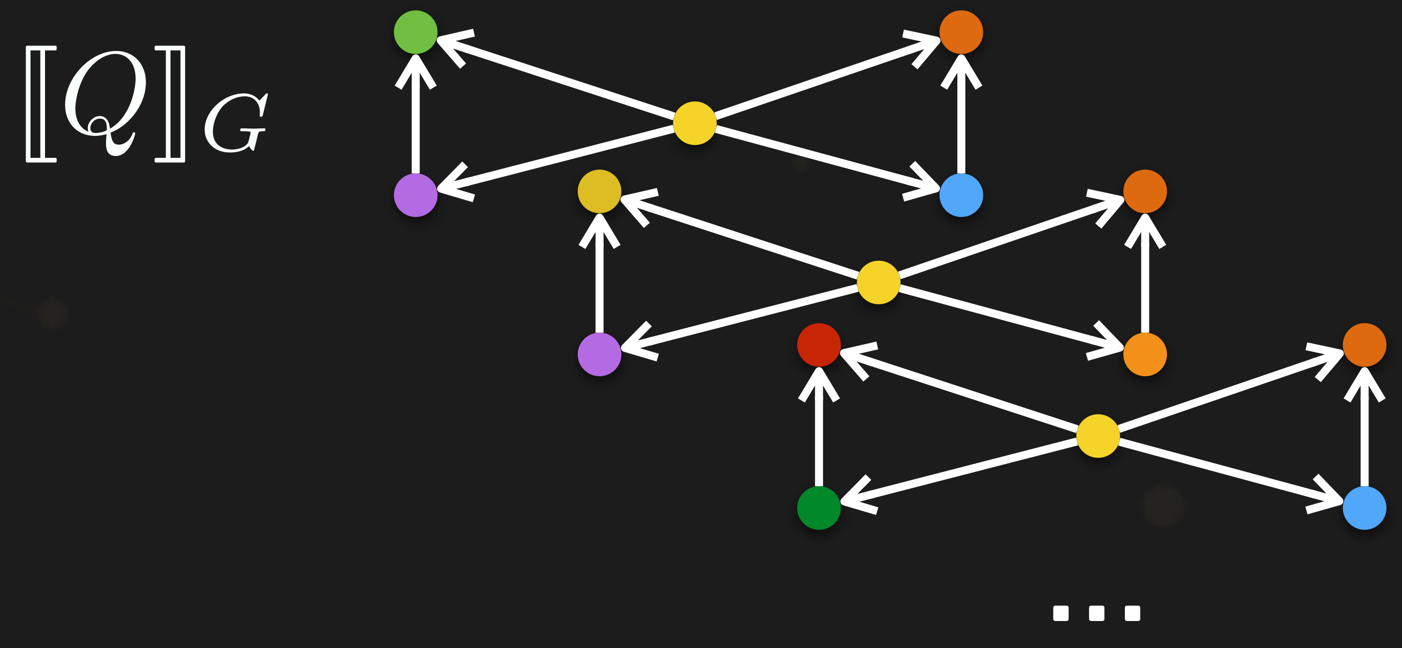
- Are tuples of nodes (embeddings) that match the conjuncts, joining in the way the query asks



Query graph



Data graph

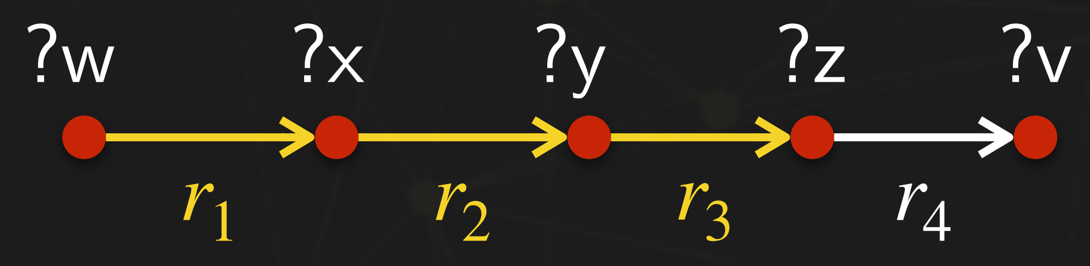


?A	?B	?C	?D	?E
Adam	Sara	Nick	Paul	Alice
Adam	Sara	Nick	Jeff	Tom
Adam	Sara	Mike	Paul	Mark
Adam	Sara	Mike	Jeff	Parke
...	...	...	...	...
...	...	...	...	...

Embeddings

Conjunctive queries - challenges

Q



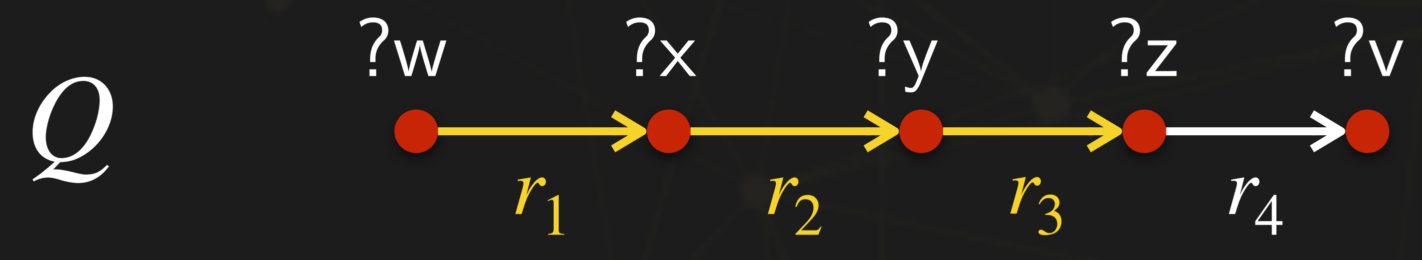
IR

{1,5,9,13} {1,5,9,14} {1,5,9,15}
{2,5,9,13} {2,5,9,14} {2,5,9,15}
{3,5,9,13} {3,5,9,14} {3,5,9,15}
{1,5,9,12} {2,5,9,12} {3,5,9,12}

Cardinality: The evaluation of graph queries is (often) dominated by the size of the **intermediate results** (IR)

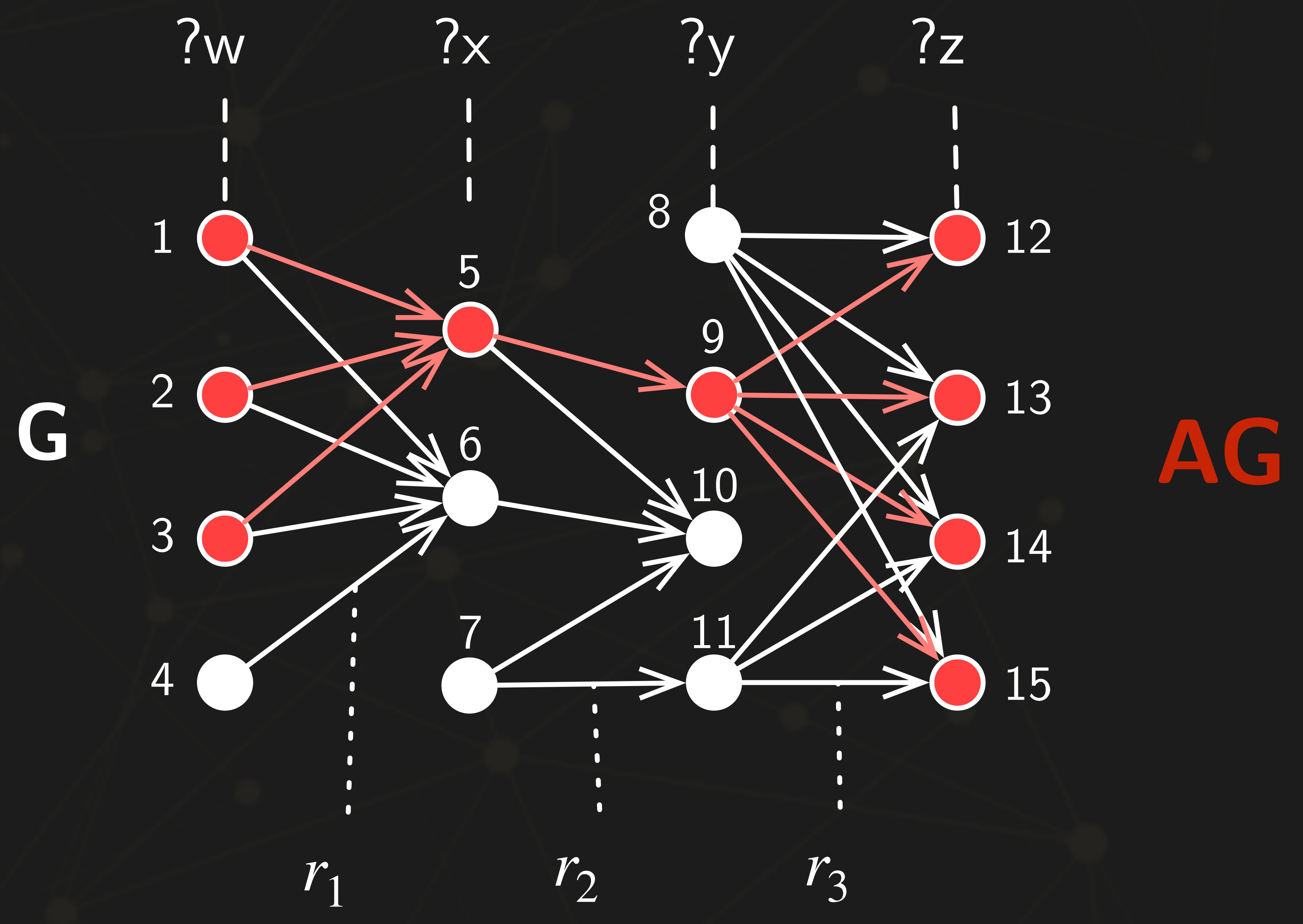
- Queries are often very **selective**
- But, during the evaluation, the size of the intermediate results can **grow exponentially** (in the size of the graph), due to many-to-many joins inherent in graph queries

The answer-graph approach



IR

{1,5,9,13} {1,5,9,14} {1,5,9,15}
{2,5,9,13} {2,5,9,14} {2,5,9,15}
{3,5,9,13} {3,5,9,14} {3,5,9,15}
{1,5,9,12} {2,5,9,12} {3,5,9,12}



Factorization: One way of reducing the size of intermediate results is to apply the concept of *factorization*

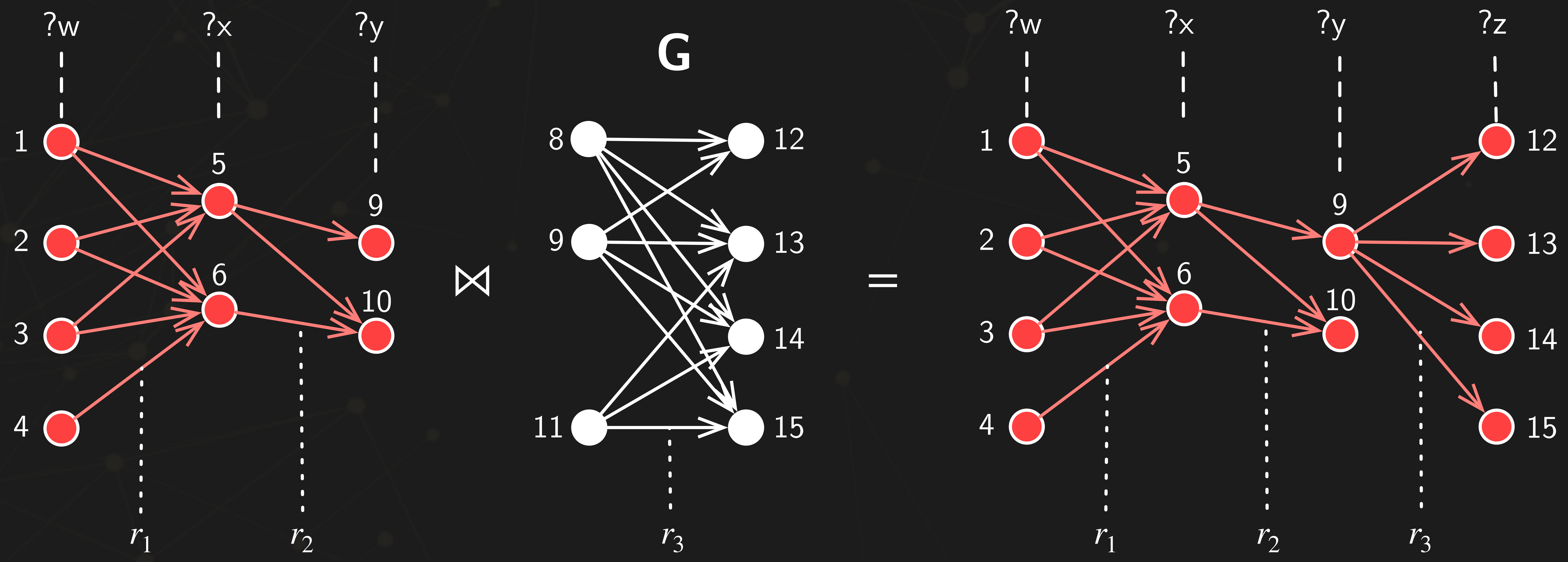
- the common unique node-pair patterns
- we call these factorized node-pairs, **Answer Graph (AG)**

