>>> PhD Defense

>>> On the foundations for the compilation of web data queries: optimization and distributed evaluation of SPARQL.

UNIVERSITÉ
Grenoble
Alpes

Date: 13 September, 2018

Defendant: Louis JACHIET

Directors: Nabil LAYAÏDA, Pierre GENEVÈS

Reviewers: Dario COLAZZO, Ioana MANOLESCU

Jury: Jérôme EUZENAT, Patrick VALDURIEZ

>>> PhD Defense

>>> On the foundations for the compilation of web data queries: optimization and distributed evaluation of SPARQL.

UNIVERSITÉ
Grenoble
Alpes

Date:	13 September, 2018
Defendant:	Louis JACHIET
Directors:	Nabil LAYAÏDA Pierre GENEVÈS
Reviewers:	Dario COLAZZO Ioana MANOLESCU
Jury:	Jérôme EUZENAT Patrick VALDURIEZ

What are the museums in Grenoble?

Who are they exposing?

WIKIPÉDIA
L'encyclopédie libre

Accueil
Portails thématiques
Article au hasard
Contact

Contribuer

Débuter sur Wikipédia
Aide
Communauté
Modifications récentes
Faire un don

Article | Discussion | Lire | Modifier | Plus | Rechercher dans Wikipédi

Musée de Grenoble

45° 11′ 40″ N, 5° 43′ 57″ E

Le **musée de Grenoble**, créé en 1798, est le principal musée d'art et d'Antiquités de la Ville de Grenoble, en Isère. Situé place Lavalette, à l'emplacement d'un ancien couvent des Franciscains édifié en 1218 et dont le site deviendra militaire à la fin du XVI^e siècle, il fait partie des premiers musées d'art français et conserve l'une des plus belles collections d'art ancien.

Musée de Grenoble

Informations générales

Toutes les tendances et mouvements de la peinture sont présents, tels le fauvisme avec des tableaux de Henri Matisse (8 peintures), André Derain, Albert Marquet, Raoul Dufy, Maurice de Vlaminck, Emile Othon Friesz (6 peintures), Jean Puy, Charles Camoin (*Nu à la chemise mauve*, acquis en 2012) Kees van Dongen, le cubisme avec Georges Braque, Albert Gleizes, André Lhote, Fernand Léger et Le Corbusier, l'école de Paris, représentée par Amedeo Modigliani, Pinchus Krémègne, Chaïm Soutine, Maurice Utrillo ainsi que Marc Chagall. Quatre peintures

The goals of a data standard are to:

* Encode data in a machine readable & processable way
* Allow exchange of data on the web
* Facilitate querying

RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.

RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.

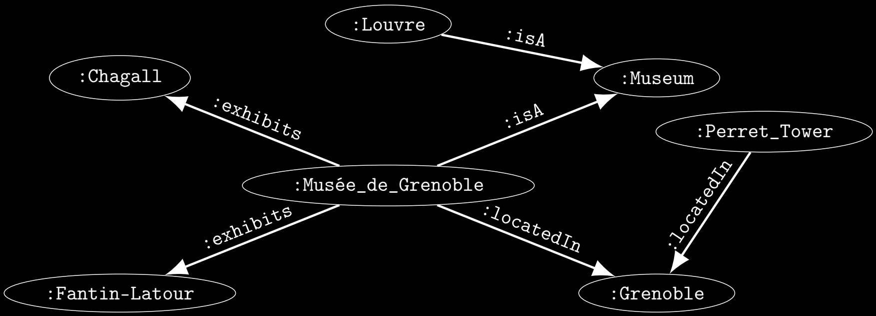
In N-Triples format:

subject	predicate	object	
:Musée_de_Grenoble	:isA	:Museum	.
:Musée_de_Grenoble	:locatedIn	:Grenoble	.
:Musée_de_Grenoble	:exhibits	:Chagall	.
:Musée_de_Grenoble	:exhibits	:Fantin-Latour	.
:Perret_Tower	:locatedIn	:Grenoble	.
:Louvre	:isA	:Museum	.

RDF: Resource Description Framework [RCM14]

In RDF, data is represented as entities and as statements expressing relationships between these entities.

Viewed as graphs:



Triple Pattern (TP)

A Triple Path is a triple (s, p, o) where s, p, o are constants or variables.

Triple Pattern (TP)

A Triple Path is a triple (s, p, o) where s, p, o are constants or variables.

"What are the museums?"

Triple Pattern (TP)

A Triple Path is a triple (s, p, o) where s, p, o are constants or variables.

"What are the museums?"

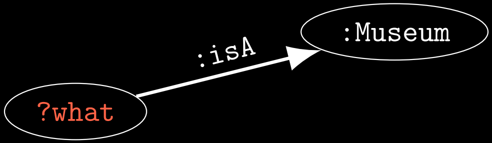
In SPARQL:

```
?what :isA :Museum .
```

Triple Pattern (TP)

A Triple Path is a triple (s, p, o) where s, p, o are constants or variables.

"What are the museums?"



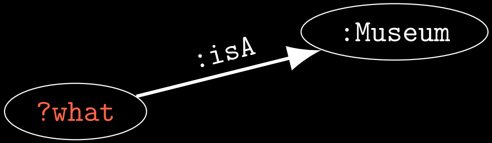
In SPARQL:

```
?what :isA :Museum .
```

Triple Pattern (TP)

A Triple Path is a triple (s, p, o) where s, p, o are constants or variables.

"What are the museums?"



In SPARQL:

```
?what :isA :Museum .
```

Solution:

(?What → :Musée_de_Grenoble)
(?What → :Louvre)

Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

"What are the museums in Grenoble exposing Chagall?"

Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

"What are the museums in Grenoble exposing Chagall?"

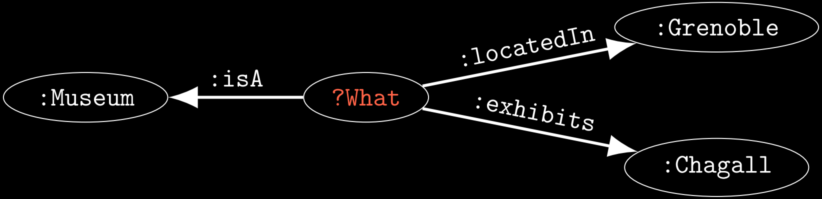
In SPARQL:

```
?what :isA :Museum .
?what :locatedIn :Grenoble .
?what :exhibits :Chagall .
```

Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”



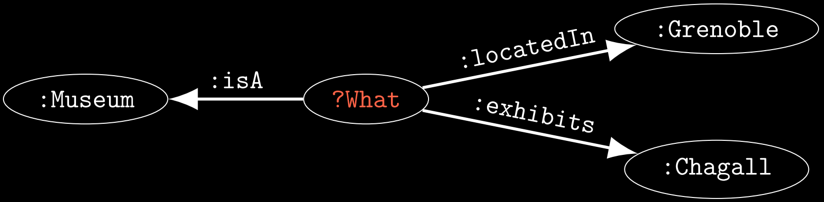
In SPARQL:

```
?what :isA :Museum .
?what :locatedIn :Grenoble .
?what :exhibits :Chagall .
```

Basic Graph Patterns (BGP)

Basic Graph Patterns are conjunctions of Triple Patterns.

“What are the museums in Grenoble exposing Chagall?”



In SPARQL:

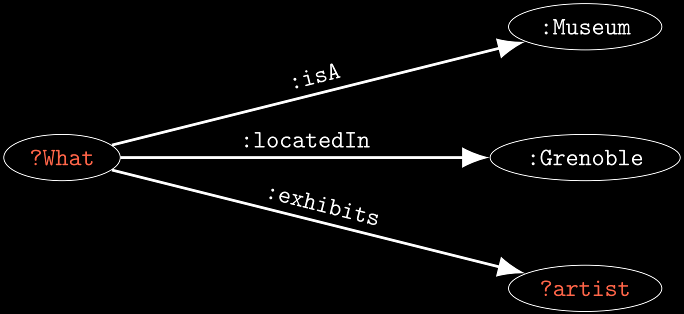
```
?what :isA :Museum .
?what :locatedIn :Grenoble .
?what :exhibits :Chagall .
```

Solution:

(?What → :Musée_de_Grenoble)

"What are the museums in Grenoble and who are they exposing?"

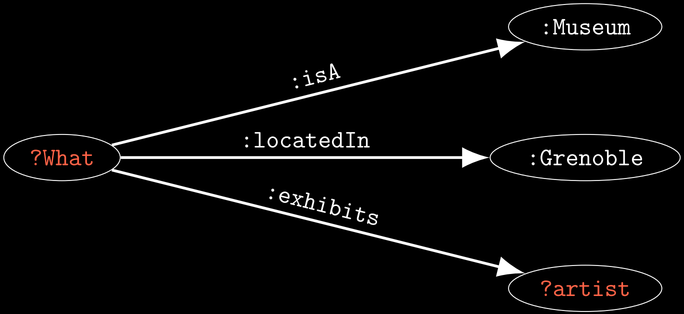
"What are the museums in Grenoble and who are they exposing?"



