Evaluating Property Path Queries on online Knowledge Graphs

Julien AIMONIER-DAVAT, Hala-Skaf Molli, Pascal Molli

julien.aimonier-davat@ls2n.fr

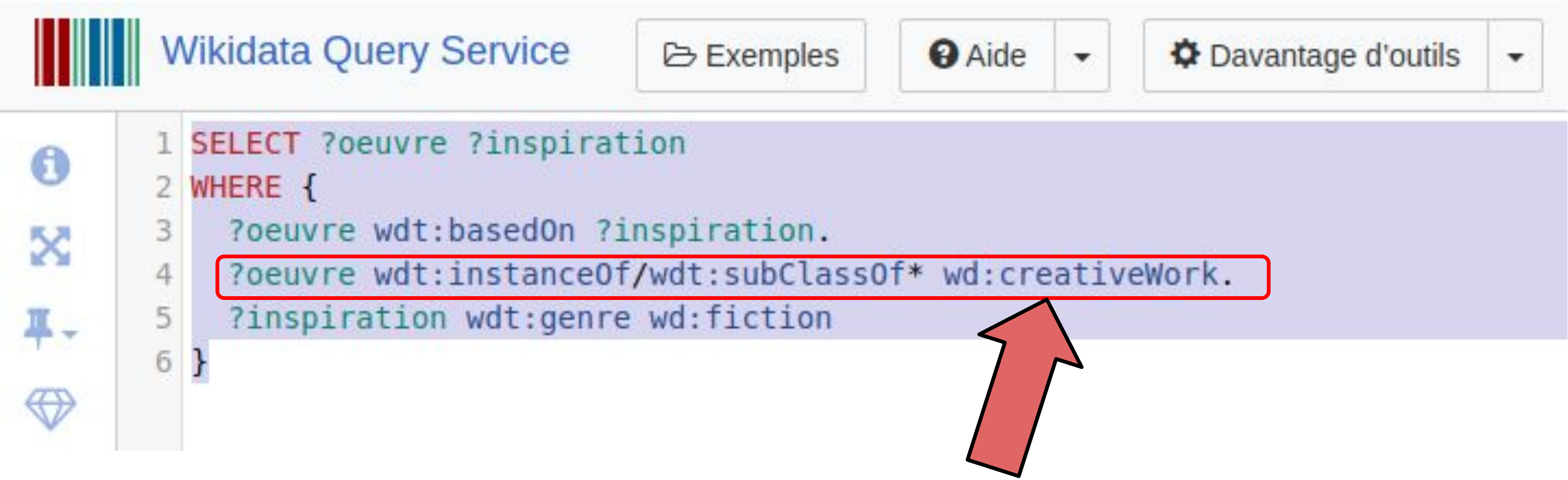
QuWeDa 2020 @ISWC 2020

UNIVERSITÉ DE NANTES

LS2N

I want to retrieve for each creative work the list of fiction works that inspired it !

Creative works and the list of fiction works that inspired it

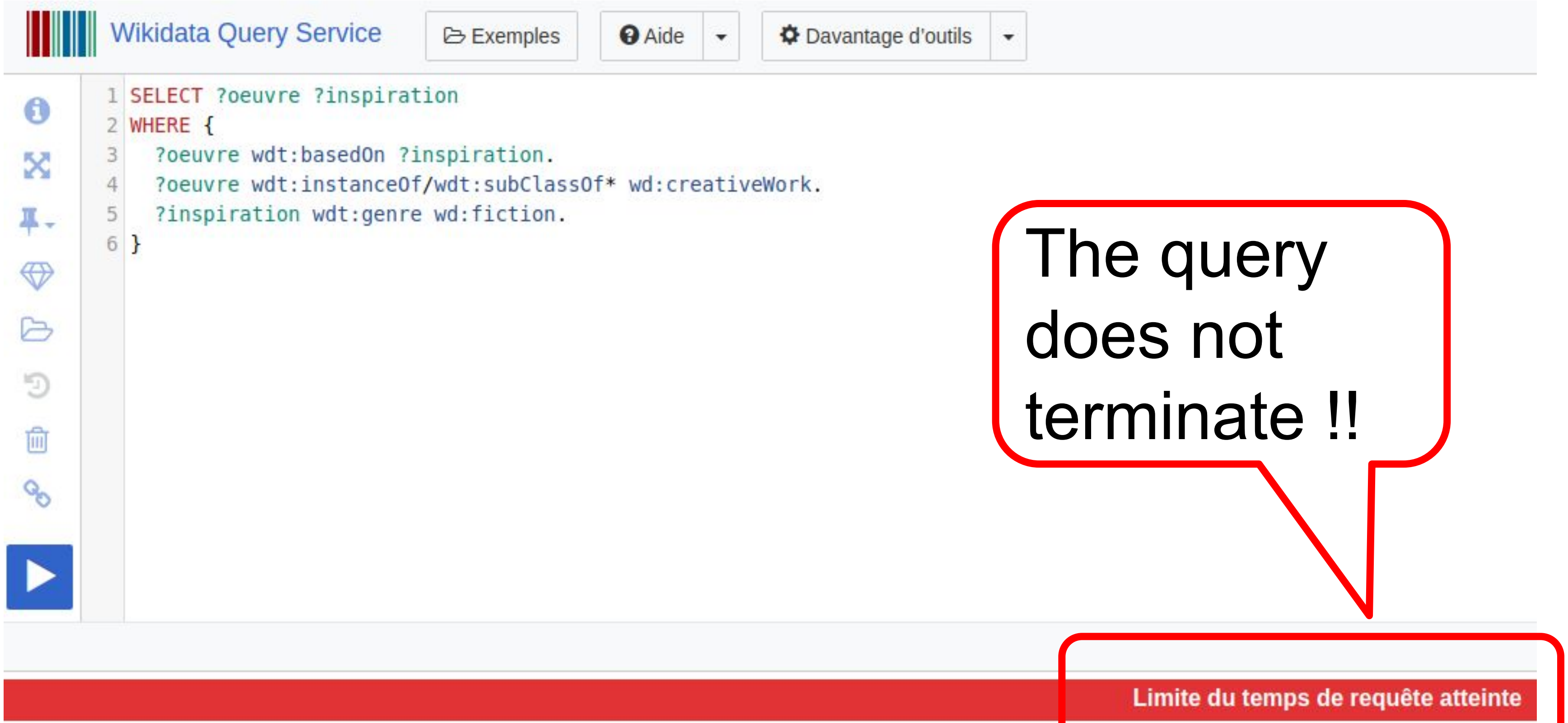


```
SELECT ?oeuvre ?inspiration
WHERE {
  ?oeuvre wdt:basedOn ?inspiration.
  ?oeuvre wdt:instanceOf/wdt:subClassOf* wd:creativeWork.
  ?inspiration wdt:genre wd:fiction
}
```