Defactorization: To find all final result tuples (i.e., embeddings), all we need to do is to *defactorize* this answer graph

The evaluation model

Answer graph generation

- Edge extension: to fetch data edges from the graph
- Node burn-back: to ensure the generated AG is minimal (in size)

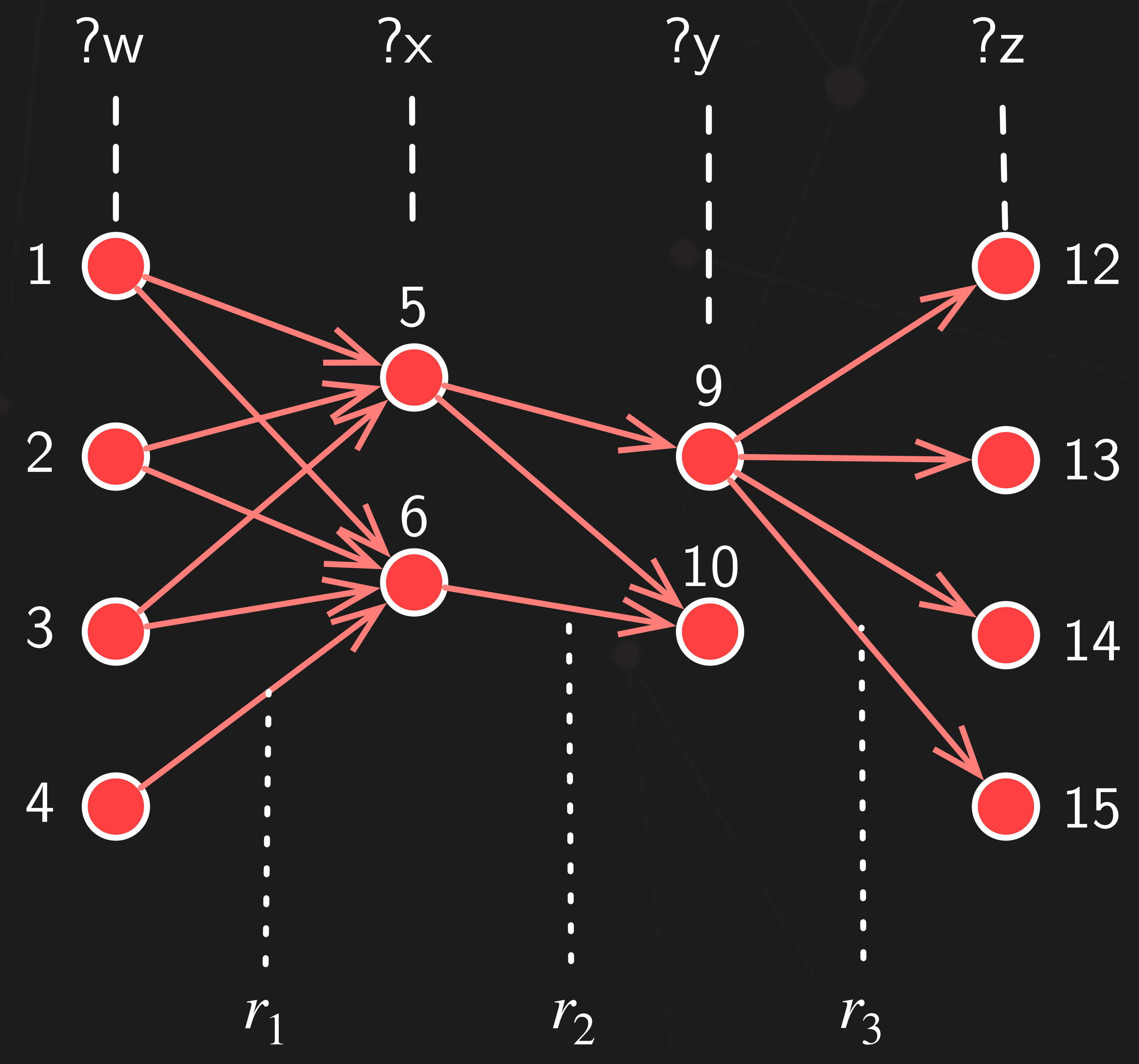


edge extension

The evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges

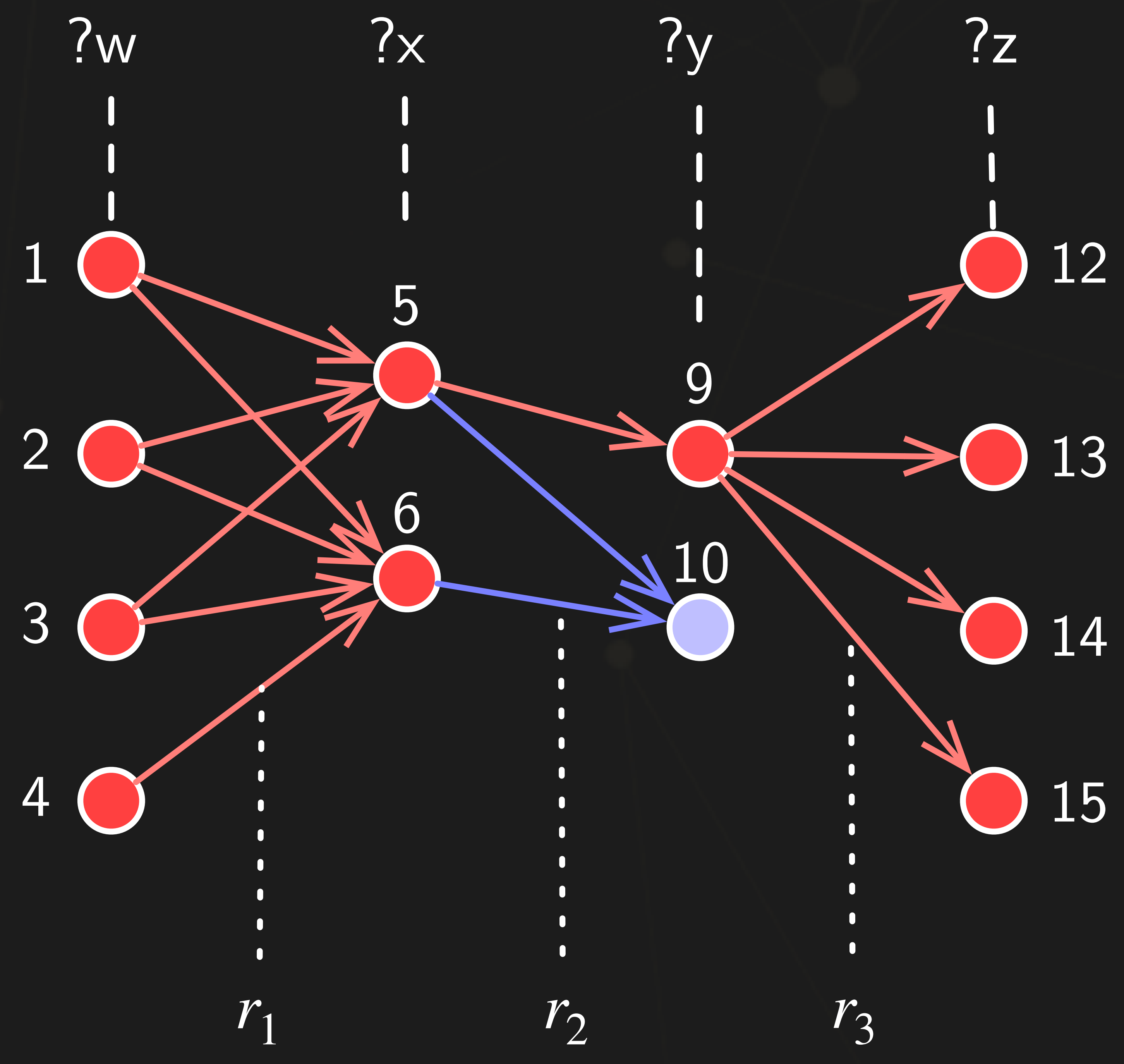


burning nodes

The evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges

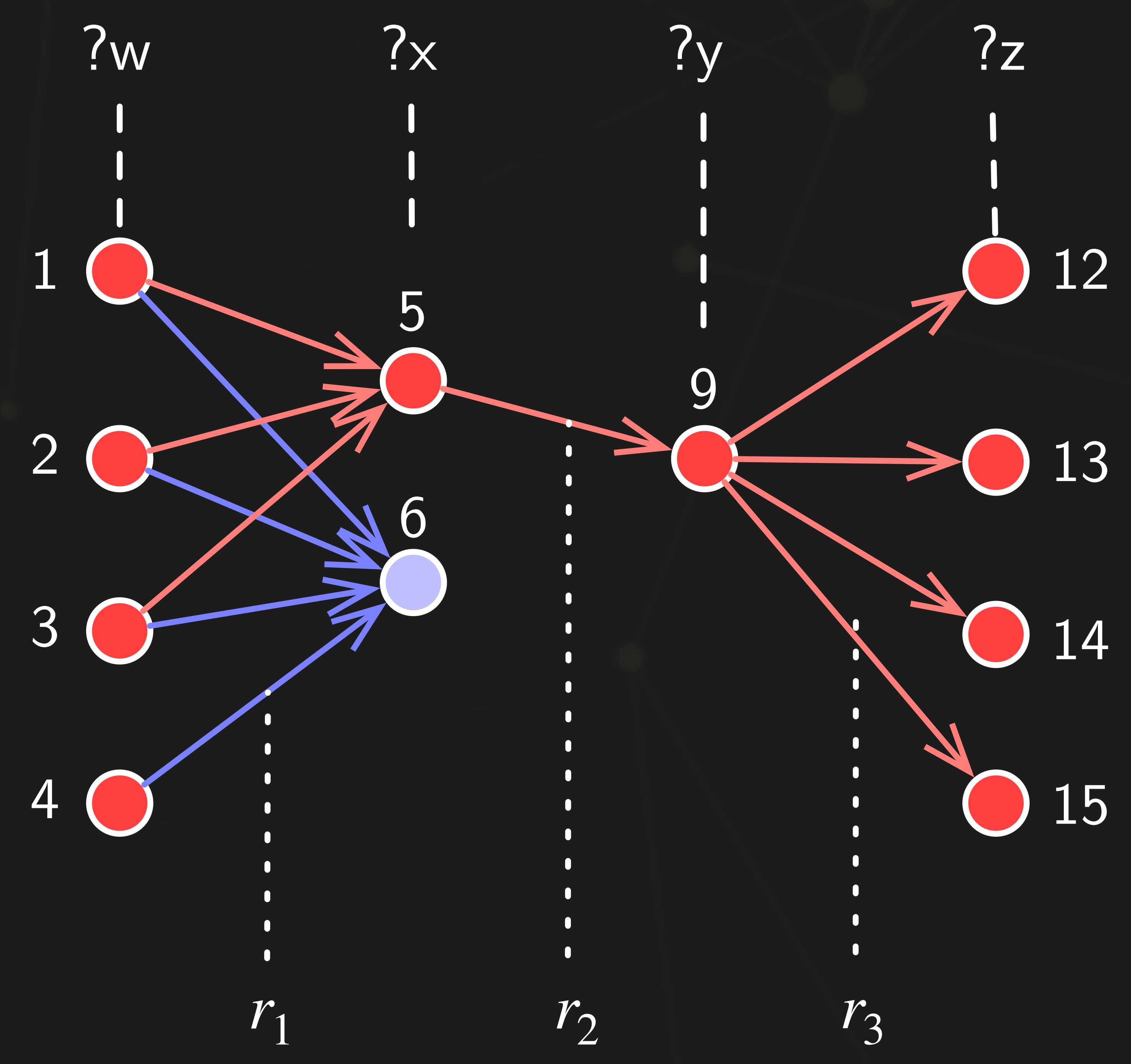


burning nodes - step 1

The evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges

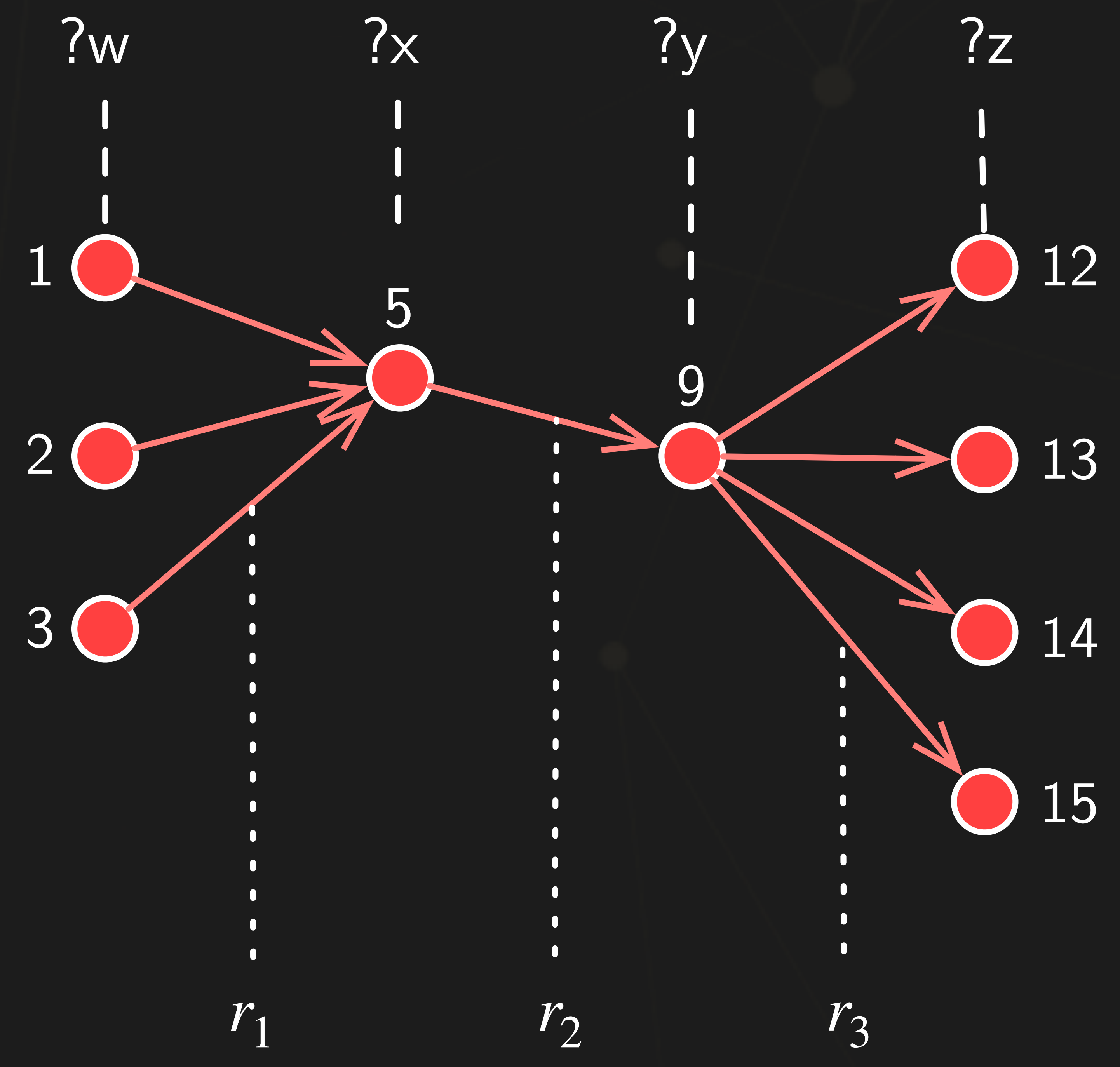


burning nodes - step 2

The evaluation model