In SPARQL:

```
?what :isA :Museum .
?what :locatedIn :Grenoble .
?what :exhibits ?artist .
```

"What are the museums in Grenoble and who are they exposing?"



In SPARQL:

```
?what :isA :Museum .
?what :locatedIn :Grenoble .
?what :exhibits ?artist .
```

Solution:

(?What → :Musée_de_Grenoble;
?artist → :Chagall)
(?What → :Musée_de_Grenoble;
?artist → :Fantin-Latour)

Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns

* Conjunction

Basic Graph Patterns

Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns

* Conjunction

* Disjunction

Museums or Trekking paths

Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns

* Conjunction

* Disjunction

* Filters

Museums with artists born before 1900

Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns
* Conjunction
* Disjunction
* Filters
* Conditional optionals

Missing information

Other graph patterns in SPARQL 1.0 [PS+08]

* Triple Patterns
* Conjunction
* Disjunction
* Filters
* Conditional optionals
* Changing graphs, *etc.*

Novelties of SPARQL 1.1 [HSP13]

* Expressions

Compute the age of monuments

Novelties of SPARQL 1.1 [HSP13]

* Expressions
* Minus, Exists

Cities with museums but without operas

Novelties of SPARQL 1.1 [HSP13]

* Expressions
* Minus, Exists
* Group by & Aggregation

Count the number of museums per city

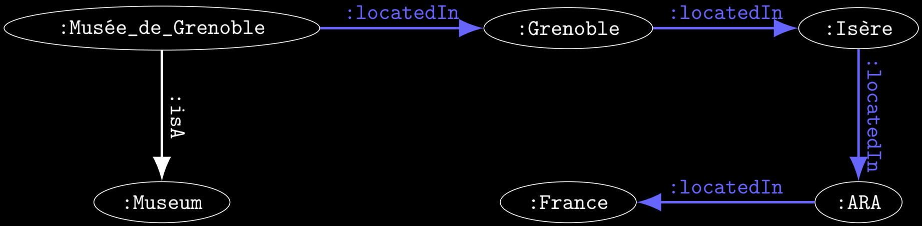
Novelties of SPARQL 1.1 [HSP13]

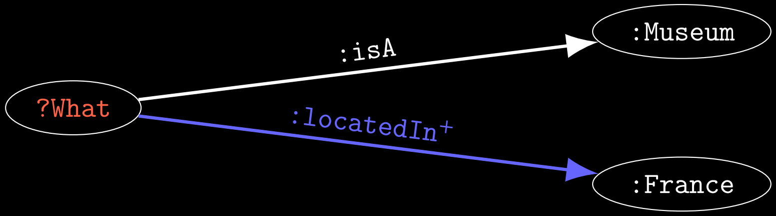
* Expressions
* Minus, Exists
* Group by & Aggregation

  Count the number of museums per city
* Property Paths

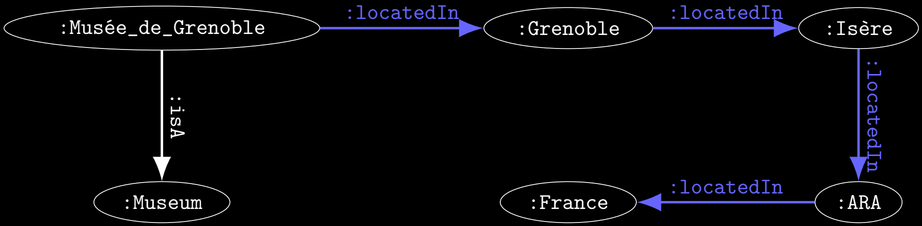
“What are the museums in France?”

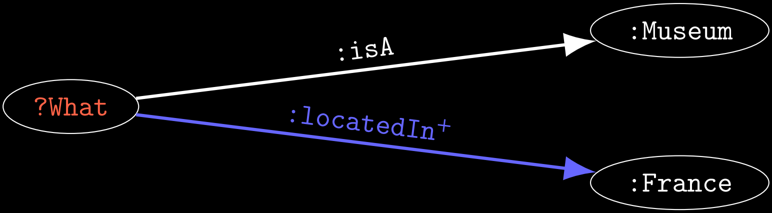
"What are the museums in France?"





"What are the museums in France?"



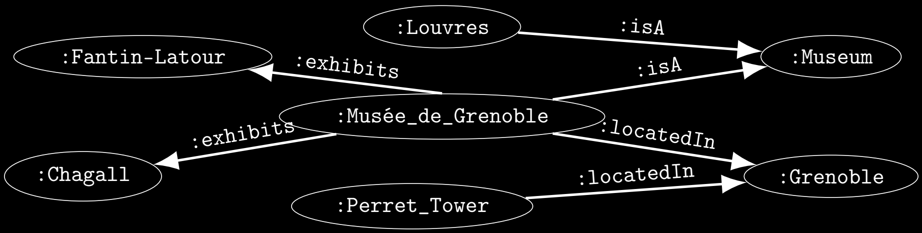


"What are the museums in France?"

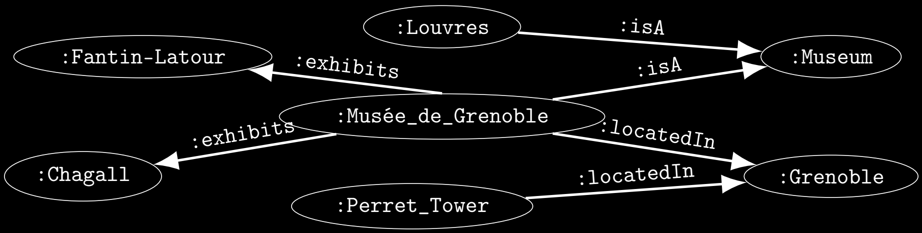
Property Path (PP)

A Property Path is a triple (s, r, o) where r is a path expression.

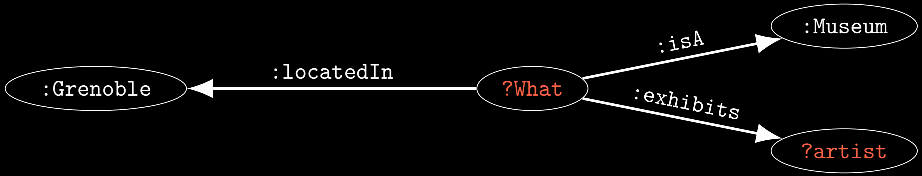
“What are the museums in Grenoble and who are they exposing?”

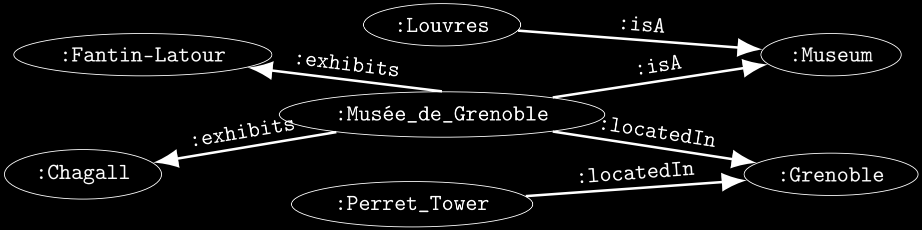


"What are the museums in Grenoble and who are they exposing?"

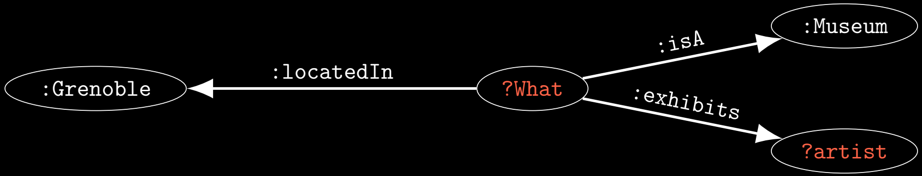


"What are the museums in Grenoble and who are they exposing?"