Property Path

Match path of arbitrary length !

On Wikidata: No Results

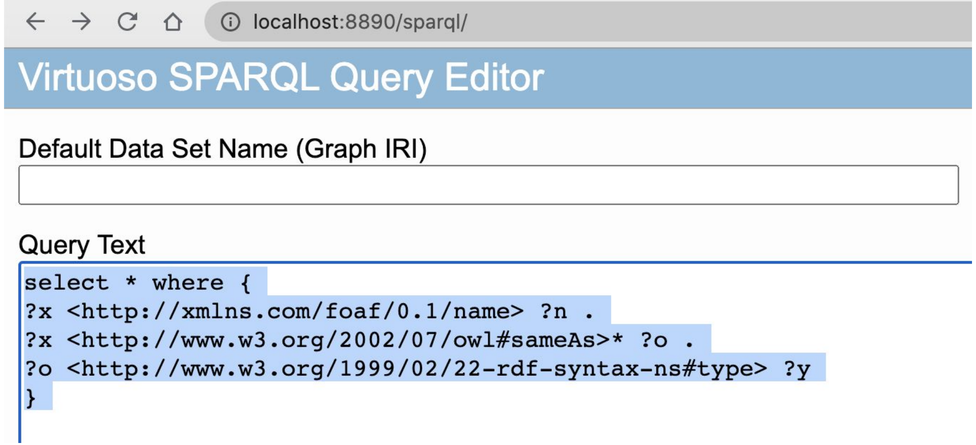


```
SELECT ?oeuvre ?inspiration
WHERE {
  ?oeuvre wdt:basedOn ?inspiration.
  ?oeuvre wdt:instanceOf/wdt:subClassOf* wd:creativeWork.
  ?inspiration wdt:genre wd:fiction.
}
```

Limite du temps de requête atteinte

The query does not terminate !!

On DBpedia: Resources

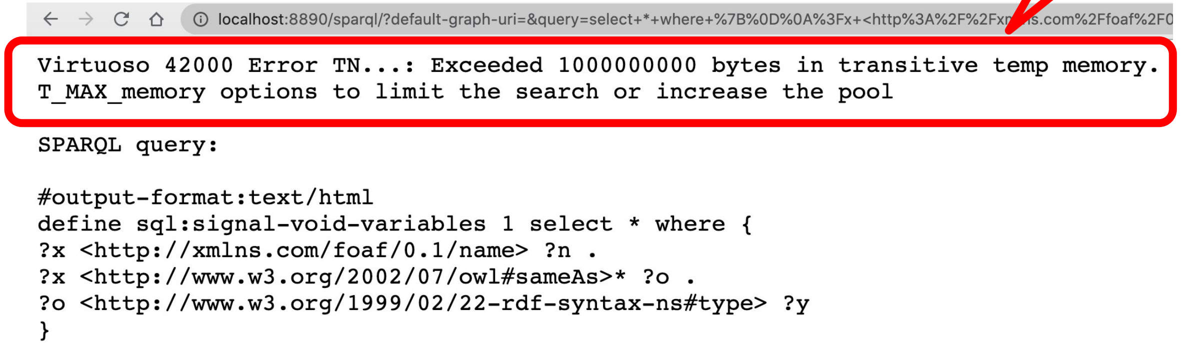


Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)

Query Text

```
select * where {
?x <http://xmlns.com/foaf/0.1/name> ?n .
?x <http://www.w3.org/2002/07/owl#sameAs>* ?o .
?o <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?y
}
```


localhost:8890/sparql/?default-graph-uri=&query=select+*+where+%7B%0D%0A%3Fx+<http%3A%2F%2Fxmlns.com%2Ffoaf%2F0

```
Virtuoso 42000 Error TN...: Exceeded 1000000000 bytes in transitive temp memory.
T_MAX_memory options to limit the search or increase the pool

SPARQL query:

#output-format:text/html
define sql:signal-void-variables 1 select * where {
?x <http://xmlns.com/foaf/0.1/name> ?n .
?x <http://www.w3.org/2002/07/owl#sameAs>* ?o .
?o <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?y
}
```

The query does not terminate !!