Node burn-back is a cascaded filter operation and has three processes

- Nodes of AG that are not extendable with new edges are removed
- Removing any node from AG will trigger the removal of edges that are attached to this node along with removal of nodes at the other side of removed edges

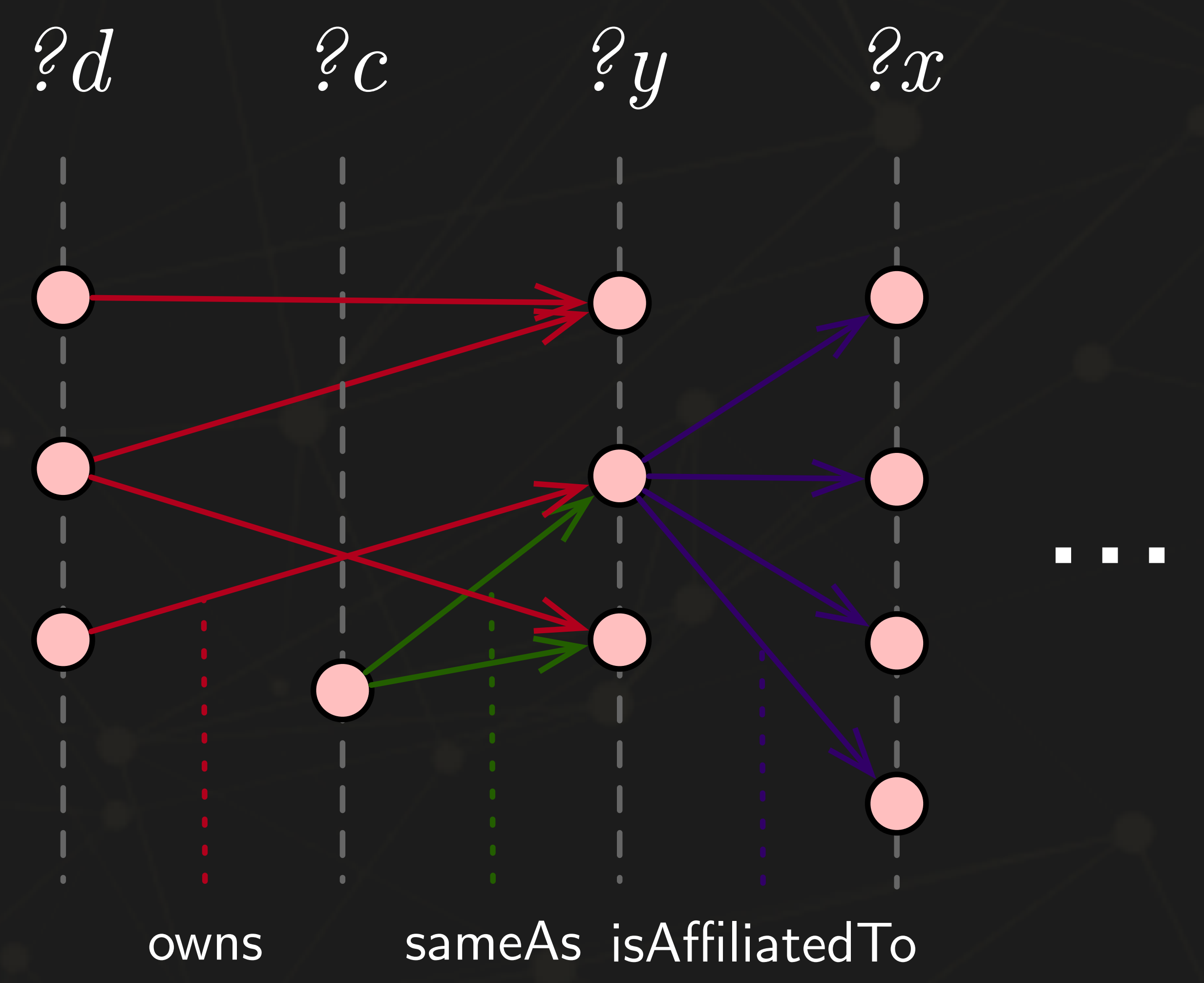


done burning nodes

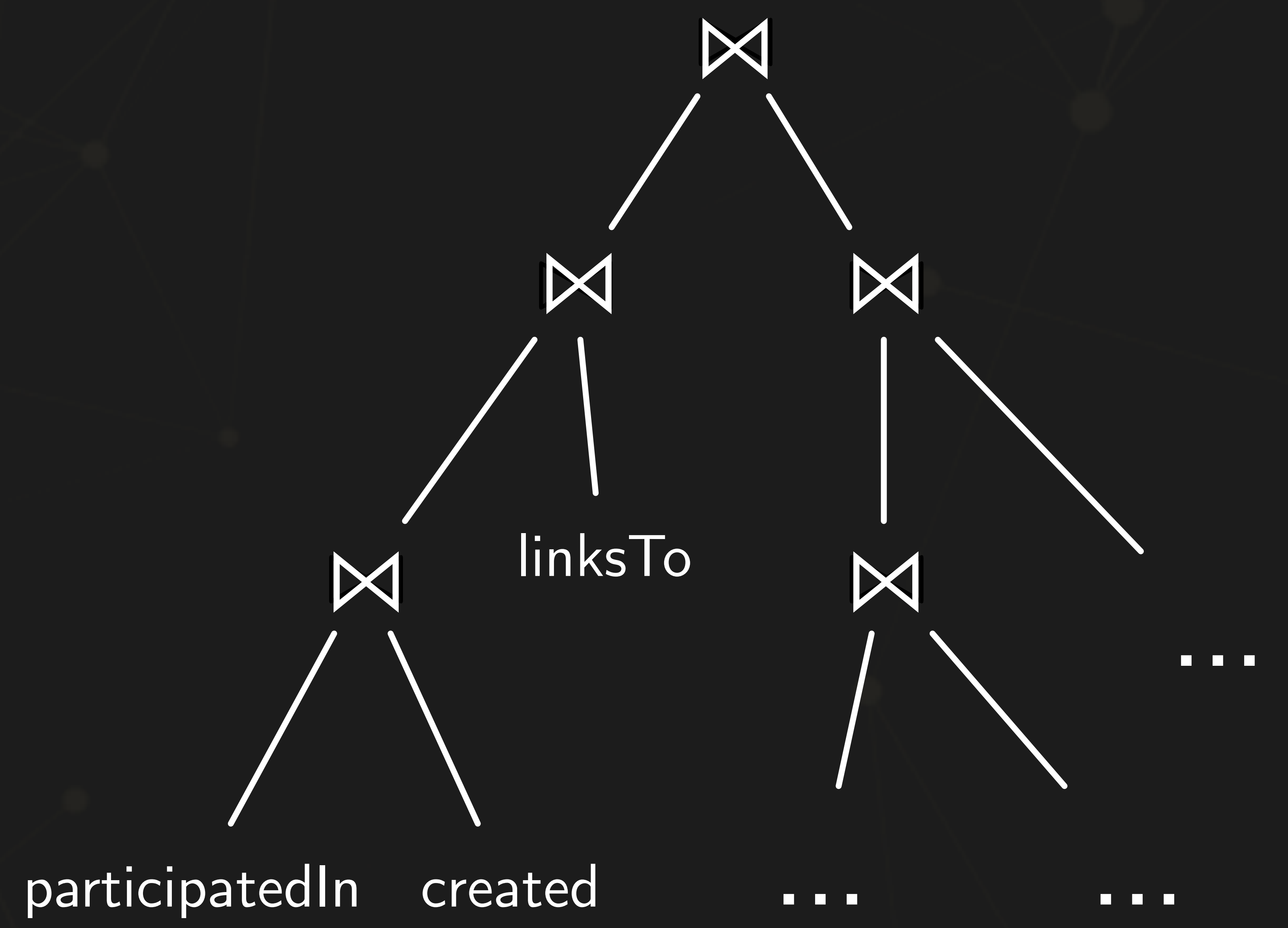
The evaluation model

Embedding generation

- generates the final answer tuples of a CQ from an AG
- executing a join tree to further filter out tuples which do not belong to the final result



Answer graph



Embedding plan

$?d$	$?c$	$?y$	$?a$	$?x$	$?z$	$?m$	$?b$	$?f$	$?e$
60	13	2	10	4	11	22	23	24	14
60	13	2	10	6	11	22	23	24	14
60	13	2	10	15	11	22	23	24	14
..	..	..	..	..	..	..	..	..	..