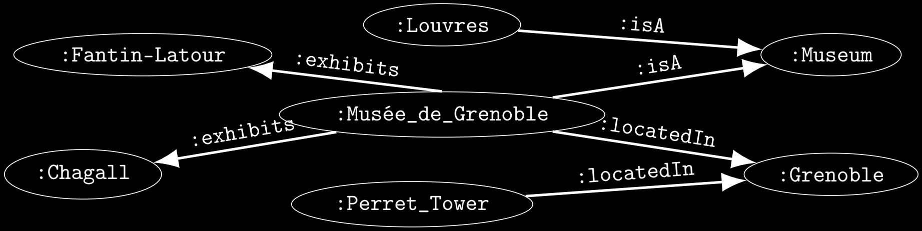




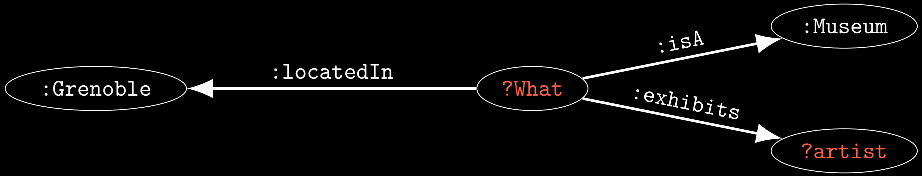
"What are the museums in Grenoble and who are they exposing?"



(?What → :Musée_de_Grenoble; ?artist → :Chagall)
(?What → :Musée_de_Grenoble; ?artist → :Fantin-Latour)



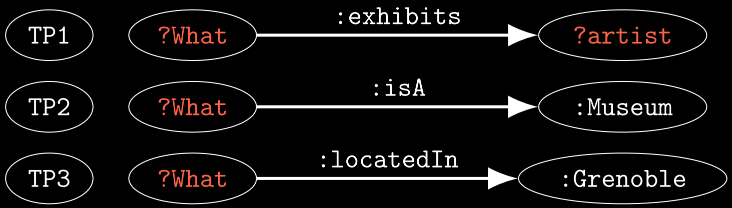
"What are the museums in Grenoble and who are they exposing?"



(?What → :Musée_de_Grenoble; ?artist → :Chagall)
(?What → :Musée_de_Grenoble; ?artist → :Fantin-Latour)

$O\left(\#nodes^{\#variables}\right)$ *checks!*

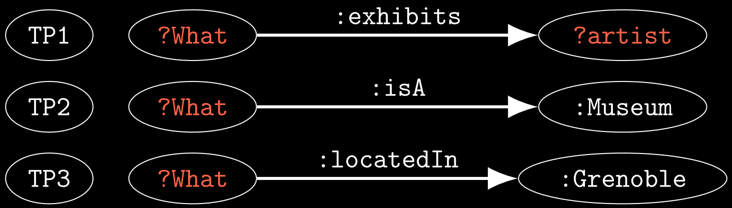
Compute solution to each individual TP



Combine individual solutions

$$(TP1 \bowtie TP2) \bowtie TP3$$

Compute solution to each individual TP



$O(\#edges)$ *per TP*

Combine individual solutions

$$(TP1 \bowtie TP2) \bowtie TP3$$

$O(|A| + |B| + |A \bowtie B|)$ *per* $A \bowtie B$

$(BGP, \bowtie)$ as an algebraic structure:

* $\bowtie$ is associative

$$TP1 \bowtie (TP2 \bowtie TP3) = (TP1 \bowtie TP2) \bowtie TP3$$

* $\bowtie$ is commutative

$$TP1 \bowtie TP2 = TP2 \bowtie TP1$$

An optimization method:

Generate all the equivalent terms, run the most efficient.

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatchs in the semantics

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatchs in the semantics
* Poor optimization of Property Paths

The relational algebra [Cod70]

* Base relations
* Combined through a set of operators ($\bowtie$, $\cup$, σ, etc.)

Optimization of relational languages

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Problems:

* Not specifically for graphs
* Mismatchs in the semantics
* Poor optimization of Property Paths

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

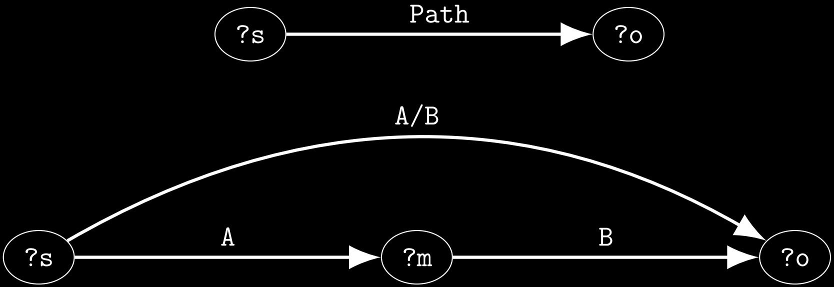
The μ-algebra:

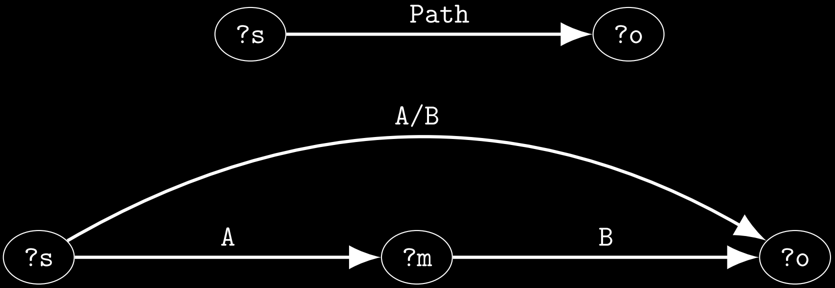
* a variation of the relational algebra
* equipped with fixpoints
* matches the SPARQL semantics

φ	::=		formula
	|	$\varphi_1 \cup \varphi_2$	union
	|	$\varphi_1 \setminus\setminus \varphi_2$	normal minus
	|	$\varphi_1 - \varphi_2$	set minus
	|	$\varphi_1 \setminus \varphi_2$	strict minus
	|	$\varphi_1 ⟕ \varphi_2$	left-join
	|	$\varphi_1 \bowtie \varphi_2$	join
	|	$\rho_a^b(\varphi)$	column exchange (or rename)
	|	$\pi_a(\varphi)$	projection
	|	$\beta_a^b(\varphi)$	column multiplying
	|	$\theta(\varphi, g : \mathcal{C} \to \mathcal{D})$	apply a function to mappings
	|	$\Theta(\varphi, g, \mathcal{C}, \mathcal{D})$	reduce
	|	$\sigma_{filter}(\varphi)$	row filtering
	|	$\mu(X = \varphi)$	fixpoint
	|	let $(X = \varphi)$ in ψ	let-binder
	|	X	variable
	|	$\emptyset$	no mapping
	|	$\lvert c_1 \to v_1, \ldots, c_n \to v_n \rvert$	a mapping

φ ::=		formula
|	$\varphi_1 \cup \varphi_2$	union
|	$\varphi_1 \setminus\!\setminus \varphi_2$	normal minus
|	$\varphi_1 - \varphi_2$	set minus
|	$\varphi_1 \setminus \varphi_2$	strict minus
|	$\varphi_1 ⟕ \varphi_2$	left-join
|	$\varphi_1 \bowtie \varphi_2$	join
|	$\rho_a^b(\varphi)$	column exchange (or rename)
|	$\pi_a(\varphi)$	projection
|	$\beta_a^b(\varphi)$	column multiplying
|	$\theta(\varphi, g : \mathcal{C} \to \mathcal{D})$	apply a function to mappings
|	$\Theta(\varphi, g, \mathcal{C}, \mathcal{D})$	reduce
|	$\sigma_{filter}(\varphi)$	row filtering
|	$\mu(X = \varphi)$	fixpoint
|	let $(X = \varphi)$ in ψ	let-binder
|	X	variable
|	$\emptyset$	no mapping
|	$\lvert c_1 \to v_1, \ldots, c_n \to v_n \rvert$	a mapping