Property path queries are widely used

- Property path queries are about **34%** of Wikidata's queries[1]
- **62%** of them use **transitive** closure expressions
  - ?x (b|c)* ?y

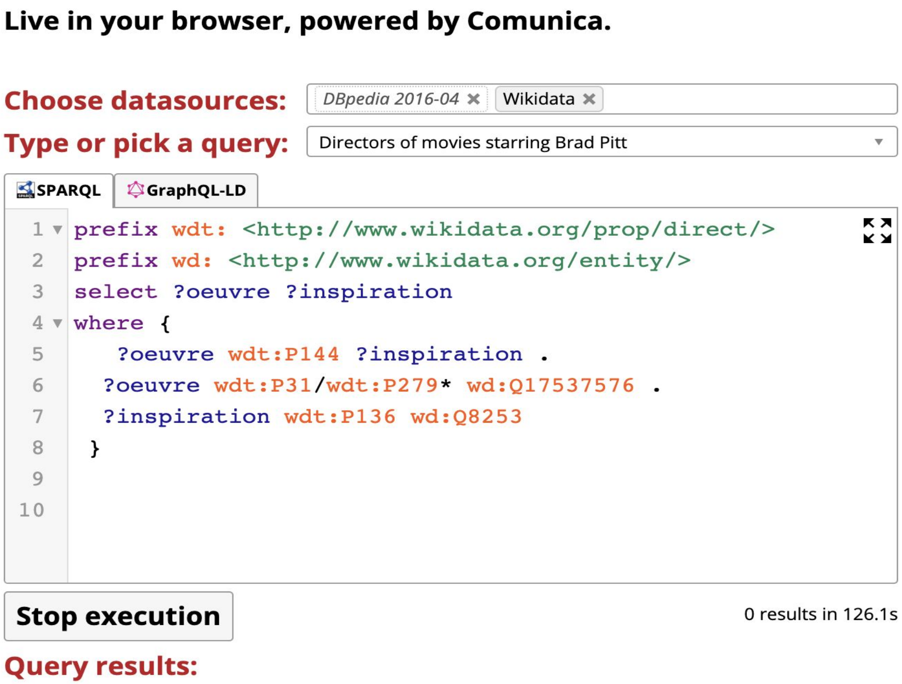
[1] Angela Bonifati, Wim Martens, and Thomas Timm. “Navigating the Maze of Wikidata Query Logs” 2019. The World Wide Web Conference.

How to execute queries with property paths online and get complete results ?

State of art

Evaluation of property path queries with TPF and Comunica

- TPF only supports **paginated triple pattern** queries
  - But any **query terminates**
- Comunica[2] decomposed property paths into triple pattern queries
  - We get complete results
- Poor performance



Live in your browser, powered by Comunica.

Choose datasources: *DBpedia 2016-04* ✖ Wikidata ✖

Type or pick a query: Directors of movies starring Brad Pitt

```
prefix wdt: <http://www.wikidata.org/prop/direct/>
prefix wd: <http://www.wikidata.org/entity/>
select ?oeuvre ?inspiration
where {
    ?oeuvre wdt:P144 ?inspiration .
   ?oeuvre wdt:P31/wdt:P279* wd:Q17537576 .
   ?inspiration wdt:P136 wd:Q8253
 }
```

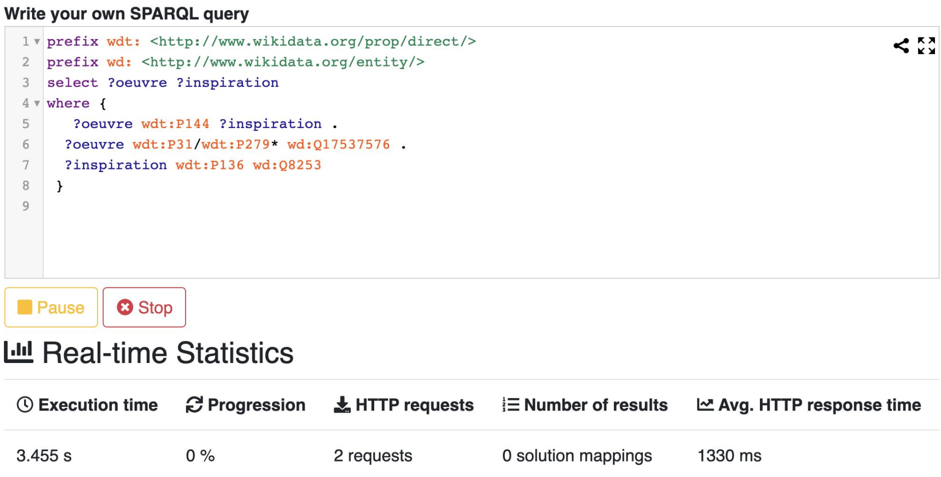
0 results in 126.1s

Query results:

[2] Ruben Taelman, Joachim Van Herwegen, Miel Vander Sande, and Ruben Verborgh. "Comunica: a Modular SPARQL Query Engine for the Web" 2018. *17th International Semantic Web Conference*.

Evaluation of property path queries with SaGe and SaGe-Jena

- SaGe server[3] can process Basic Graph Patterns, Unions, Filters, etc.
  - But not property paths
- SaGe-Jena reuses Jena property paths implementation
- Same problems as Comunica
  - Poor performance



Write your own SPARQL query

```
prefix wdt: <http://www.wikidata.org/prop/direct/>
prefix wd: <http://www.wikidata.org/entity/>
select ?oeuvre ?inspiration
where {
   ?oeuvre wdt:P144 ?inspiration .
  ?oeuvre wdt:P31/wdt:P279* wd:Q17537576 .
  ?inspiration wdt:P136 wd:Q8253
 }
```

Real-time Statistics

Execution time	Progression	HTTP requests	Number of results	Avg. HTTP response time
3.455 s	0 %	2 requests	0 solution mappings	1330 ms

[3] Thomas Minier, Hala Skaf-Molli and Pascal Molli. "SaGe: Web Preemption for Public SPARQL Query services" 2019. The World Wide Web Conference (WWW'19).