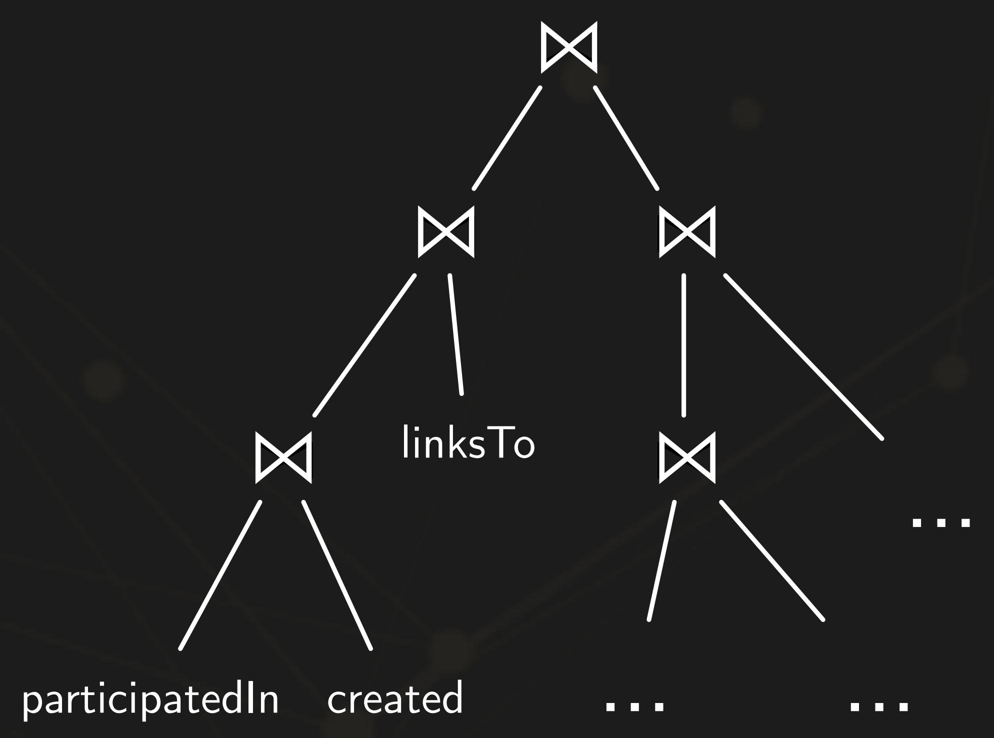
Embeddings

WireFrame: the framework

Query plan

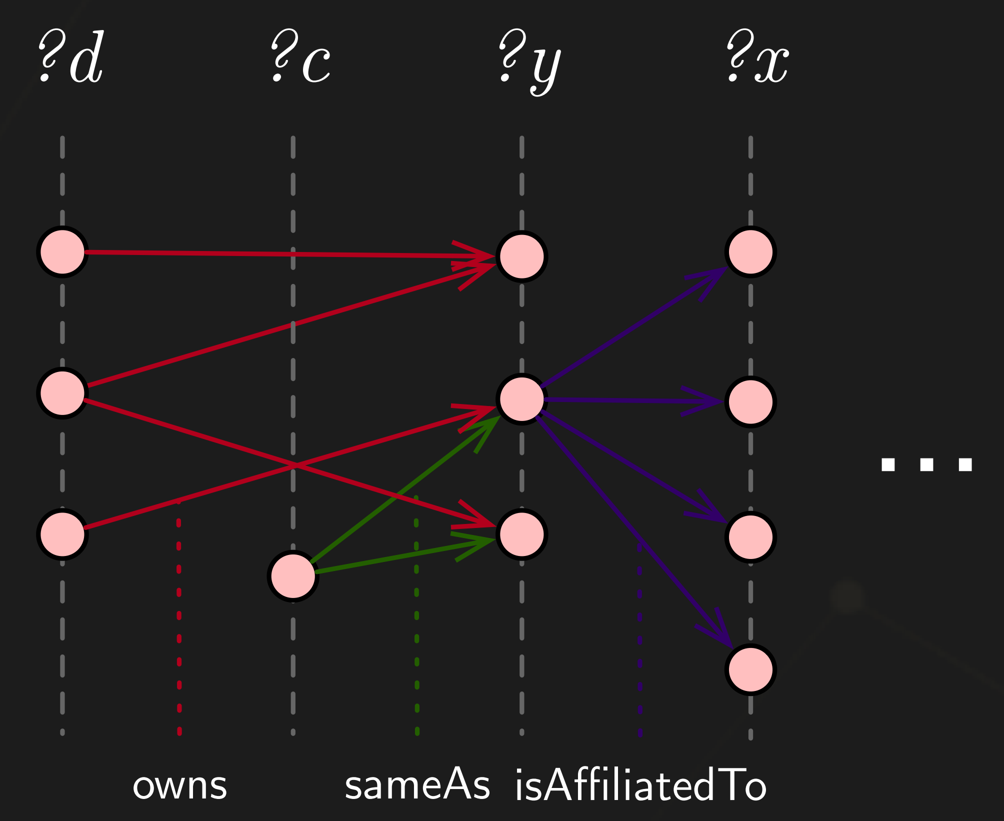
?d	?c	?y	?a	?x	?z	?m	?b	?f	?e
60	13	2	10	4	11	22	23	24	14
60	13	2	10	6	11	22	23	24	14
60	13	2	10	15	11	22	23	24	14
..	..	..	..	..	..	..	..	..	..

Embeddings



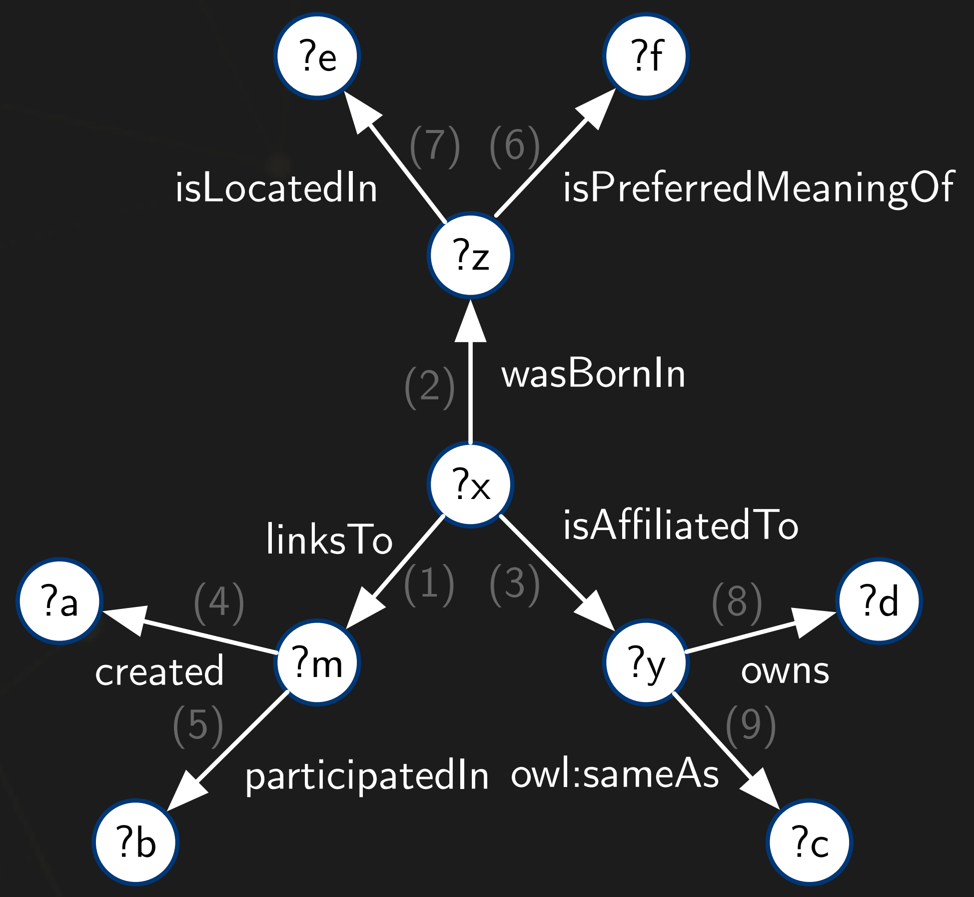
Embedding plan

Answer Graph

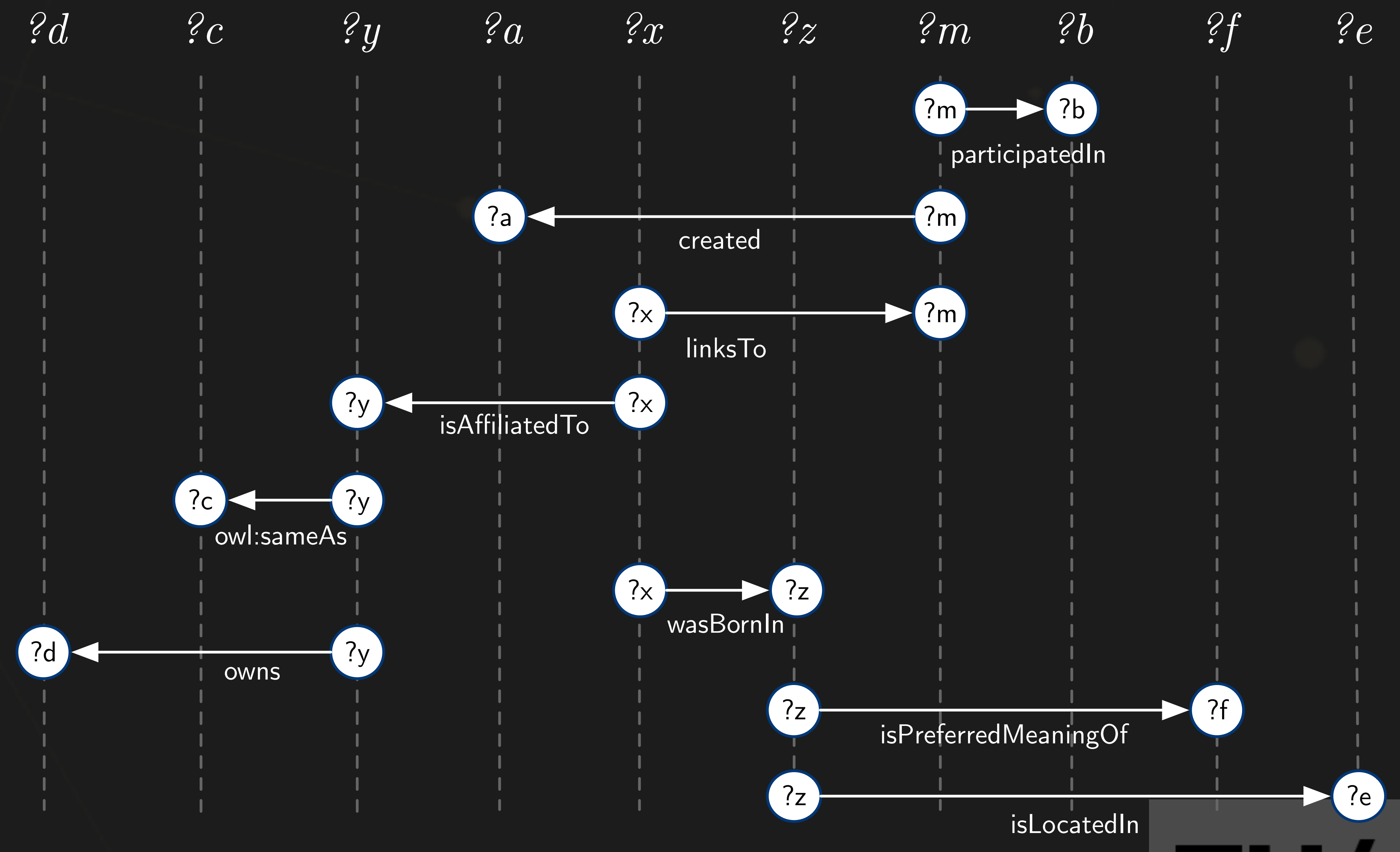


select distinct ?x, ?m, ?y, ?z, ?a
?b, ?c, ?d, ?e, ?f
where {
?x linksTo ?m .
?x isAffiliatedTo ?y .
?x wasBornIn ?z .
?m participatedIn ?b .
?m created ?a .
?y owl:sameAs ?c .
?y owns ?d .
?z isLocatedIn ?e .
?z isPreferredMeaningOf ?f . }

Conjunctive query



Query graph



Answer graph plan

The planners

A cost-based dynamic programming approach to find an optimal plan for evaluating the answer graph (AG) of a query

The plan space

- all execution orders of edges/sub-patterns in the CQ (left-deep, right-deep, zig-zag) + bushy
- finding the best plan is equivalent to enumerating over all possible orders

Plan enumeration and a cost model

- Dynamic programming to find the optimal order
- In each iteration, the planner uses its cardinality estimators to calculate the cost of a tree plan and updates the cardinality of nodes due to the node burn-back

The evaluation model

- execute based on the selected order of edges
- node burn-back filters the dead-end nodes accordingly

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Ideal answer graph

We call an answer graph ideal if it contains only those edges which participate in at least one final embedding.