Path
?s
?o


?s
Path
?o
A/B
?s
A
?m
B
?o



$$Tr(A/B) \quad = \quad \pi_m \left(\rho_o^m \left(Tr(A) \right) \bowtie \rho_s^m \left(Tr(B) \right) \right)$$

$$A^* = \texttt{Empty Path } \mathit{or} \texttt{ Path of } A^*/A$$

$$A^* = \texttt{Empty Path } \textit{or} \texttt{ Path of } A^*/A$$

$$Tr(A^*) \quad = \quad EmptyPath \quad \cup \quad Tr(A^*)/A$$

$$A^* = \texttt{Empty Path } or \texttt{ Path of } A^*/A$$

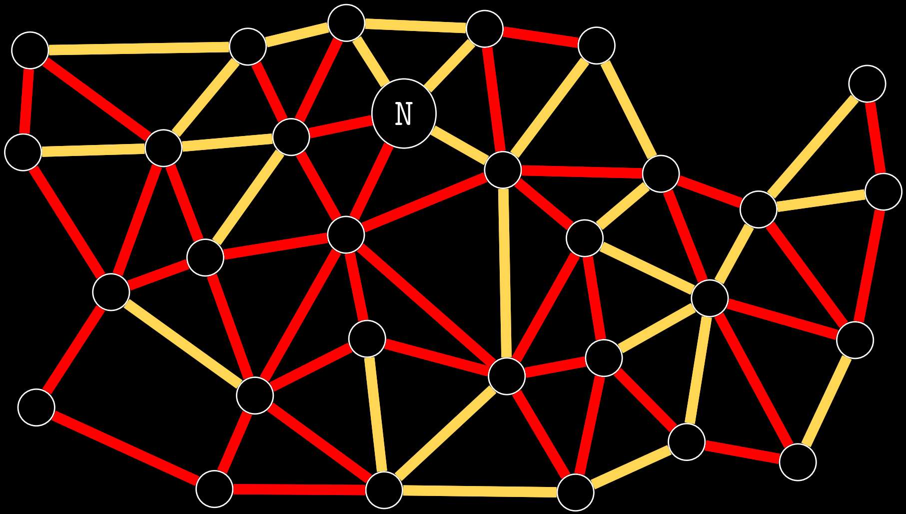
$$\begin{aligned} Tr(A^*) &= EmptyPath \cup Tr(A^*)/A \\ &= \mu\Big(X = EmptyPath \cup X/A\Big) \end{aligned}$$

$$A^* = \texttt{Empty Path } or \texttt{ Path of } A^*/A$$

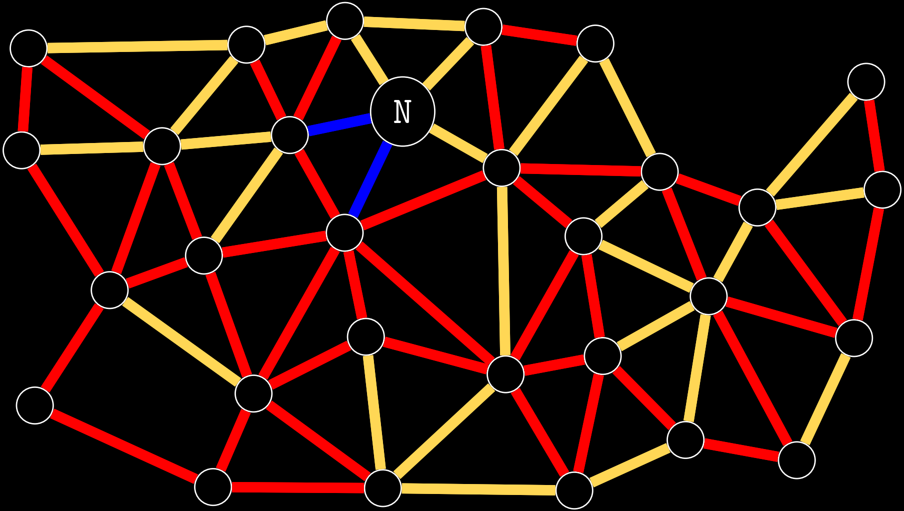
$$\begin{aligned}
Tr(A^*) &= EmptyPath \cup Tr(A^*)/A \\
&= \mu\Big(X = EmptyPath \cup X/A\Big) \\
&= \mu\Big(X = \beta_s^o(AllNodes) \cup \pi_m\left(\rho_o^m(X) \bowtie \rho_s^m(Tr(A))\right)\Big)
\end{aligned}$$