Evaluation of property path queries on smart clients

Without transitivity support on the server, smart clients decompose property paths into a set of triple pattern queries:

- **High data transfer**
  - All intermediate results are transferred from the server to the client
  - Worst case O(|V|+|E|)
- **High number of HTTP calls:**
  - Pay network latency at calls
  - Worst cast O(|V|)

Property Paths Queries

SPARQL Endpoints

- Fast when under the quota
- But, no guarantee of termination

TPF, Web preemption

- Terminates...
- But, prohibitive data transfer, slow

How to compute efficiently SPARQL queries with property path online and get complete results ?

Objectives

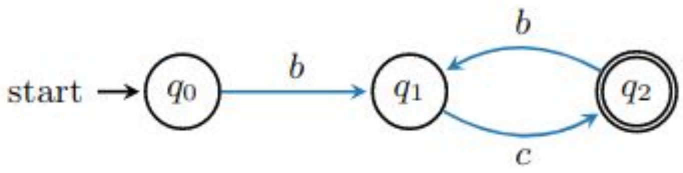
Compute property paths **without transitivity** support on **the server** while minimizing data transfer and HTTP calls.

Automaton Compression Approach

Key idea : Automaton compression

Automaton-based approach

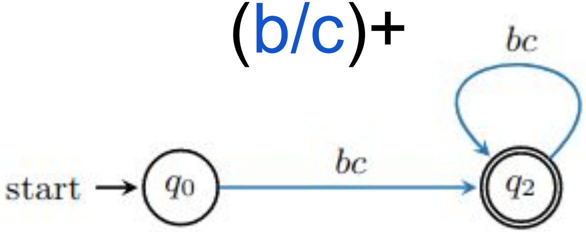
(b/c)+



- 1 transition □ 1 **triple pattern query**
- **Decomposition into a set of triple pattern queries**
- Many client-side joins, many HTTP calls, many intermediate results

Automaton Compression

(b/c)+



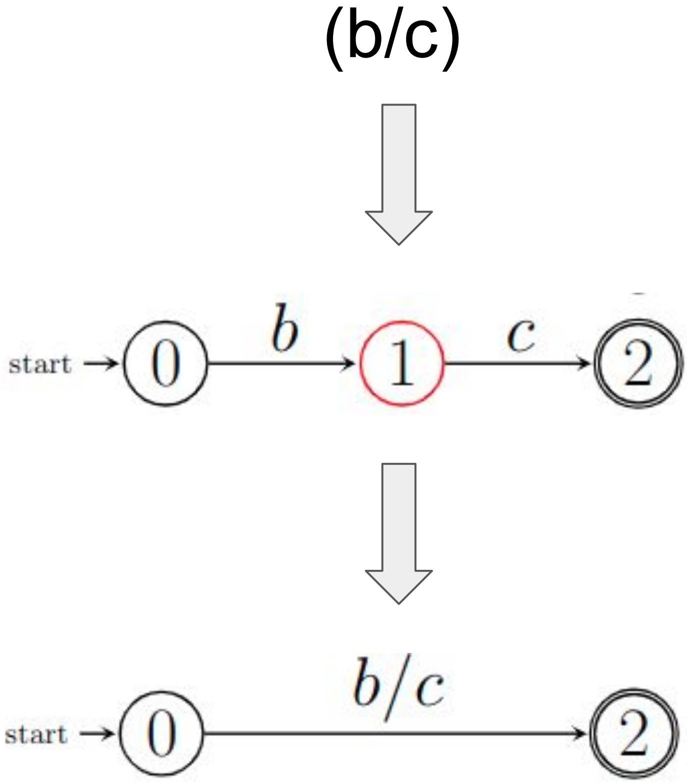
- 1 transition □ 1 **BGP query**
- **Compute as many joins as possible on the server**
- Less client-side joins, less calls, less intermediate results

Scientific Problem

Given a SPARQL query with a property paths Q and its corresponding **mono-predicate** automaton A, transform A into a **multi-predicate** automaton A’ such that L(A) = L(A’)

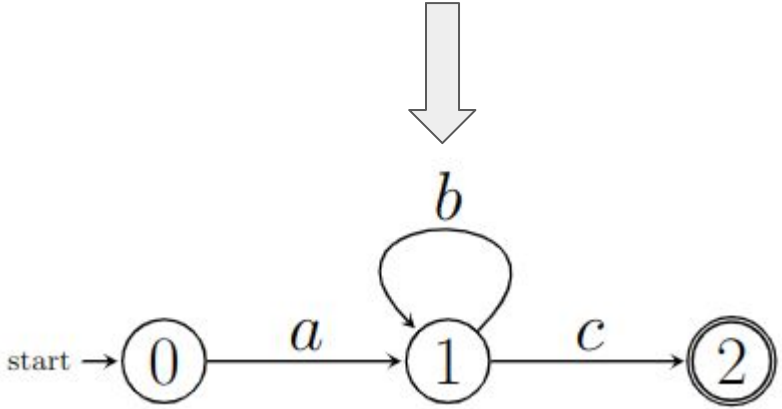
Compression with the join operator

- An intermediate state is:
  - Not the initial state
  - **A non-final state**
  - **A state without backward transitions**
- A backward transition:
  - incoming state <= outgoing state