Theorem: Node burn-back results in an ideal AG for **acyclic** graph CQs with number of cascaded semi-joins bounded in $O(|Q|)$.

Pf.:

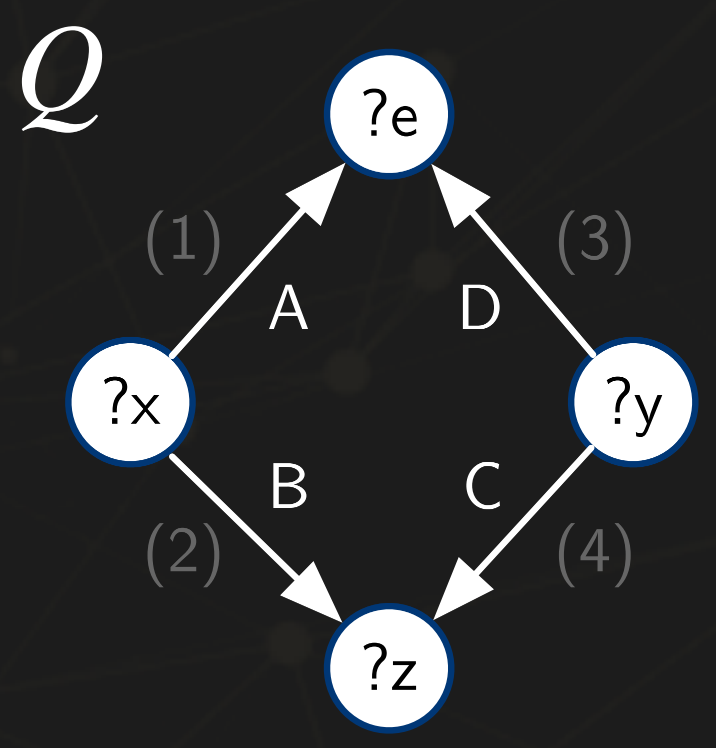
- node burn-back results in an ordered sequence of semi-joins which contain (in correct order) a semi-join ordering produced by the GYO algorithm
- this corresponds to a bounded full reducer semi-join program which guarantees no dangling tuples in the AG for acyclic CQs

Ideal answer graph (for cyclic CQs)

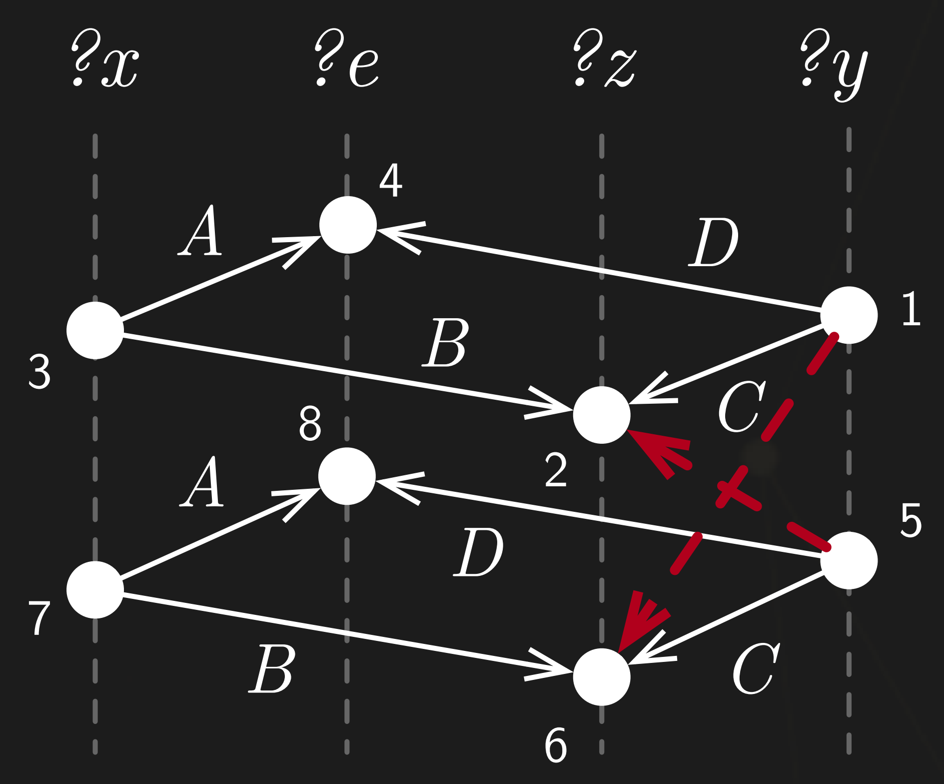
What about cyclic CQs?

Node burn-back does not generate ideal AG for cyclic CQs in a fixed number of cascading semi-joins:

- some of the edges will not participate in the final embeddings
- we can still use this AG in the embedding generation, but it will be more expensive
- can we find an ideal AG for cyclic CQs? And at what cost?



Query graph



Answer graph

$$[\![Q]\!]_G = \begin{matrix}\{3,4,2,1\} \\ \{7,8,6,5\}\end{matrix}$$

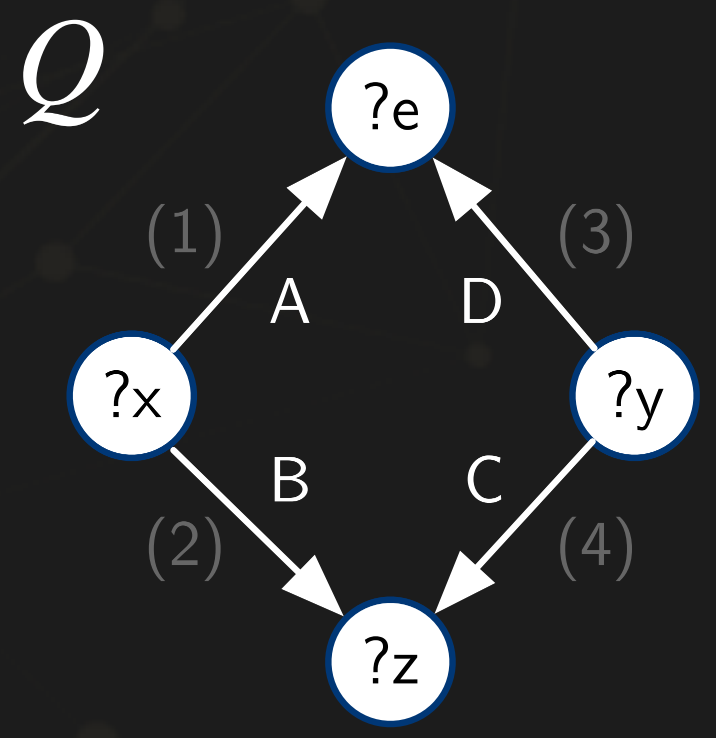
Embeddings

EINDHOVEN UNIVERSITY OF TECHNOLOGY

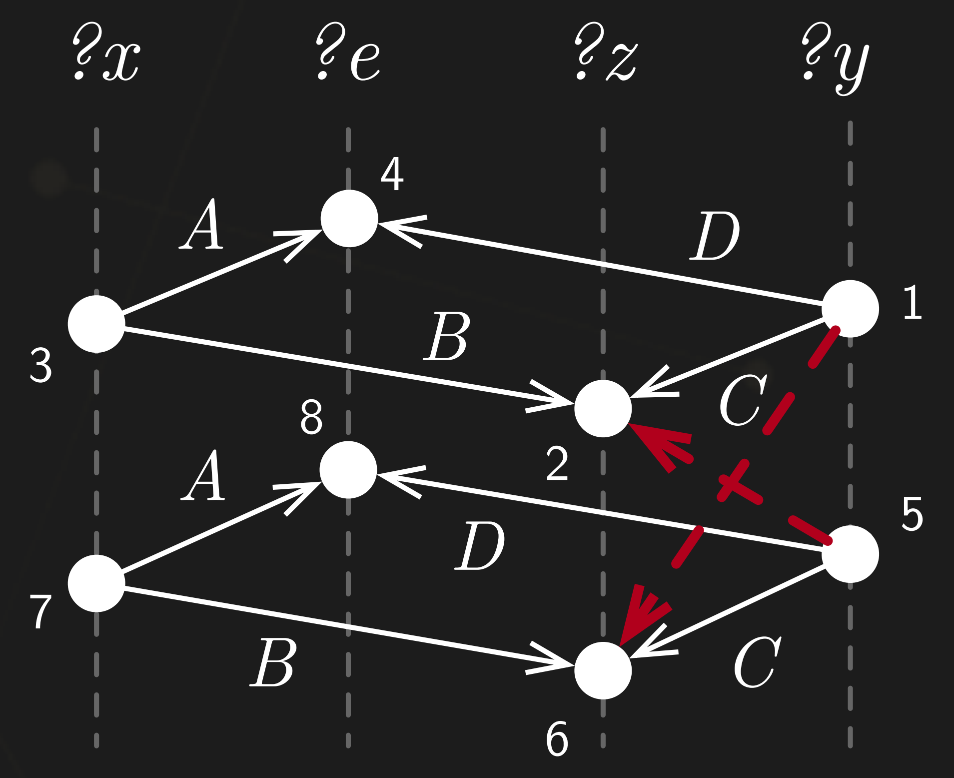
Triangulator

We triangulate a query graph to reduce the AG further:

- during evaluation, additional end-points which correspond to triangles are materialized
- this materialization becomes an edge in a query graph called a ***chord***



Query graph

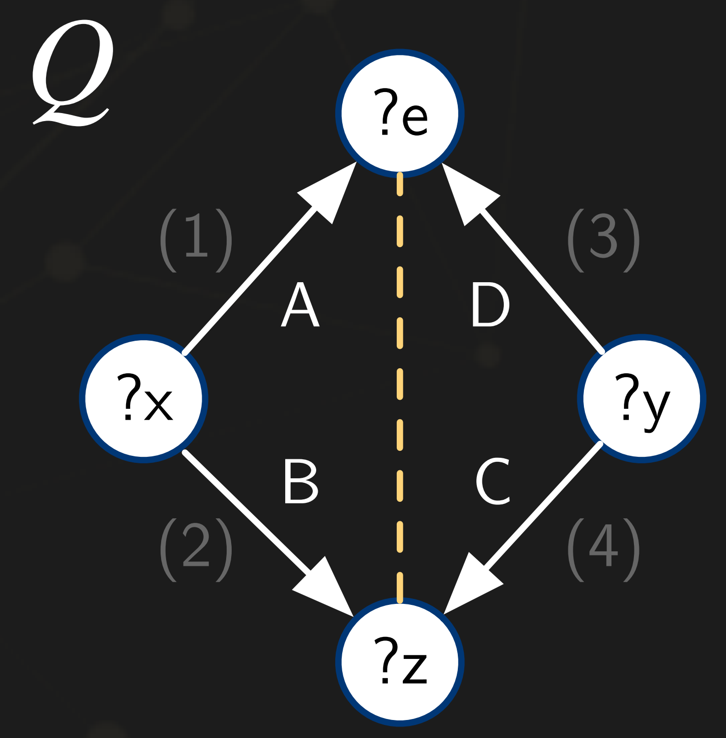


Answer graph

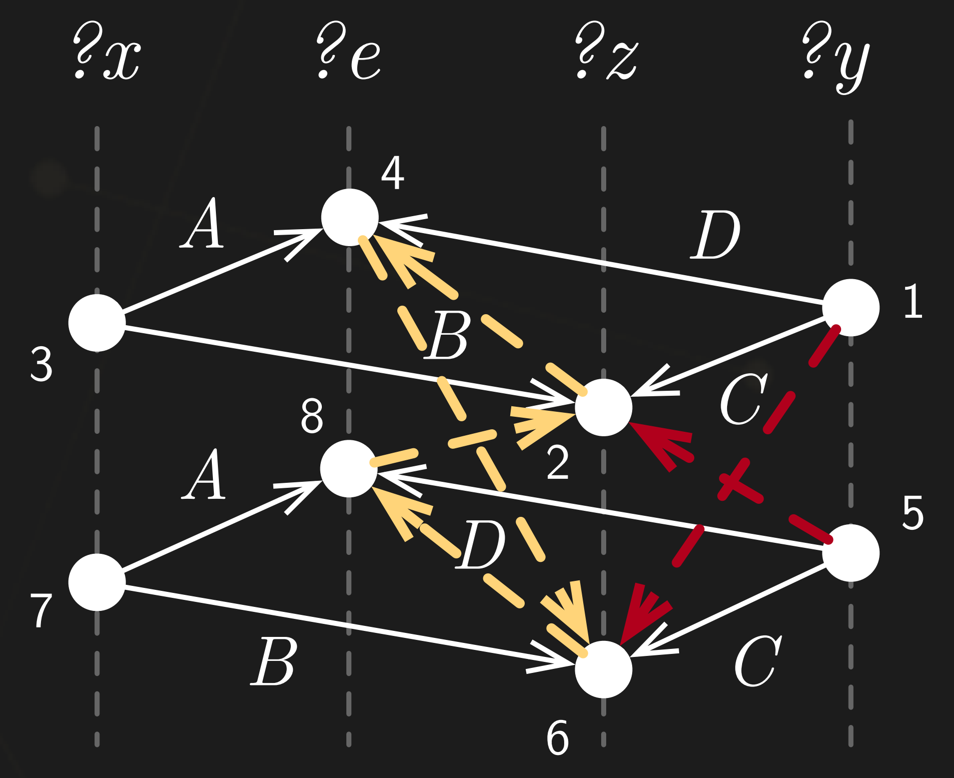
Triangulator

We triangulate a query graph to reduce the AG further:

- during evaluation, additional end-points which correspond to triangles are materialized
- this materialization becomes an edge in a query graph called a ***chord***



Query graph

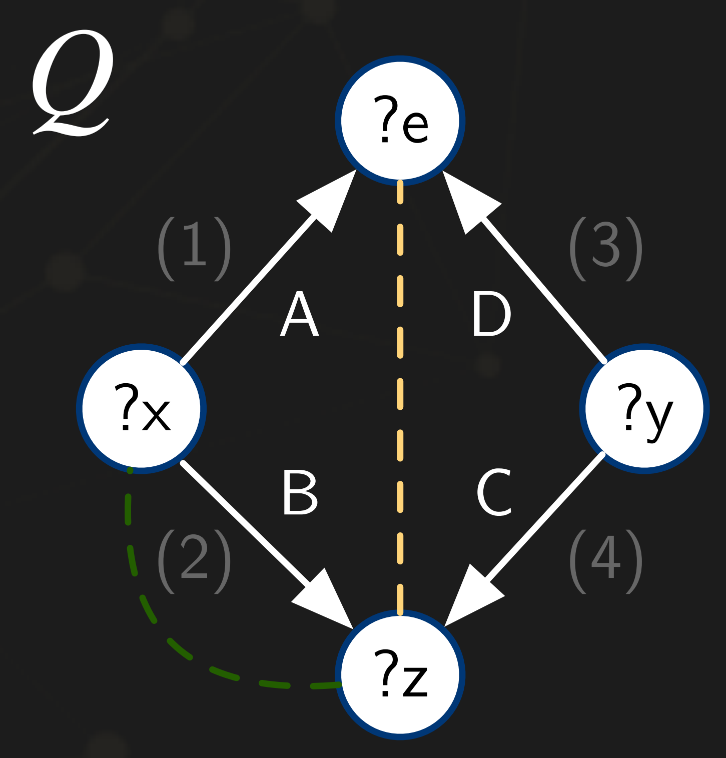


Answer graph

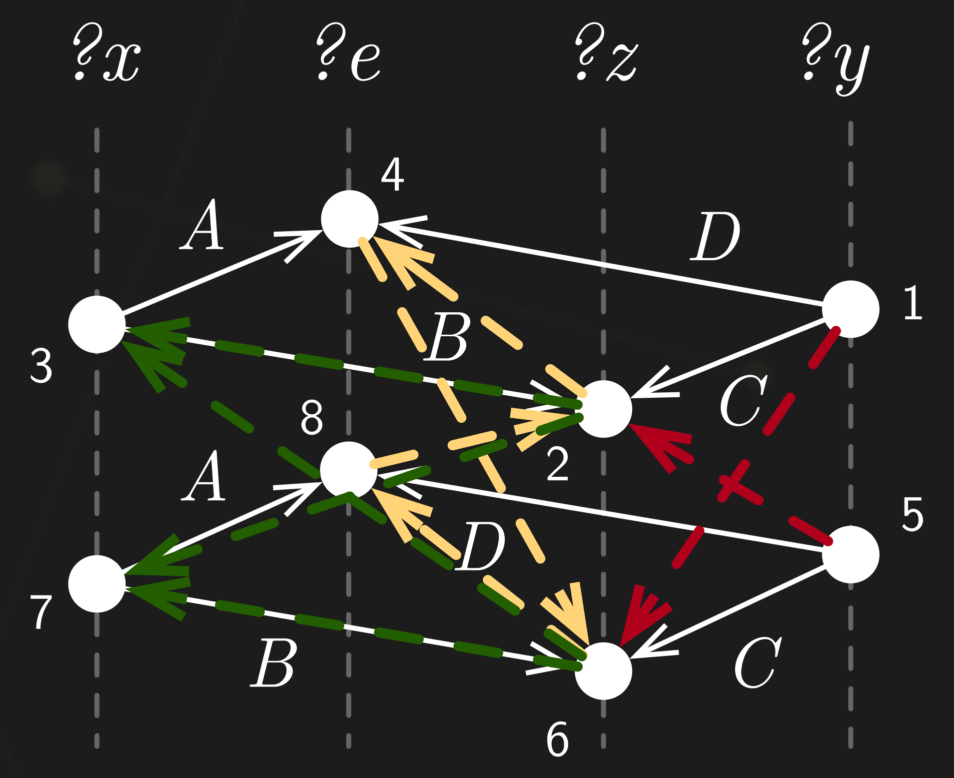
Triangulator

We triangulate a query graph to reduce the AG further:

- during evaluation, additional end-points which correspond to triangles are materialized
- this materialization becomes an edge in a query graph called a ***chord***



Query graph

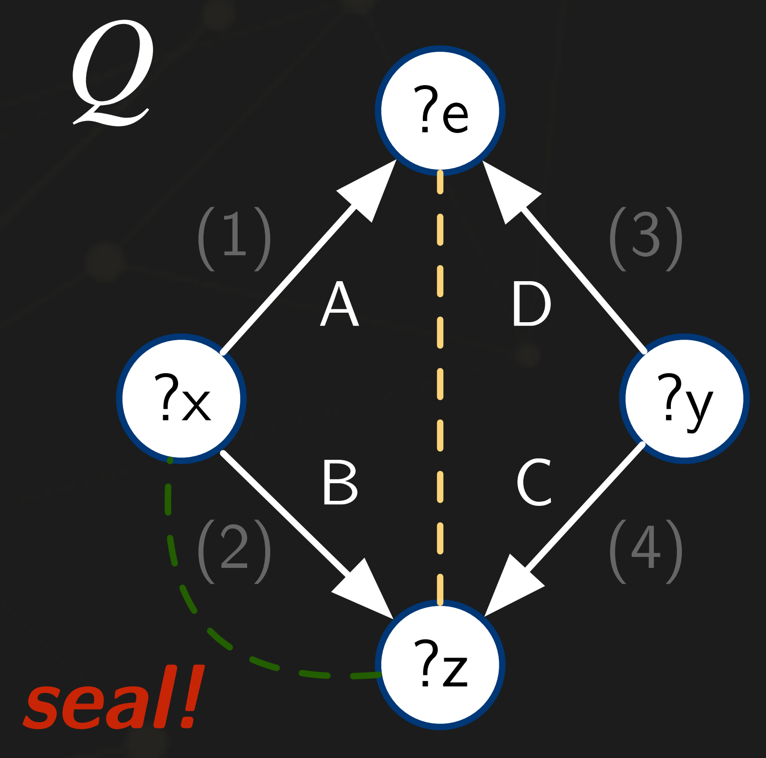


Answer graph

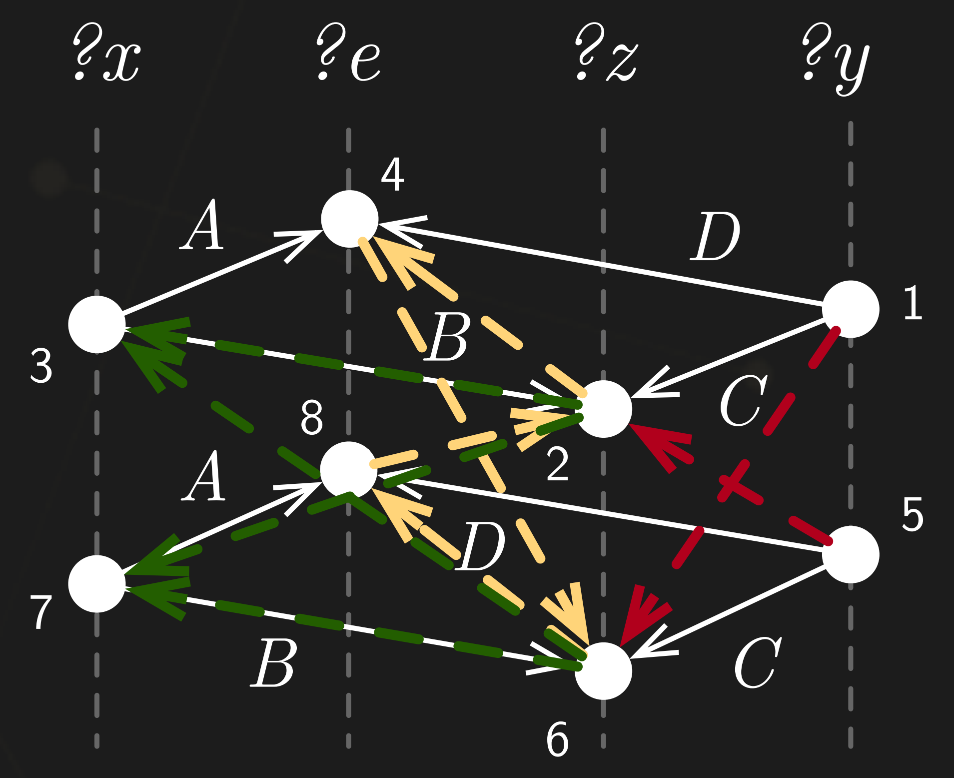
Triangulator

We triangulate a query graph to reduce the AG further:

- whenever a chord intersects a query edge (*-deep, zig-zag plan) or another chord (bushy plan), a **seal** happens
- a seal triggers the **edge burn-back** which removes the chord edges which don't participate in the final embeddings, eventually removing "spurious" edges in the AG, on cascade



Query graph

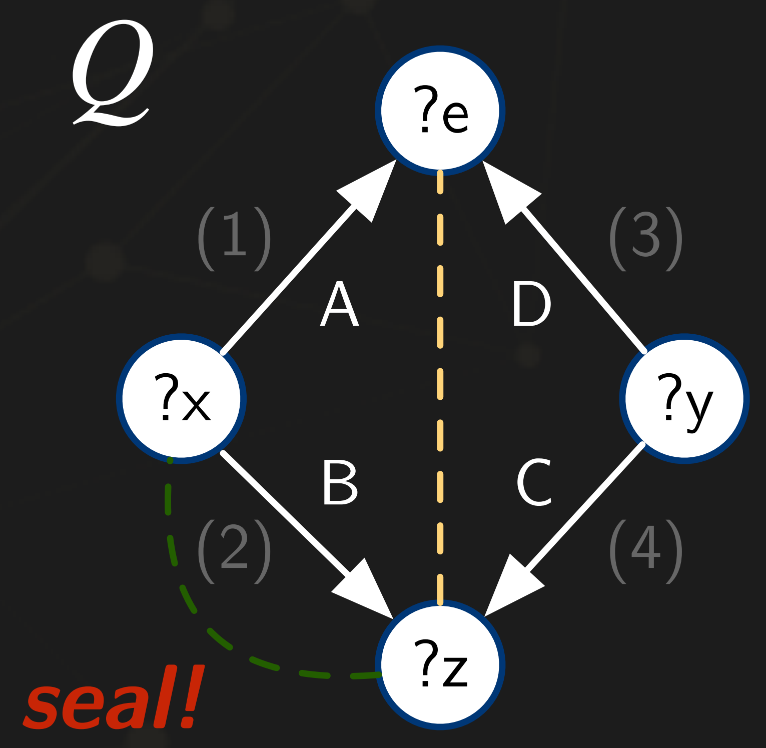


Answer graph

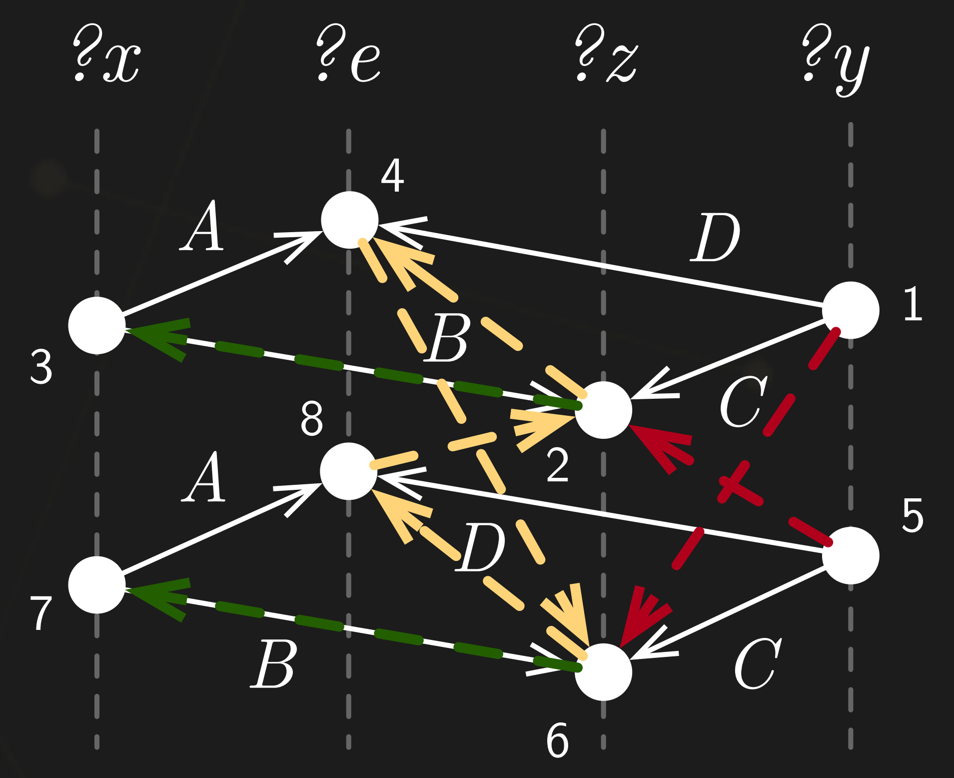
Triangulator

We triangulate a query graph to reduce the AG further:

- whenever a chord intersects a query edge (*-deep, zig-zag plan) or another chord (bushy plan), a **seal** happens
- a seal triggers the **edge burn-back** which removes the chord edges which don't participate in the final embeddings, eventually removing "spurious" edges in the AG, on cascade



Query graph



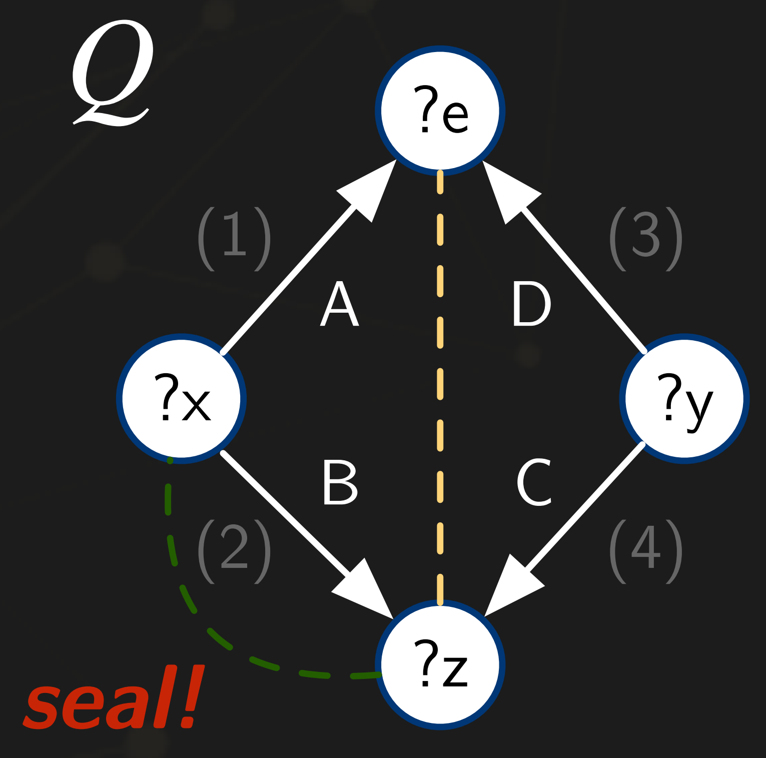
Answer graph

edge burn-back!

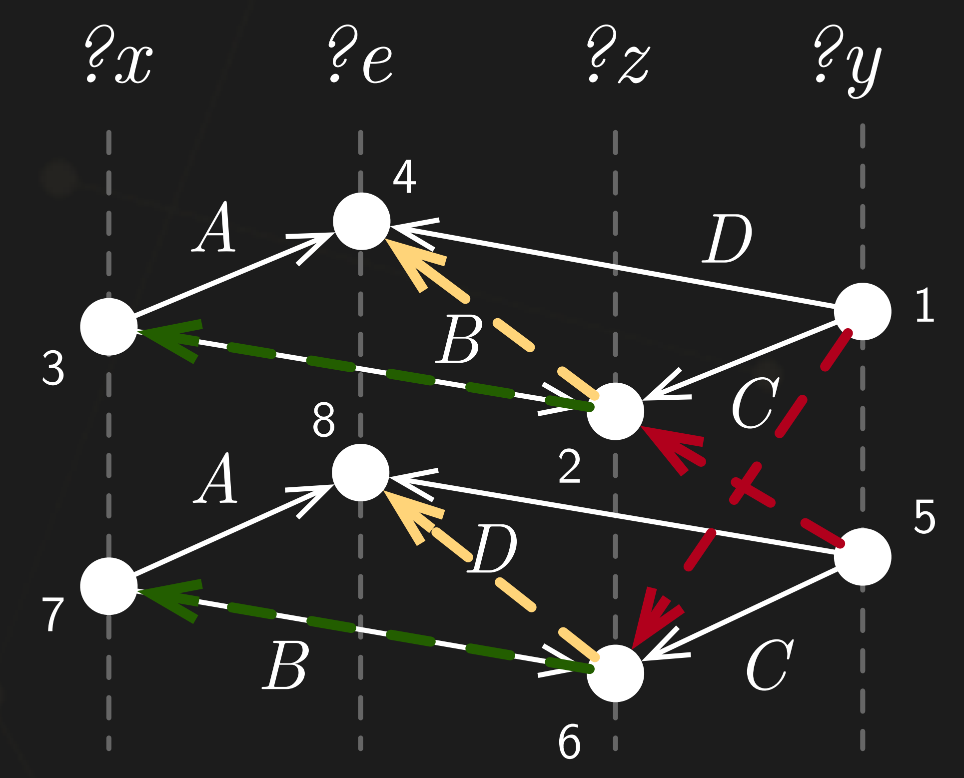
Triangulator

We triangulate a query graph to reduce the AG further:

- whenever a chord intersects a query edge (*-deep, zig-zag plan) or another chord (bushy plan), a **seal** happens
- a seal triggers the **edge burn-back** which removes the chord edges which don't participate in the final embeddings, eventually removing "spurious" edges in the AG, on cascade



Query graph



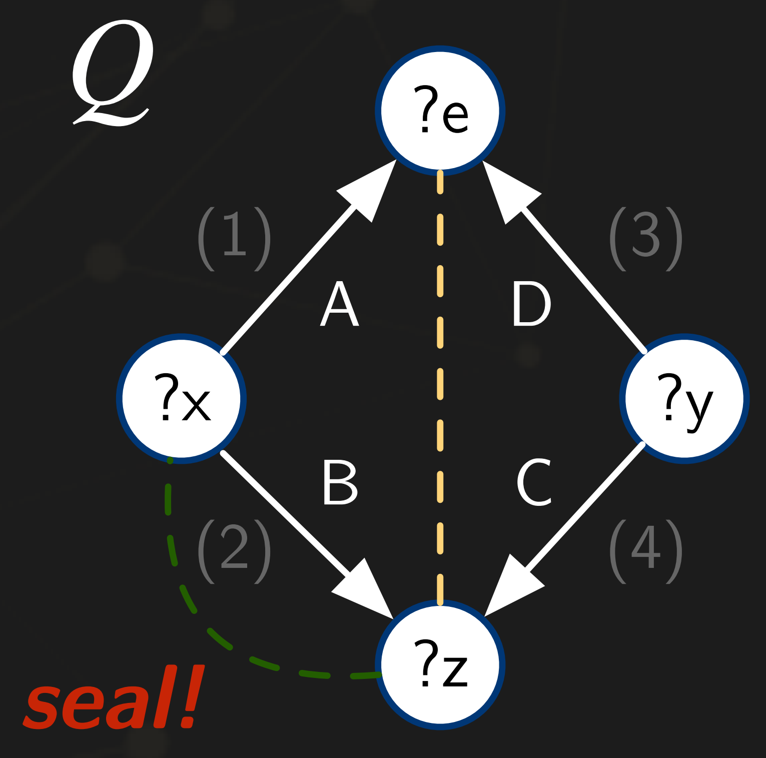
Answer graph

edge burn-back!

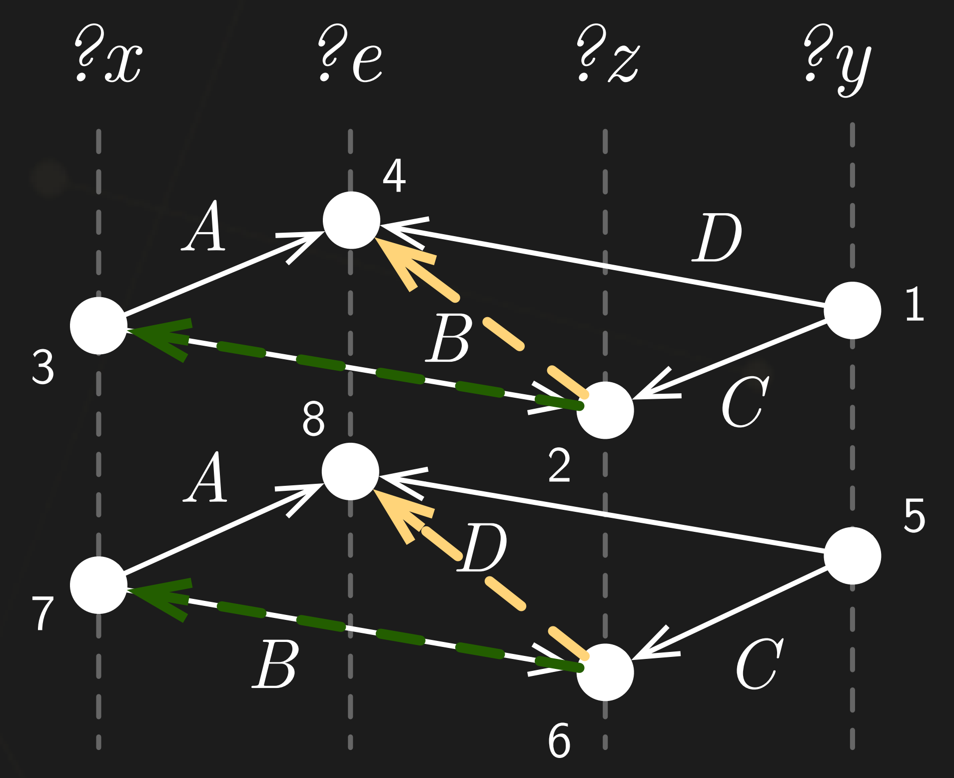
Triangulator

We triangulate a query graph to reduce the AG further:

- whenever a chord intersects a query edge (*-deep, zig-zag plan) or another chord (bushy plan), a **seal** happens
- a seal triggers the **edge burn-back** which removes the chord edges which don't participate in the final embeddings, eventually removing "spurious" edges in the AG, on cascade



Query graph



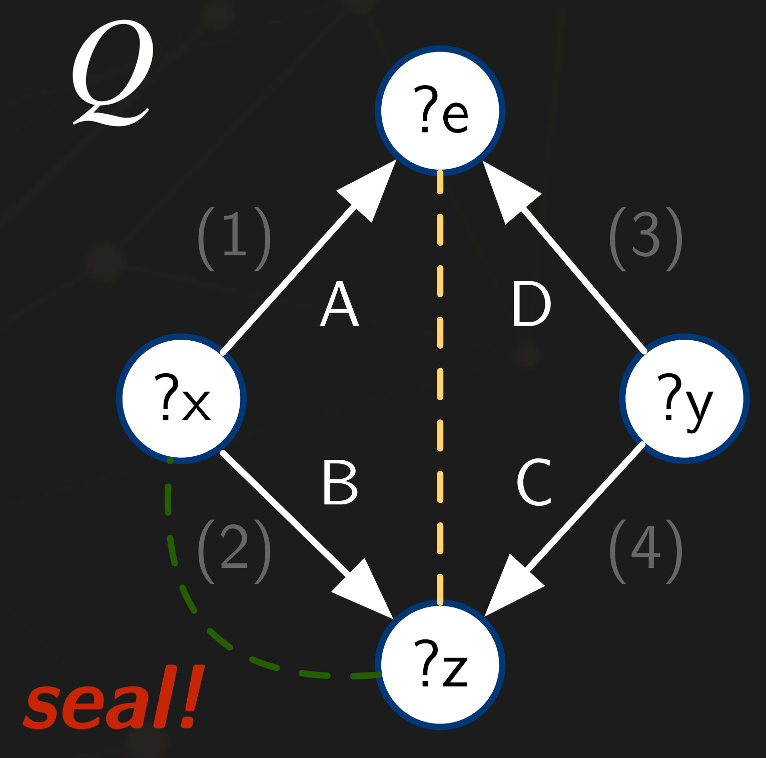
Answer graph

edge burn-back!