:N :Red/:Yellow* ?o

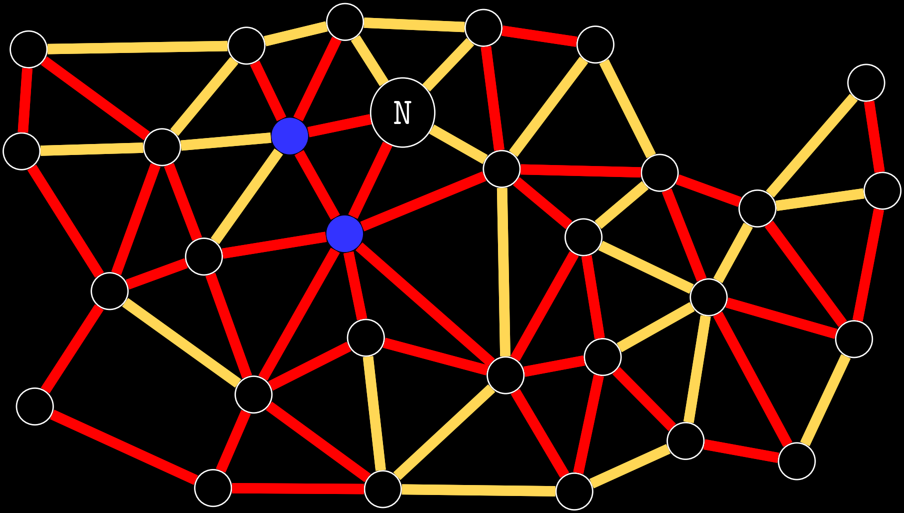
:N :Red/:Yellow* ?o



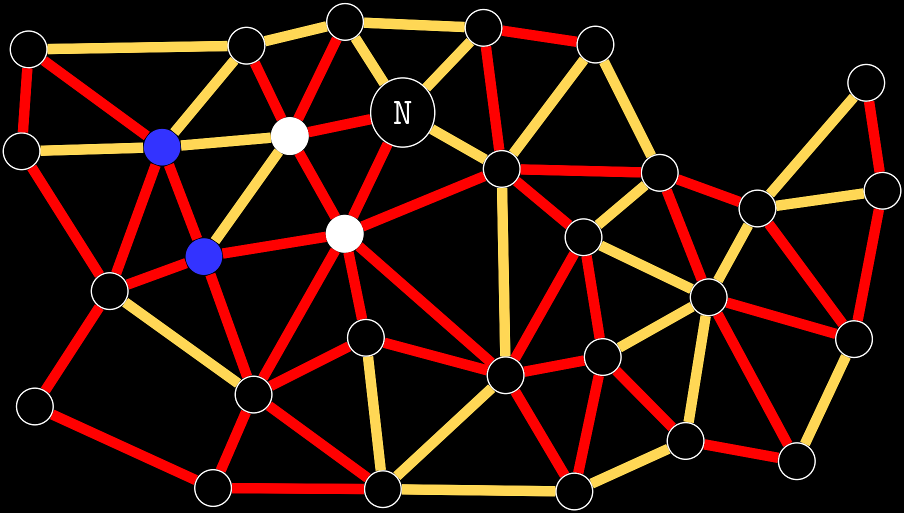
:N :Red/:Yellow* ?o



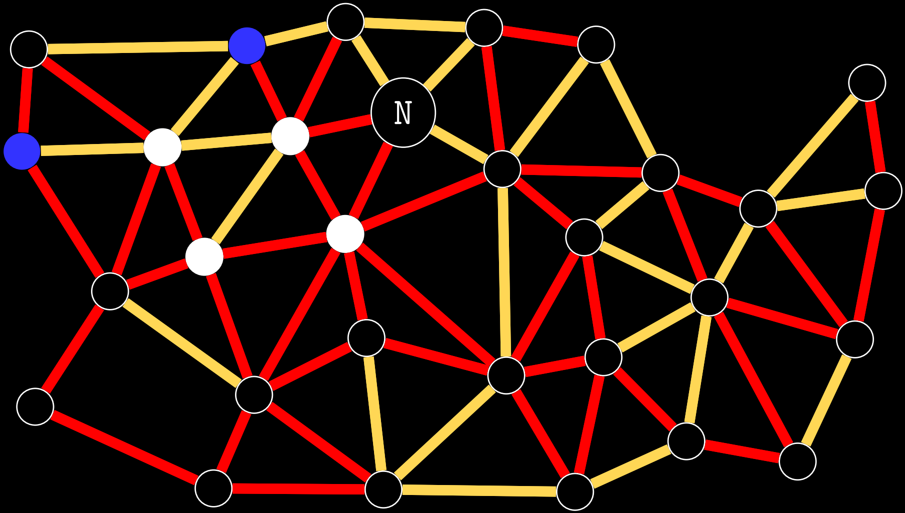
:N :Red/:Yellow* ?o



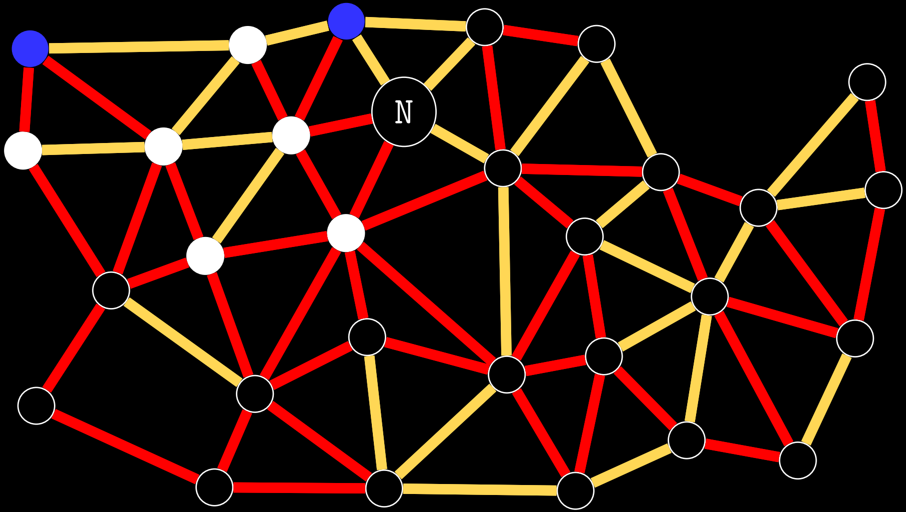
:N :Red/:Yellow* ?o



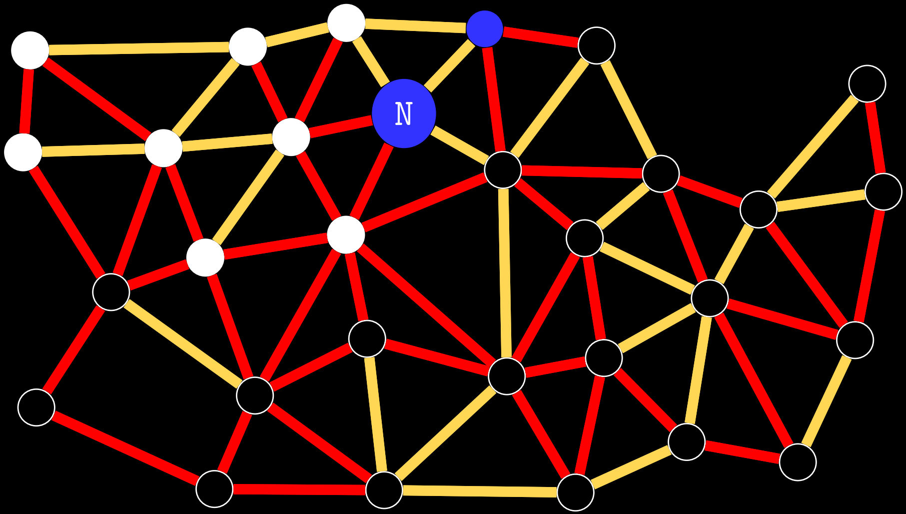
:N :Red/:Yellow* ?o



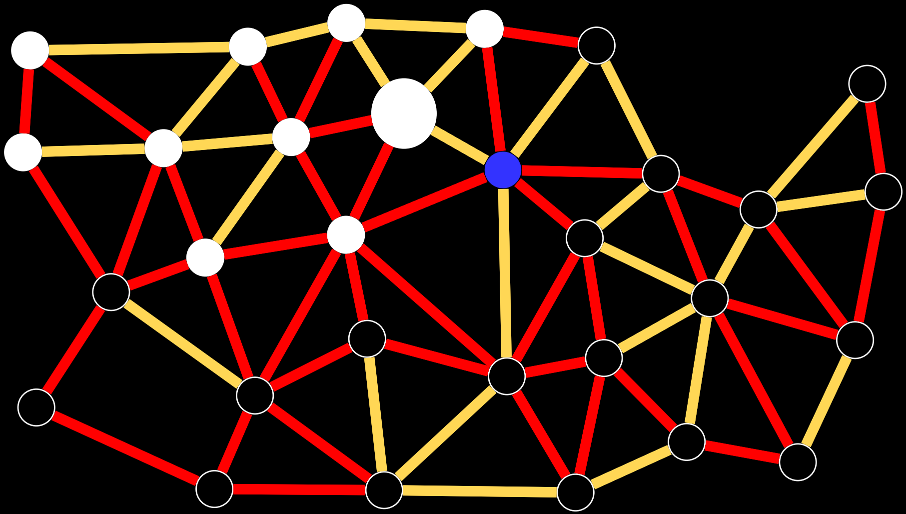
:N :Red/:Yellow* ?o



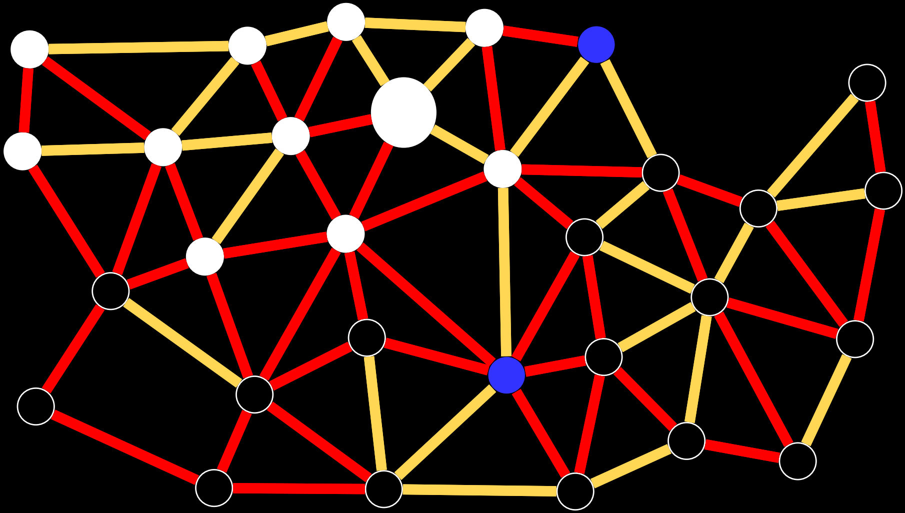
:N :Red/:Yellow* ?o



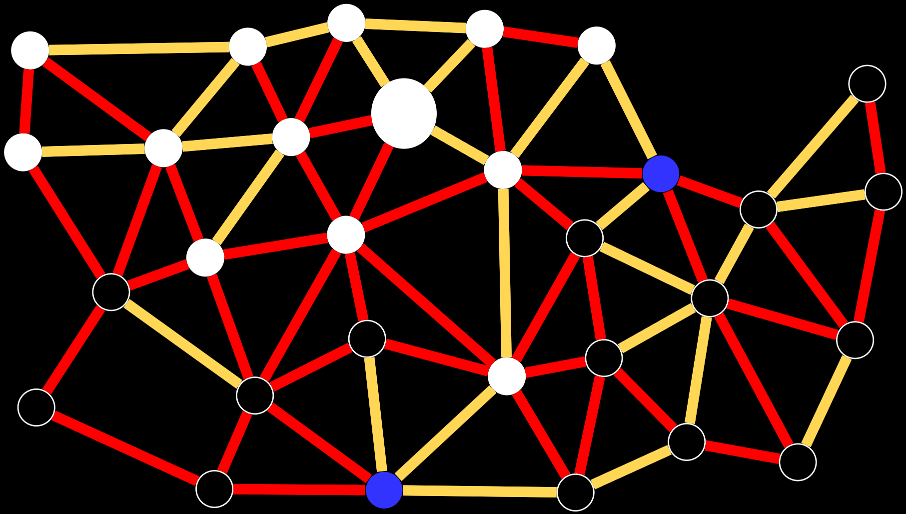
:N :Red/:Yellow* ?o



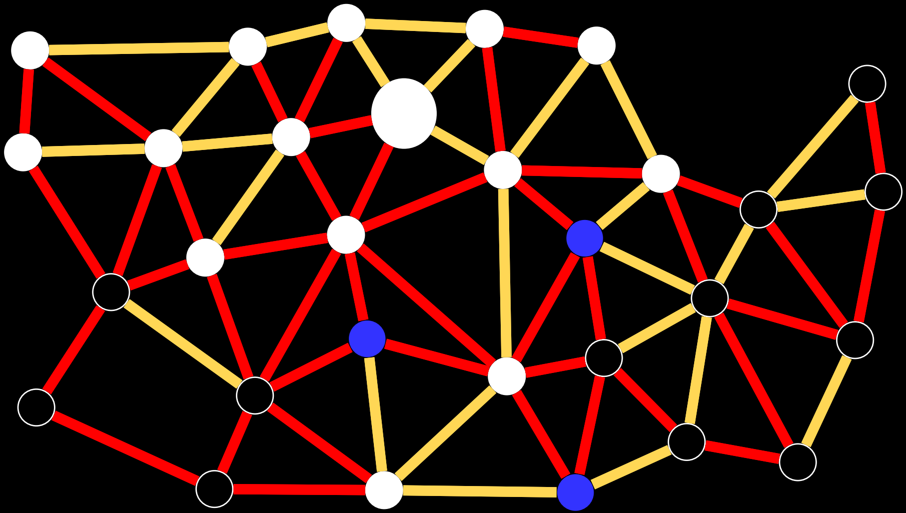
:N :Red/:Yellow* ?o



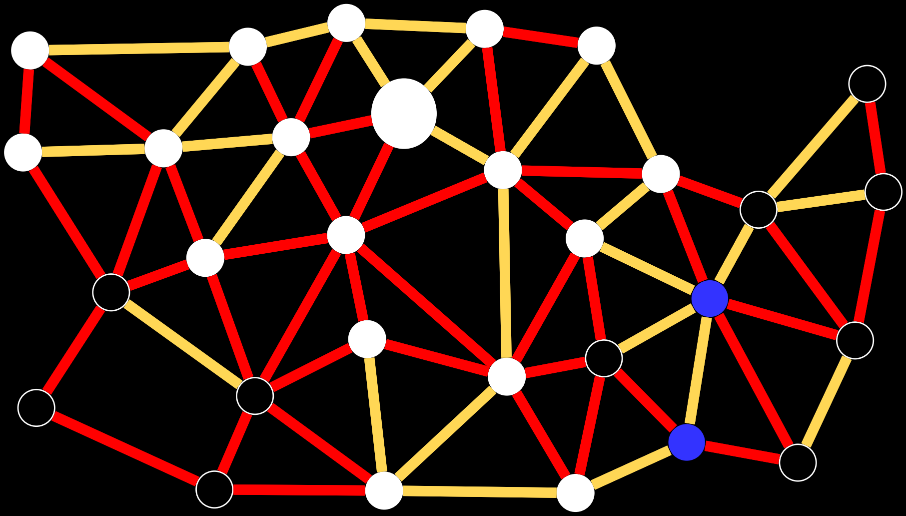
:N :Red/:Yellow* ?o



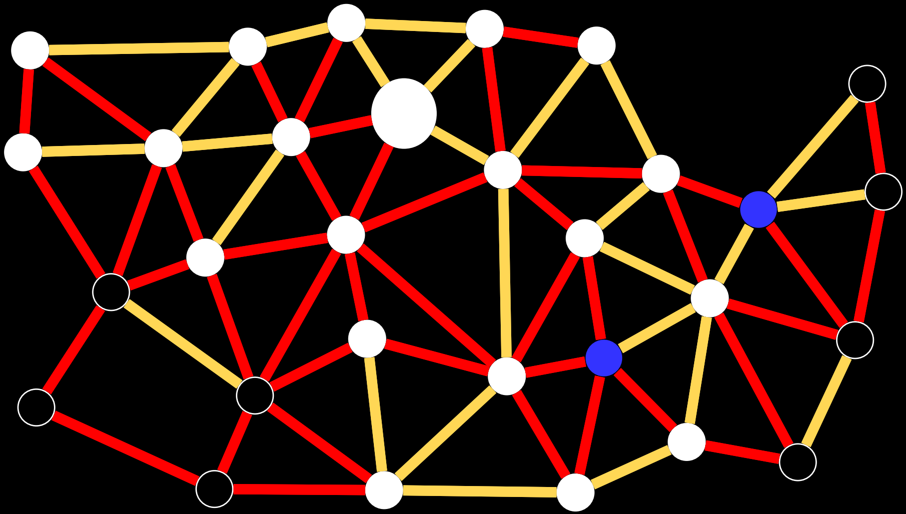
:N :Red/:Yellow* ?o



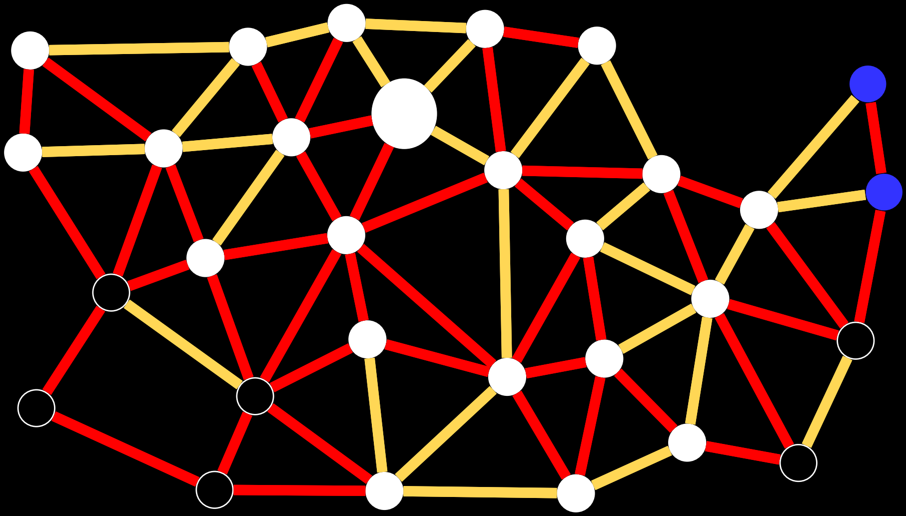
:N :Red/:Yellow* ?o



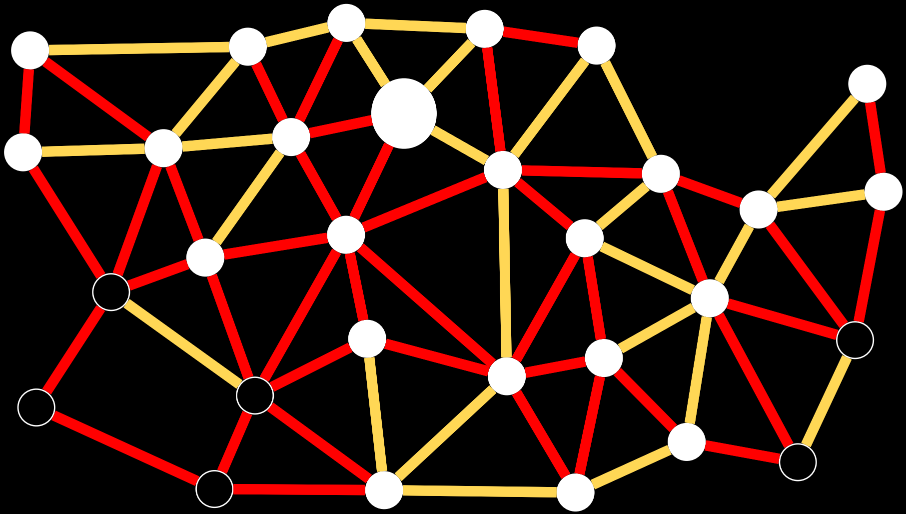
:N :Red/:Yellow* ?o



:N :Red/:Yellow* ?o



:N :Red/:Yellow* ?o



Rewrite rules for fixpoints

Rewrite rules for fixpoints

* pushing filters?

$$\sigma_{filter}(\mu(X = \varphi)) \stackrel{?}{=} \mu\left(X = \sigma_{filter}(\varphi)\right)$$

Rewrite rules for fixpoints

* pushing filters?

$$\sigma_{filter}(\mu(X = \varphi)) \stackrel{?}{=} \mu\big(X = \sigma_{filter}(\varphi)\big)$$

* return fixpoints?

Rewrite rules for fixpoints

* pushing filters?

$$\sigma_{filter}(\mu(X = \varphi)) \stackrel{?}{=} \mu\left(X = \sigma_{filter}(\varphi)\right)$$

* return fixpoints?

* pushing joins?

$$\psi \bowtie \mu(X = \varphi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi)$$

Rewrite rules for fixpoints

* pushing filters?

$$\sigma_{filter}(\mu(X = \varphi)) \stackrel{?}{=} \mu\left(X = \sigma_{filter}(\varphi)\right)$$

* return fixpoints?

* pushing joins?

$$\psi \bowtie \mu(X = \varphi) \stackrel{?}{=} \mu(X = \psi \bowtie \varphi)$$

* pushing projections?

$$\pi_p(\mu(X = \varphi)) \stackrel{?}{=} \mu\left(X = \pi_p(\varphi)\right)$$

Rewrite rules for fixpoints

* pushing filters?

$$\sigma_{filter}\left(\mu(X=\varphi)\right) \stackrel{?}{=} \mu\left(X=\sigma_{filter}(\varphi)\right)$$

* return fixpoints?

* pushing joins?

$$\psi \bowtie \mu(X=\varphi) \stackrel{?}{=} \mu(X=\psi \bowtie \varphi)$$

* pushing projections?

$$\pi_p\left(\mu(X=\varphi)\right) \stackrel{?}{=} \mu\left(X=\pi_p(\varphi)\right)$$

* combine fixpoints?

$$\mu(X=\psi \cup \kappa) \bowtie \mu(X=\varphi \cup \xi) \stackrel{?}{=} \mu(X=\psi \bowtie \varphi \cup \xi \cup \kappa)$$

:N :Red/:Yellow* ?o

```
:N :Red/:Yellow* ?o
```

$$\pi_{?s}\left(\sigma_{?s=:N}\left(:\texttt{Red}/\mu\left(X = \beta^{o}_{s}\left(AllNodes\right) \cup X/:\texttt{Yellow}\right)\right)\right)$$

:N :Red/:Yellow* ?o

$$\pi_{?s}\left(\sigma_{?s=:N}\left(:\text{Red}/\mu\left(X = \beta_s^o\,(AllNodes) \cup X/:\text{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu\left(X = :\text{Red}/\beta_s^o\,(AllNodes) \cup X/:\text{Yellow}\right)\right)\right)$$

:N :Red/:Yellow* ?o

$$\pi_{?s}\left(\sigma_{?s=:N}\left(:\mathtt{Red}/\mu\left(X=\beta^{o}_{s}\left(AllNodes\right)\cup X/:\mathtt{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu\left(X=:\mathtt{Red}/\beta^{o}_{s}\left(AllNodes\right)\cup X/:\mathtt{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu(X=:\mathtt{Red}\cup X/:\mathtt{Yellow})\right)\right)$$

:N :Red/:Yellow* ?o

$$\pi_{?s}\left(\sigma_{?s=:N}\left(:\mathtt{Red}/\mu\left(X=\beta^{o}_{s}\left(AllNodes\right)\cup X/:\mathtt{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu\left(X=:\mathtt{Red}/\beta^{o}_{s}\left(AllNodes\right)\cup X/:\mathtt{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu(X=:\mathtt{Red}\cup X/:\mathtt{Yellow})\right)\right)$$

$$\pi_{?s}\left(\mu\left(X=\sigma_{?s=:N}\left(:\mathtt{Red}\right)\cup X/:\mathtt{Yellow}\right)\right)$$

:N :Red/:Yellow* ?o

$$\pi_{?s}\left(\sigma_{?s=:N}\left(:\text{Red}/\mu\left(X=\beta_s^o(AllNodes)\cup X/:\text{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}\left(\mu\left(X=:\text{Red}/\beta_s^o(AllNodes)\cup X/:\text{Yellow}\right)\right)\right)$$

$$\pi_{?s}\left(\sigma_{?s=:N}(\mu(X=:\text{Red}\cup X/:\text{Yellow}))\right)$$

$$\pi_{?s}\left(\mu\left(X=\sigma_{?s=:N}(:\text{Red})\cup X/:\text{Yellow}\right)\right)$$

$$\mu\left(X=\pi_{?s}\left(\sigma_{?s=:N}(:\text{Red})\right)\cup X/:\text{Yellow}\right)$$

Methods of evaluating Property Paths:

* Ad-hoc

  Automata, Waveguide[YGG15]

* Fixpoints

  Datalog, Recursive SQL

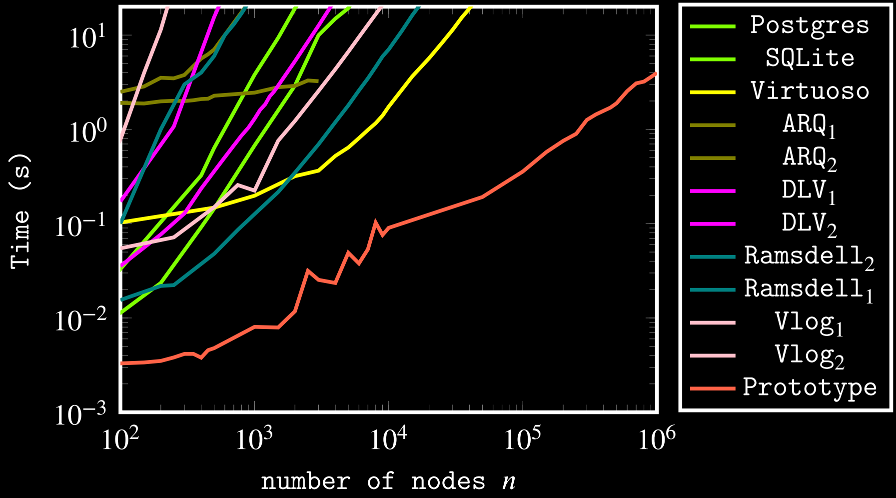


Figure: Time for : N :Red/:Yellow* $?o$ on n nodes

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

Cost model

Estimate the running time to evaluate a term.

$$Cost(A \bowtie B) = Cost(A) + Cost(B) + O(size(A \bowtie B))$$

Cost model

Estimate the running time to evaluate a term.

$$Cost(A \bowtie B) = Cost(A) + Cost(B) + O(size(A \bowtie B))$$

Cardinality estimation

Estimate the number of solutions to a term.

CardEst

A worst-case cardinality estimation with a new tool: summaries.

SPARQLGX

SPARQLGX is a distributed SPARQL query evaluator based on Apache Spark.

CardEst

A worst-case cardinality estimation with a new tool: summaries.

The hash join algorithm

$Map(Cogroup(A, B), iter)$

$$O((|A| + |B|) \times shuffle + |A \bowtie B|)$$

The broadcast join algorithm

$MapValues(A, f_B)$

$$O(|A| + |B| \times \#workers + |A \bowtie B|)$$

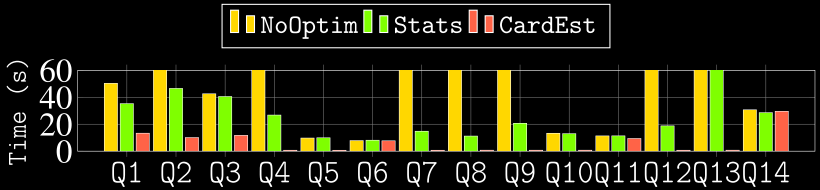


Figure: LUBM [GPH05] 10k (1,4 billions triples)

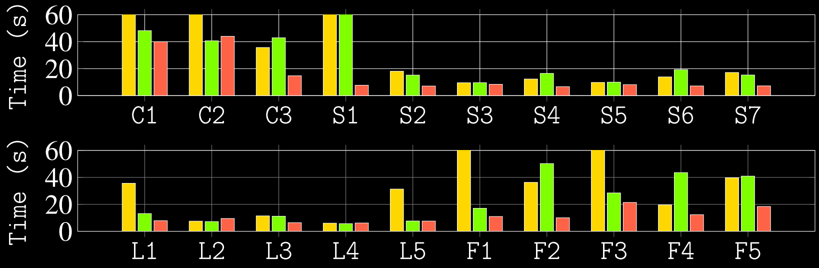
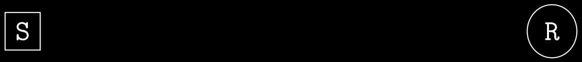


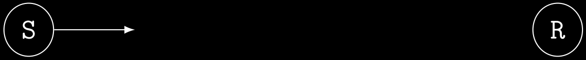
Figure: WatDiv [AHÖD14] 1k (140 millions triples)

1. Generate equivalent terms
2. Select an estimated most efficient
3. Execute it

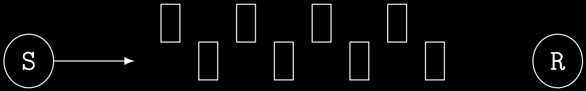
Streams are one-way communication channels



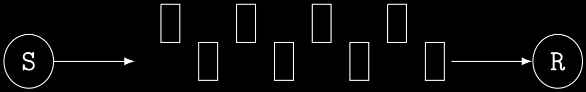
Streams are one-way communication channels



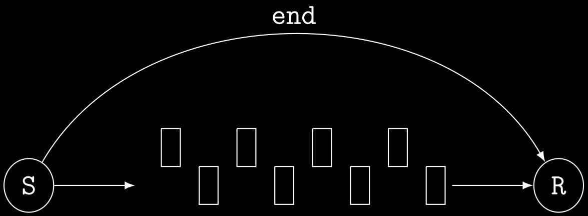
Streams are one-way communication channels



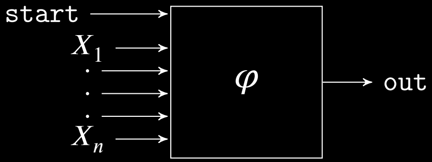
Streams are one-way communication channels



Streams are one-way communication channels



Execution of μ-algebra terms with streams



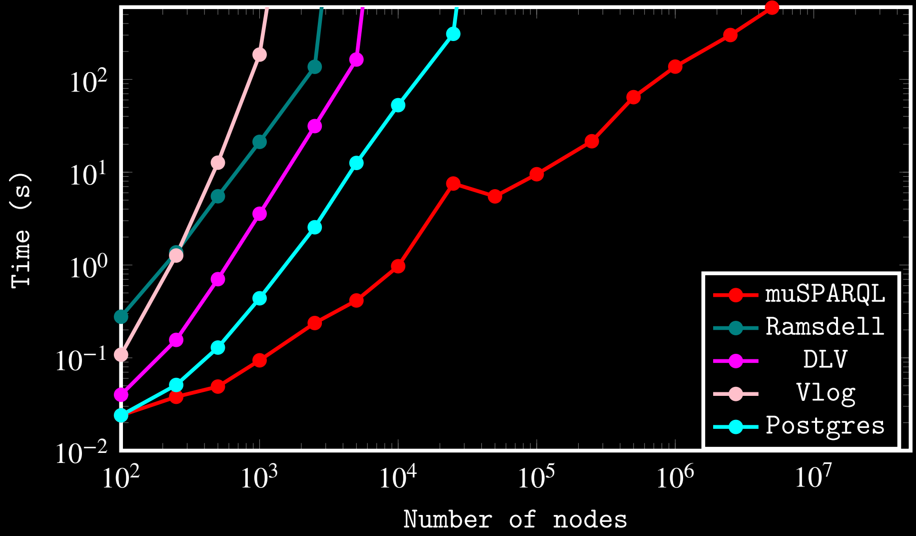


Figure: ?a (P1+)/(P5+) ?b.

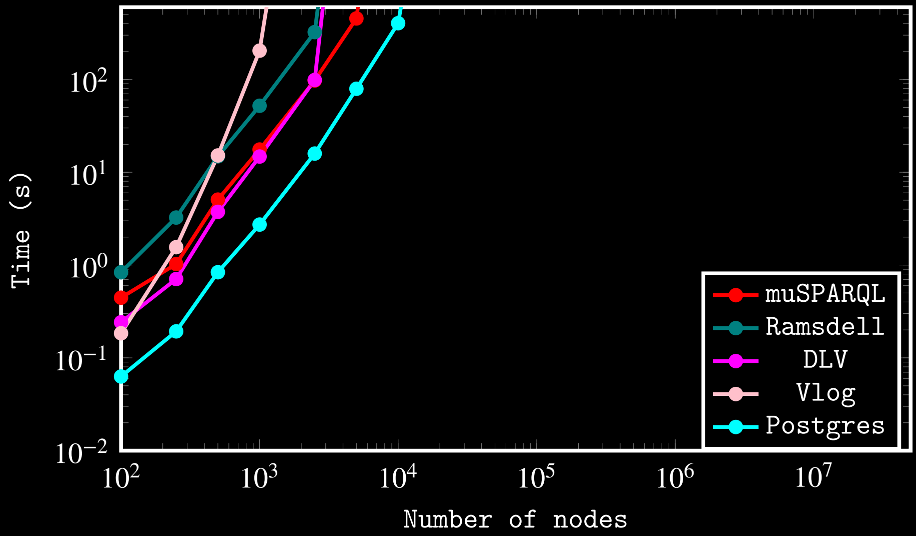


Figure: $?a \quad (P1+)/P2 \quad ?b \;. \quad ?b \quad P3+ \quad ?c\,.$

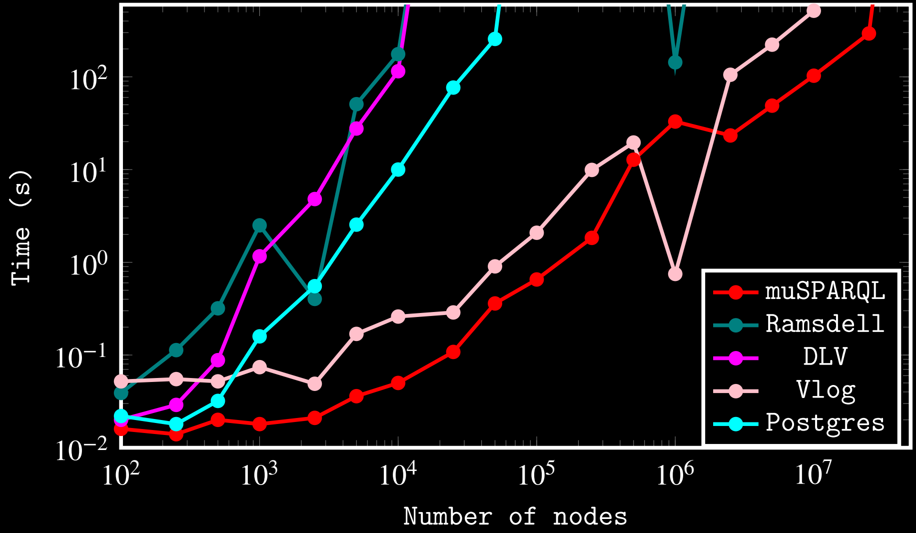


Figure: $N0\ P1/(P2+)\ ?a$

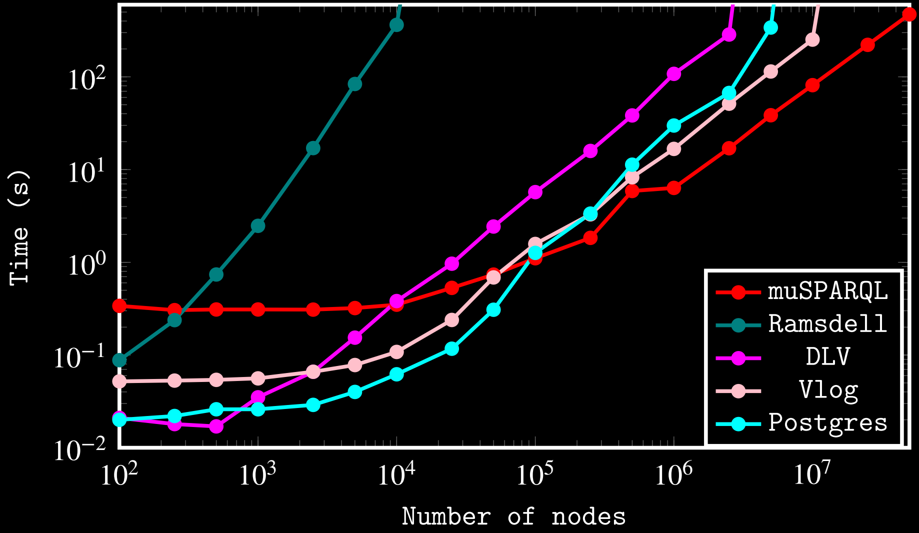


Figure: $?a \quad (P4+)/(P5+)/(P3+) \quad ?b$

>>> Conclusion / Contributions

Main contributions:

- **μ-Algebra** A new algebra with new efficient rewriting rules for fixpoints
- **muSPARQL** A prototype evaluator based on streams
- **CardEst** A new cardinality estimation technique
- **SPARQLGX** A distributed SPARQL query evaluator

Collaborations:

* Leaves enumeration technique.
* Distributed SPARQL query evaluators benchmarks.
* ProvSQL: provenance for SQL.

>>> Conclusion / Contributions

Main contributions:

μ-Algebra	A new algebra with new efficient rewriting rules for fixpoints	BDA'17, BDA'18
muSPARQL	A prototype evaluator based on streams	
CardEst	A new cardinality estimation technique	
SPARQLGX	A distributed SPARQL query evaluator	ISWC'16

Collaborations:

* Leaves enumeration technique. ICALP'17
* Distributed SPARQL query evaluators benchmarks. BDA'17
* ProvSQL: provenance for SQL. VLDB'18

Short-term perspectives

* Complete the distributed implementation of μ-algebra
* Use μ-algebra for SPARQL with ontology-based data access

Long-term perspectives

* Improve cardinality estimation scheme
* Port the optimization of the μ-algebra to SQL and Datalog

>>> Conclusion / Questions?

Questions?

Güneş Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee.
Diversified stress testing of rdf data management systems.

In *International Semantic Web Conference*, pages 197-212. Springer, 2014.

Edgar F Codd.
A relational model of data for large shared data banks.
Communications of the ACM, 13(6):377-387, 1970.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin.
LUBM: A benchmark for OWL knowledge base systems.
Web Semantics: Science, Services and Agents on the World Wide Web, 3(2):158-182, October 2005.

Steve Harris, Andy Seaborne, and Eric Prud'hommeaux.
SPARQL 1.1 query language.
W3C recommendation, 21(10), 2013.

Eric Prud'Hommeaux, Andy Seaborne, et al.
SPARQL query language for RDF.
W3C recommendation, 15, 2008.
www.w3.org/TR/rdf-sparql-query/.

David Wood Richard Cyganiak and Lanthaler Markus.
RDF 1.1 concepts and abstract syntax, February 2014.

Nikolay Yakovets, Parke Godfrey, and Jarek Gryz.
Towards Query Optimization for SPARQL Property Paths.
arXiv:1504.08262 [cs], April 2015.
arXiv: 1504.08262.