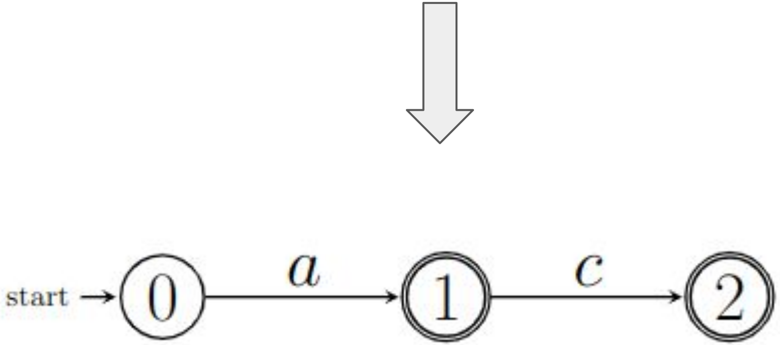
Compression with the join operator

(a/b*/c)



- State 1 has a **backward transition**
- State 1 cannot be removed without transitivity support on the server

(a?/c)

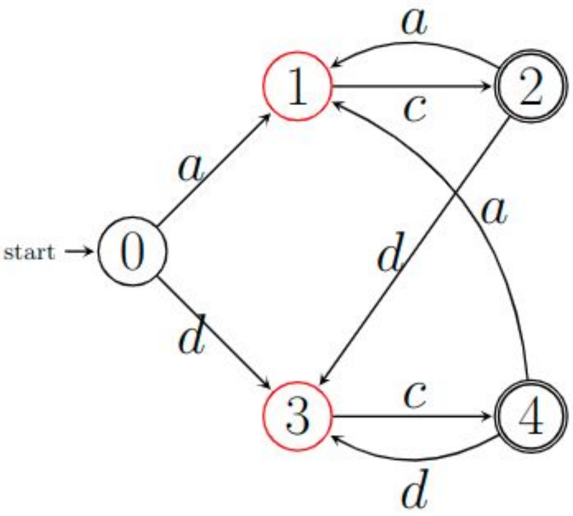


- State 1 is a **final state**
- State 1 cannot be removed

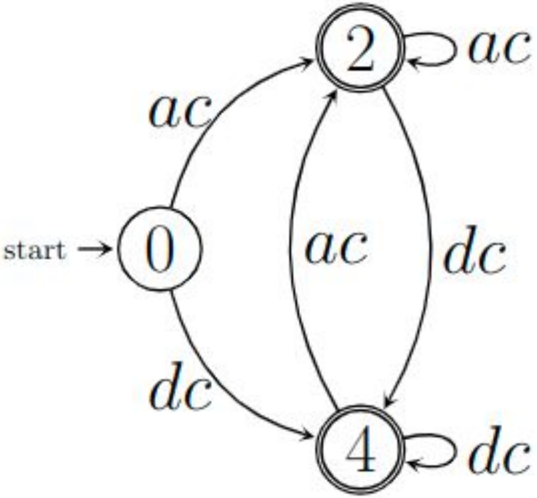
From mono-predicate to multi-predicate

- Replace each sequence of transitions by a single transition when:
  - Extremities are non-intermediate states
  - The other states are intermediate states
- Shortest paths between non-intermediate states

Example with the Property Path ((a/c) | (d/c))+



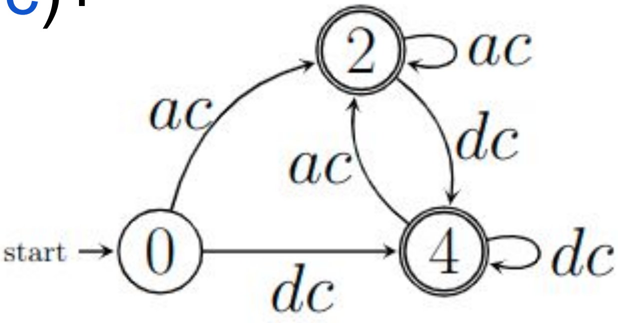
Shortest paths



Key idea : Automaton compression

Compression without UNION

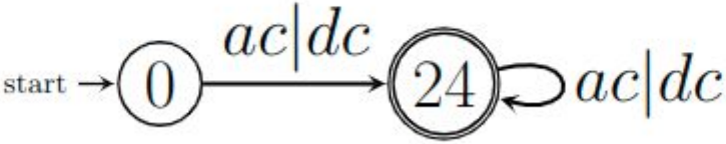
(ac|dc)+



- 1 transition □ 1 **BGP query**
- **2 calls** at each iteration
- More calls, poorer execution time (network latency cost)

Compression with UNION

(ac|dc)+



- 1 call □ 1 **union of BGPs query**
- **1 call** at each iteration
- Less calls, better execution time
- **+50% of human queries on Wikidata**

Experimental study

Experimental study

- Does **automaton compression** outperform **Comunica** and **SaGe-Jena** ?

- Does using a **multi-predicate** automaton improve property path queries evaluation compared to a **mono-predicate** automaton ?

Datasets and Queries

- **BeSEPPI**'s benchmark[4] updated with the clique test introduced in[5]
  - 242 queries (73 ASK and 169 SELECT)
  - Dataset with 59 triples, stored using HDT
  - Used to test the conformance of all approaches with the W3C's semantics

[4] A. Skubella, D. Janke and S. Staab. "BeSEPPI: Semantic-Based Benchmarking of Property Path Implementations" 2019. The European Semantic Web Conference.
[5] Arenas and al. "Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard" 2012. Proceedings of the 21st international conference on World Wide Web.

Datasets and Queries

- **GMark framework**[6]
  - Generated graph
    - Same settings as “Shop” use-case
    - 10^6 triples, stored using HDT
  - Generate 30 BGP queries
    - 10 without transitive closures
    - 20 with transitive closures
  - Used to test the **automaton compression** on queries with complex property path expressions

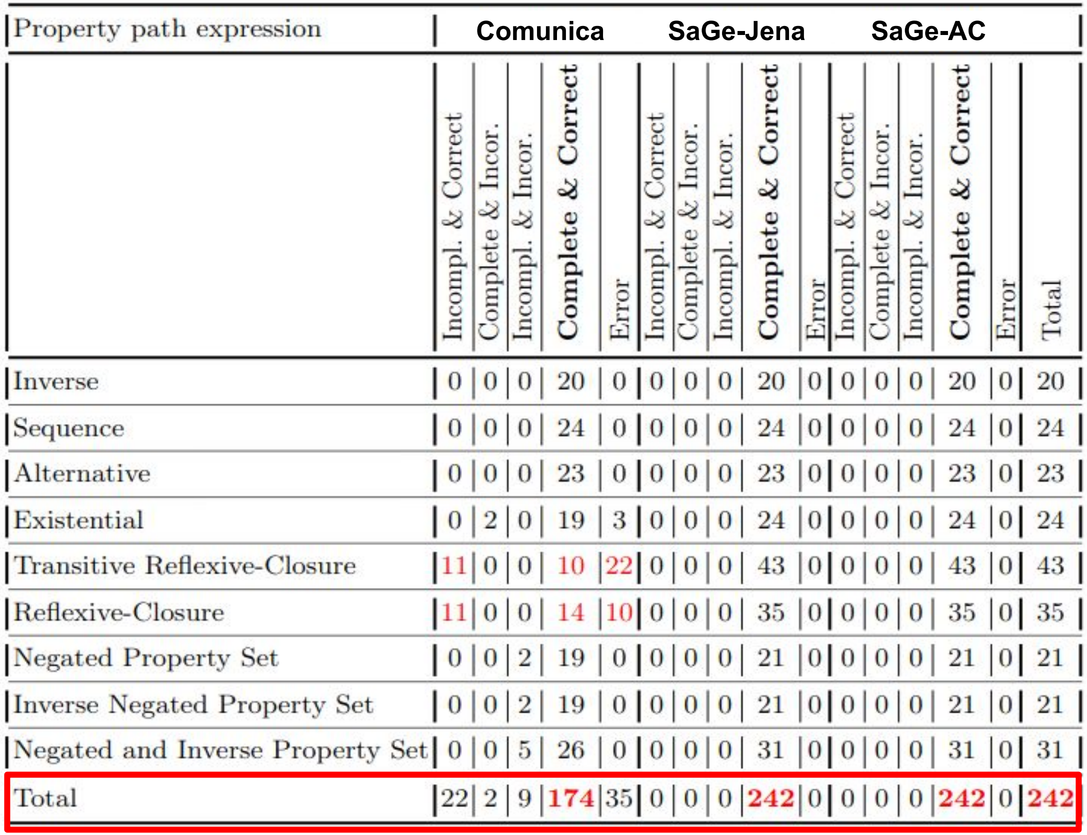
[6] Bagan and al. “gMark: Schema-driven generation of graphs and queries” 2016. IEEE Transactions on Knowledge and Data Engineering.

Compared approaches

- **Baseline: Comunica TPF smart client**
  - TPF server, page-size 2000 mappings
- **Baseline: SaGe-Jena (standard automaton)**
  - SaGe client based on Apache Jena
  - Quantum of 75ms
  - Page-size of 2000 mappings
- **Automaton without compression: SaGe-A**
  - Extension of the SaGe client framework
- **Automaton Compression: SaGe-AC**
  - Extension of the SaGe client framework

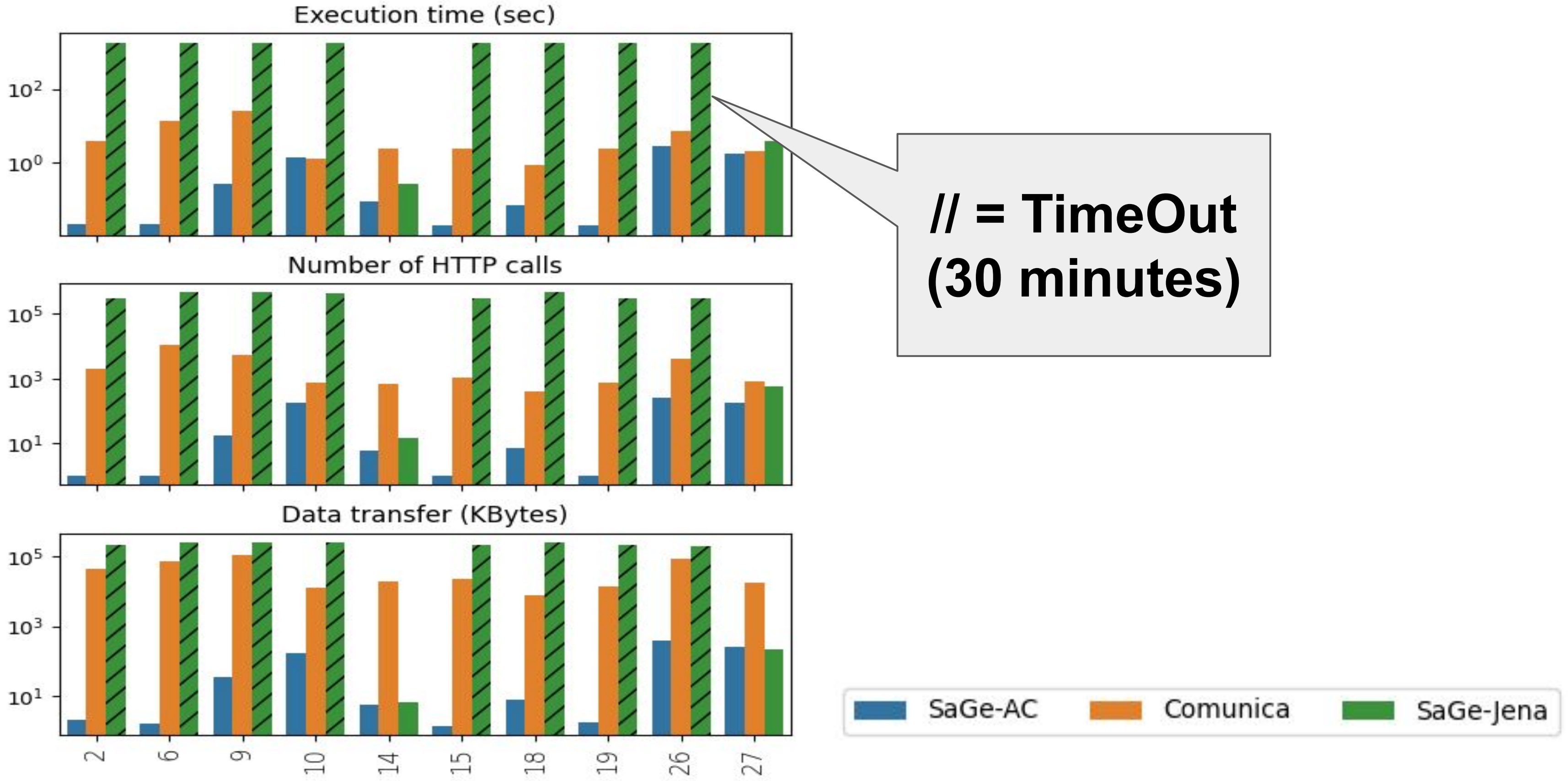
Conformance to property path semantics

- Automaton Compression and SaGe-Jena follow the semantics
- Comunica cannot evaluate some transitive closure expressions...



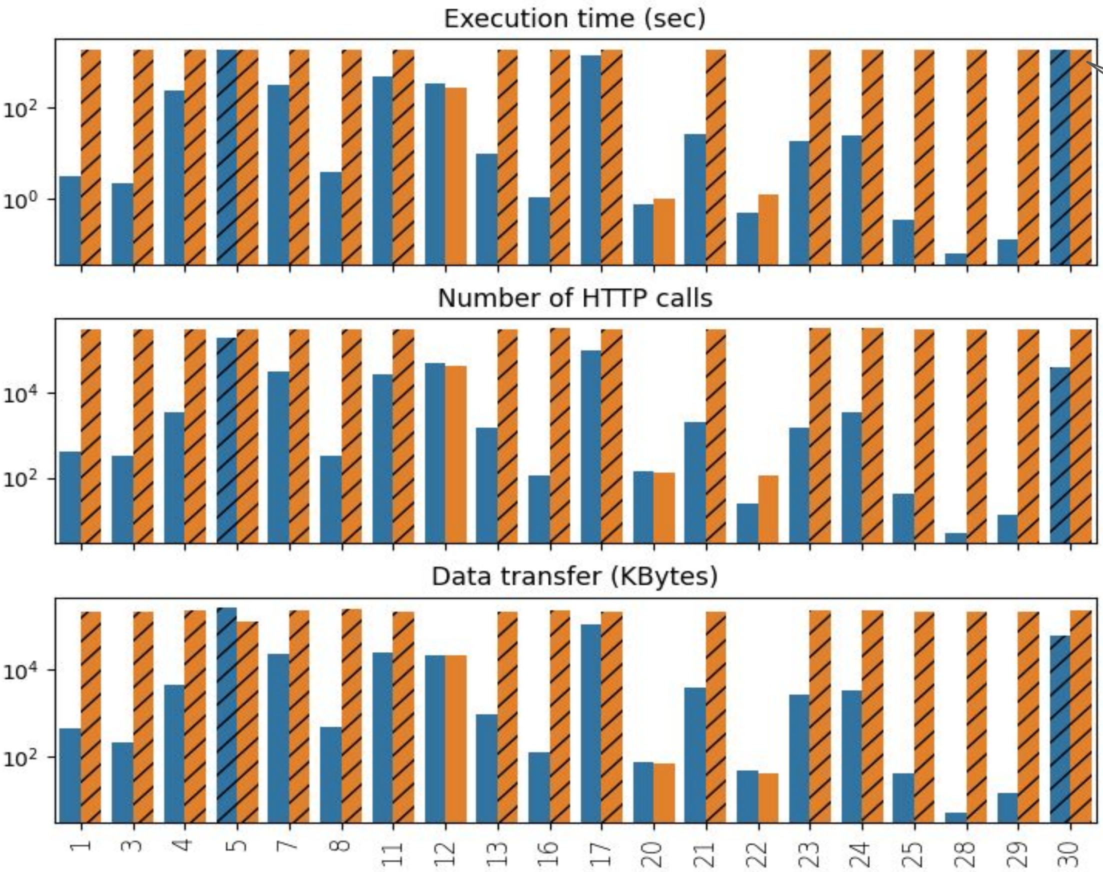
Property path expression	Comunica					SaGe-Jena					SaGe-AC					
	Incompl. & Correct	Complete & Incor.	Incompl. & Incor.	**Complete & Correct**	Error	Incompl. & Correct	Complete & Incor.	Incompl. & Incor.	**Complete & Correct**	Error	Incompl. & Correct	Complete & Incor.	Incompl. & Incor.	**Complete & Correct**	Error	Total
Inverse	0	0	0	20	0	0	0	0	20	0	0	0	0	20	0	20
Sequence	0	0	0	24	0	0	0	0	24	0	0	0	0	24	0	24
Alternative	0	0	0	23	0	0	0	0	23	0	0	0	0	23	0	23
Existential	0	2	0	19	3	0	0	0	24	0	0	0	0	24	0	24
Transitive Reflexive-Closure	11	0	0	10	22	0	0	0	43	0	0	0	0	43	0	43
Reflexive-Closure	11	0	0	14	10	0	0	0	35	0	0	0	0	35	0	35
Negated Property Set	0	0	2	19	0	0	0	0	21	0	0	0	0	21	0	21
Inverse Negated Property Set	0	0	2	19	0	0	0	0	21	0	0	0	0	21	0	21
Negated and Inverse Property Set	0	0	5	26	0	0	0	0	31	0	0	0	0	31	0	31
Total	22	2	9	**174**	35	0	0	0	**242**	0	0	0	0	**242**	0	**242**

Execution time, number of HTTP calls and data transfer for property path queries without transitive closures



// = TimeOut (30 minutes)

Execution time, number of HTTP calls and data transfer for property path queries with transitive closure

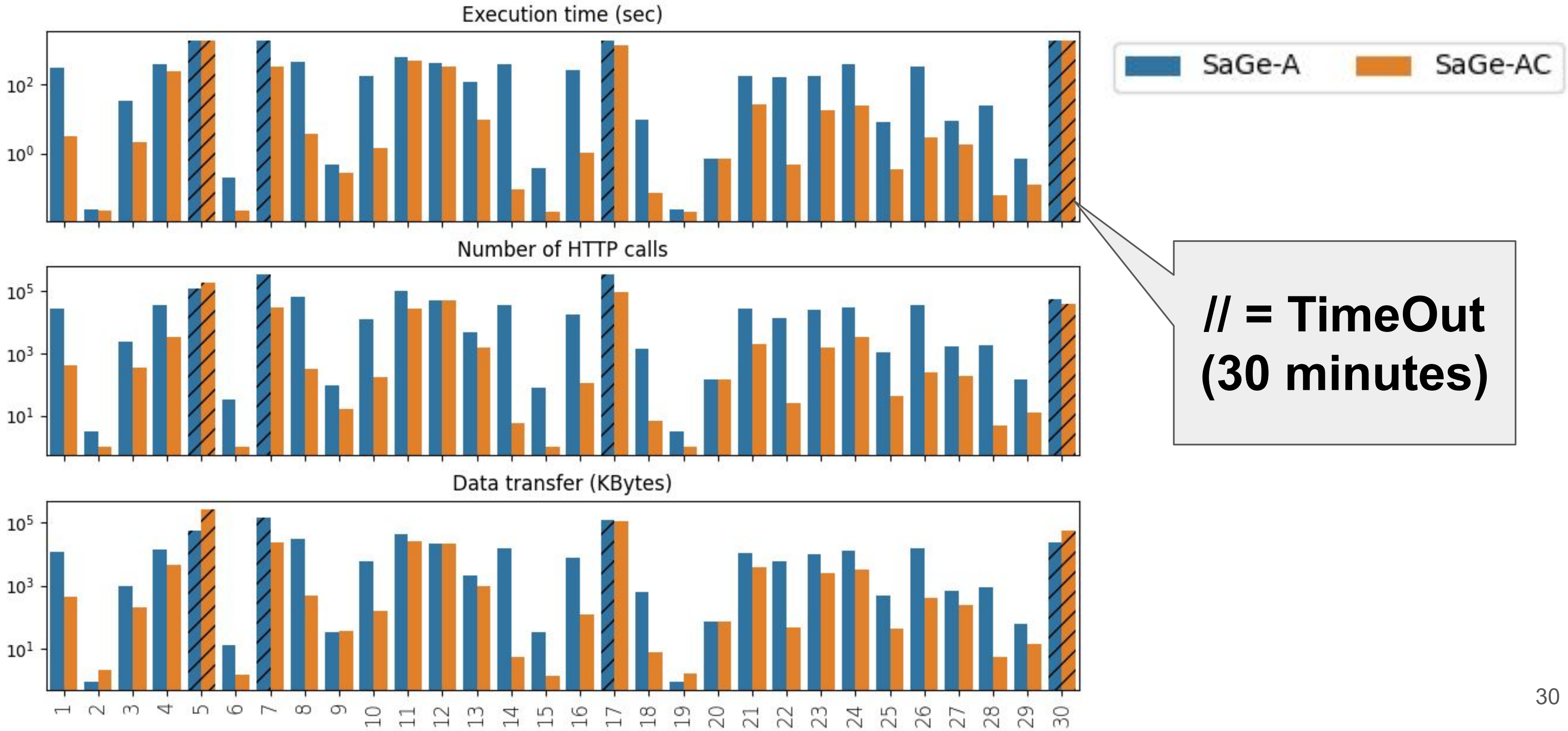


// = TimeOut (30 minutes)

Comunica does not handle these queries

🟦 SaGe-AC 🟧 SaGe-Jena

Execution time, number of HTTP calls and data transfer for property path queries for SaGe-A and SaGe-AC



// = TimeOut (30 minutes)

Conclusions

- We proposed a new algorithm to build an automaton that takes advantage of the server-side operators (joins, unions, filters) to compute property paths
- Our proposal outperforms existing client-side solutions

Perspectives

- Proving the optimality of the compression
- Looking for a preemptive decomposition of property path queries
  - May be an operator to compute “partial” property path on the server.

Thank you