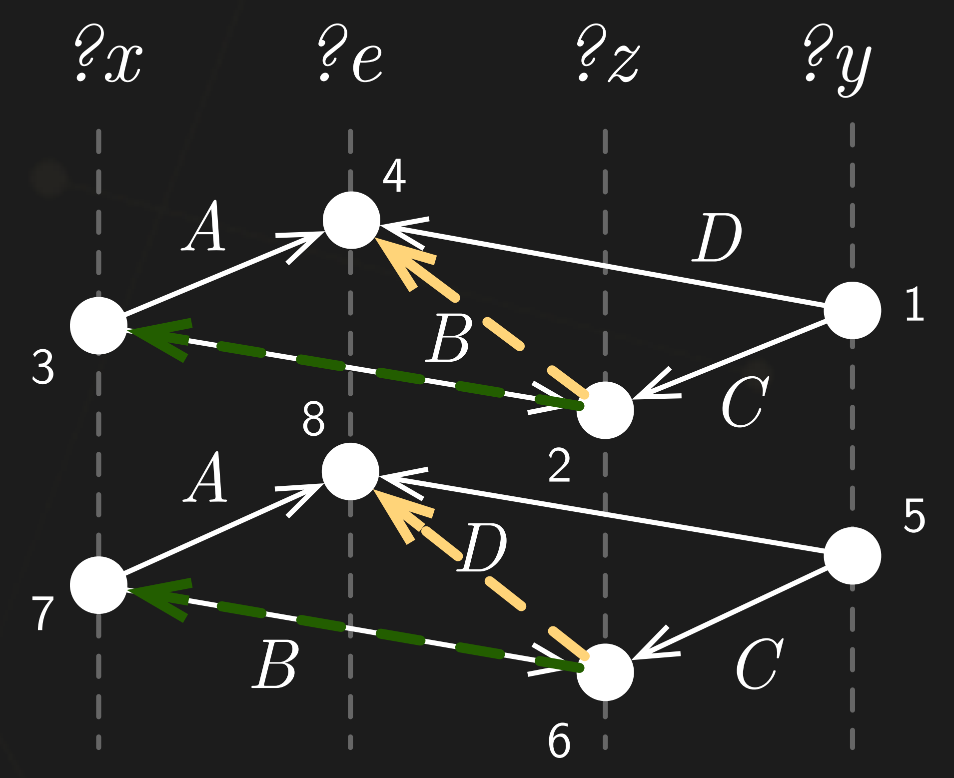
Triangulator

We triangulate a query graph to reduce the AG further:

- there are many different ways to triangulate the query graph
- similar to node burn-back, a DP cost-based enumeration is used to decide the best way to triangulate



Query graph



Answer graph

edge burn-back!

Ideal answer graph

We call an answer graph ideal if it contains only those edges which participate in at least one final embedding.

Theorem: Edge burn-back results in an ideal AG for **cyclic** graph CQs with treewidth of 2 in cascade of at most $O(|Q|)$.

Pf.:

- triangulation corresponds to a tree decomposition of the query graph (with a max. treewidth = 2)
- similar to node burn-back, the seal on cascade generates a semi-join program which contains (in correct order) a valid bounded full reducer program produced by the GYO algorithm ran on the tree decomposition
- this guarantees no dangling tuples in the materialized triangles
- with easy book-keeping, we can remove the corresponding edges from the binary edge relations to guarantee no dangling tuples there

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Ideal answer graph

We call an answer graph ideal if it contains only those edges which participate in at least one final embedding.

We can handle queries with higher treewidth graphs, but this requires more materialization to produce the ideal AG or using fix-point cascade

- Ultimately, this is a cost-based decision whether the extra materialization is worth the effort
- 99% of queries in practice are near acyclic

Experiments

Implementation. WireFrame is implemented on top of PostgreSQL

- Edgifier in the first phase outputs an optimal left-deep tree plan
- For defactorizer, we use a greedy approach to generate a tree plan
- Node burn-back procedure is implemented via procedural SQL

Fuller-featured implementation is available in ...

Ag
AvantGraph

TU/e
EINDHOVEN UNIVERSITY OF TECHNOLOGY

Experiments

CQ_S	**Snowflake-shaped Queries** (1/2/3/4/5/6/7/8/9)	PG	WF	VT	MD	NJ	|iAG|	|Embeddings|
1	diedIn/influences/actedIn/owns/wasCreatedOnDate/actedIn/created/hasDuration/wasCreatedOnDate	66	**4**	*	*	*	1660	2931986
2	hasChild/influences/actedIn/actedIn/wasBornIn/created/actedIn/hasDuration/wasCreatedOnDate	63	**3**	246	*	*	993	2847184
3	isCitizenOf/influences/actedIn/exports/wasCreatedOnDate/actedIn/created/hasDuration/wasCreatedOnDate	37	**7**	287	*	*	1140	2670339
4	isMarriedTo/influences/actedIn/actedIn/wasBornOnDate/created/actedIn/hasDuration/wasCreatedOnDate	59	**3**	286	*	*	3317	2569017
5	isMarriedTo/influences/actedIn/wasBornOnDate/isMarriedTo/actedIn/created/wasCreatedOnDate/hasDuration	57	**17**	268	*	*	3580	2127992
6	isMarriedTo/influences/actedIn/hasGender/isMarriedTo/actedIn/created/hasDuration/wasCreatedOnDate	57	**12**	268	*	*	3580	2123951
7	diedIn/isMarriedTo/actedIn/owns/wasCreatedOnDate/actedIn/created/hasDuration/wasCreatedOnDate	30	**15**	266	*	*	10761	2111948
8	isMarriedTo/influences/actedIn/hasFamilyName/isMarriedTo/created/actedIn/hasDuration/wasCreatedOnDate	32	**14**	261	*	*	3580	2102297
9	isMarriedTo/hasChild/actedIn/wroteMusicFor/created/created/actedIn/hasDuration/wasCreatedOnDate	35	**9**	256	*	*	7330	1786626
10	isMarriedTo/influences/actedIn/actedIn/created/created/directed/hasDuration/wasCreatedOnDate	39	**4**	237	*	*	3317	1533188

- The size of the AG is exceedingly smaller that the number of embeddings
    - For Q2, 3000X smaller
    - this indicates lots of M-M joins and multiplicative effect (also in the IR)

- AG achieves excellent performance on such queries
    - as it avoids redundant edge walks in the IR

Experiments

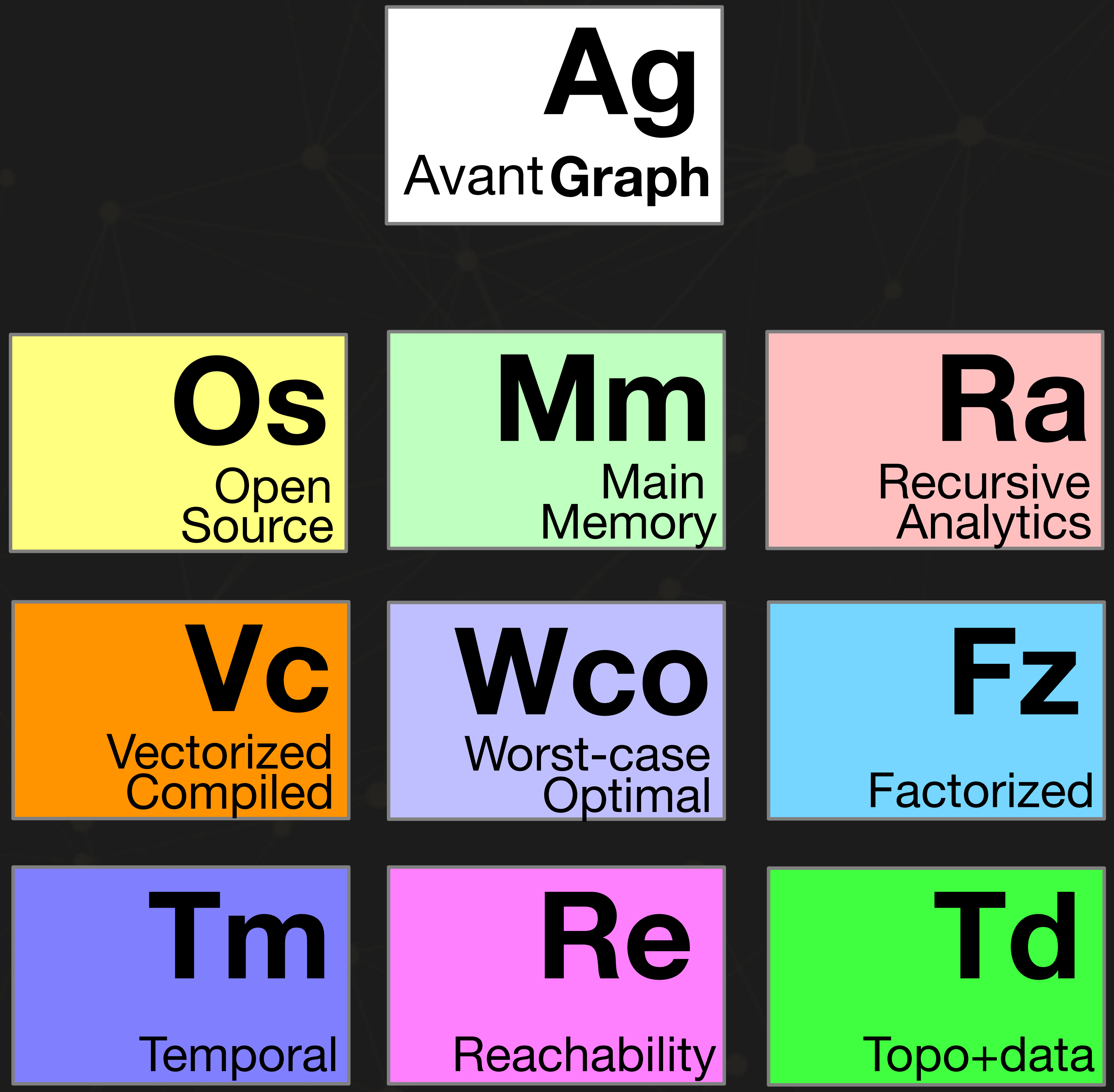
CQ_D	**Diamond-shaped Queries** (1/2/3/4)	PG	WF	VT	MD	NJ	|AG|	|Embeddings|
11	isLocatedIn/linksTo/isCitizenOf/livesIn	*	**39**	*	*	*	813311	59695937
12	livesIn/isCitizenOf/isLocatedIn/linksTo	*	**81**	*	*	*	833355	58785214
13	isCitizenOf/wasBornIn/linksTo/diedIn	*	**12**	*	*	297	132961	3141996
14	isCitizenOf/diedIn/linksTo/wasBornIn	*	**21**	*	*	296	251054	3124213
15	wasBornIn/isAffiliatedTo/linksTo/playsFor	*	**37**	*	*	*	470196	2310680
16	wasBornIn/playsFor/linksTo/isAffiliatedTo	*	**39**	*	*	*	471520	2299729
17	isConnectedTo/linksTo/extractionSource/byTransport	*	**33**	67	*	140	112040	1312372
18	created/rdfs:label/linksTo/isPreferredMeaningOf	*	264	**65**	203	130	772994	169380
19	linksTo/isPreferredMeaningOf/created/skos:prefLabel	*	114	**22**	111	135	766785	169324
20	diedIn/linksTo/wasBornIn/graduatedFrom	*	**12**	92	*	195	68720	106214

Employing only node burn-back does not guarantee the ideal AG

- The resulting AGs can be significantly larger that the ideal AG
- For this reason, WF employing only node burn-back was slower on some of the cyclic CQs
- Even so, the overall performance was quite good

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Thank you!



For more information, see our EDBT 21 paper:

- *Zahid Abul-Basher, Nikolay Yakovets, Parke Godfrey, Stanley Clark, and Mark Chignell.* “Answer Graph: Factorization Matters in Large Graphs”. Proceedings of EDBT21.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY