TOWARDS A SYSTEMATIC BENCHMARKING OF ONTOLOGY-BASED QUERY REWRITING SYSTEMS

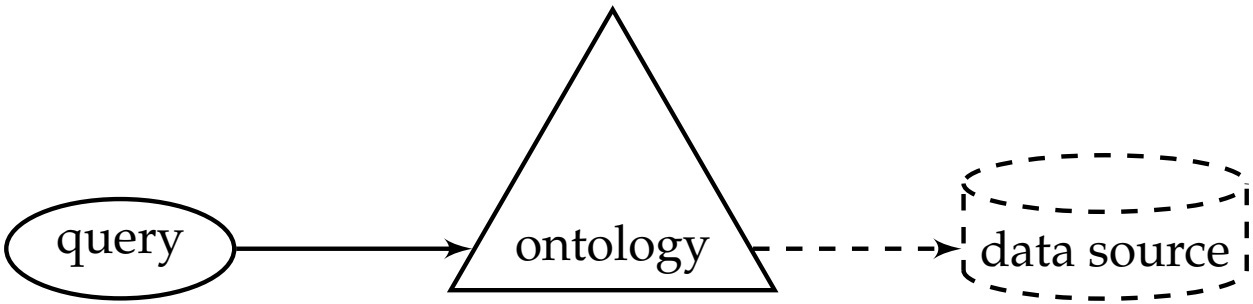
José Mora and Óscar Corcho

{jmora, ocorcho}@fi.upm.es

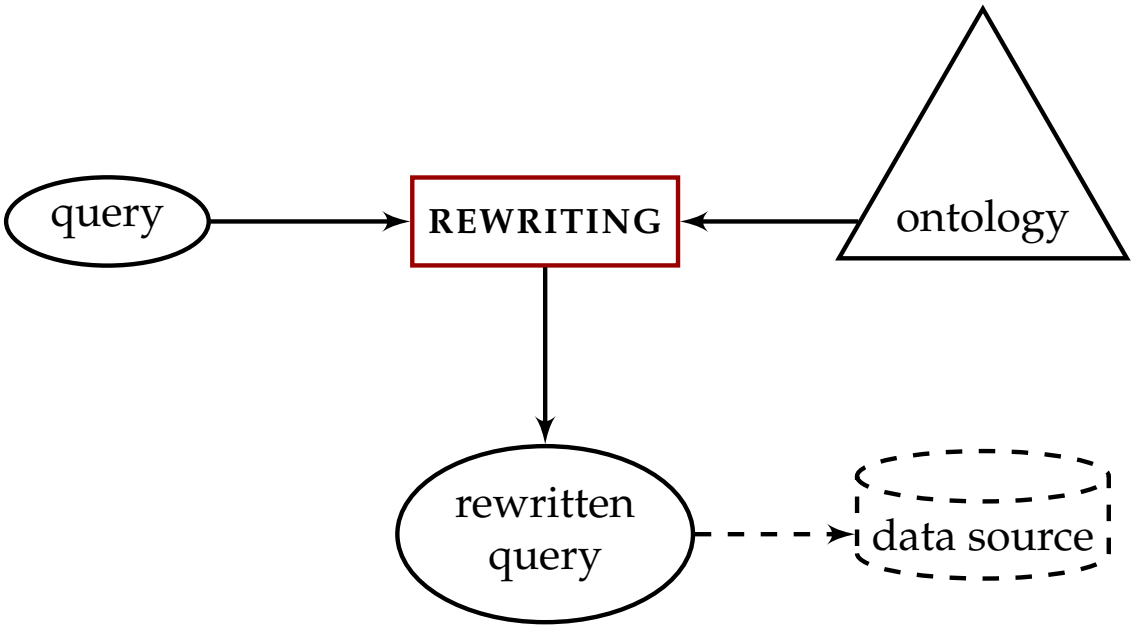
Boadilla - October 10, 2013

Index

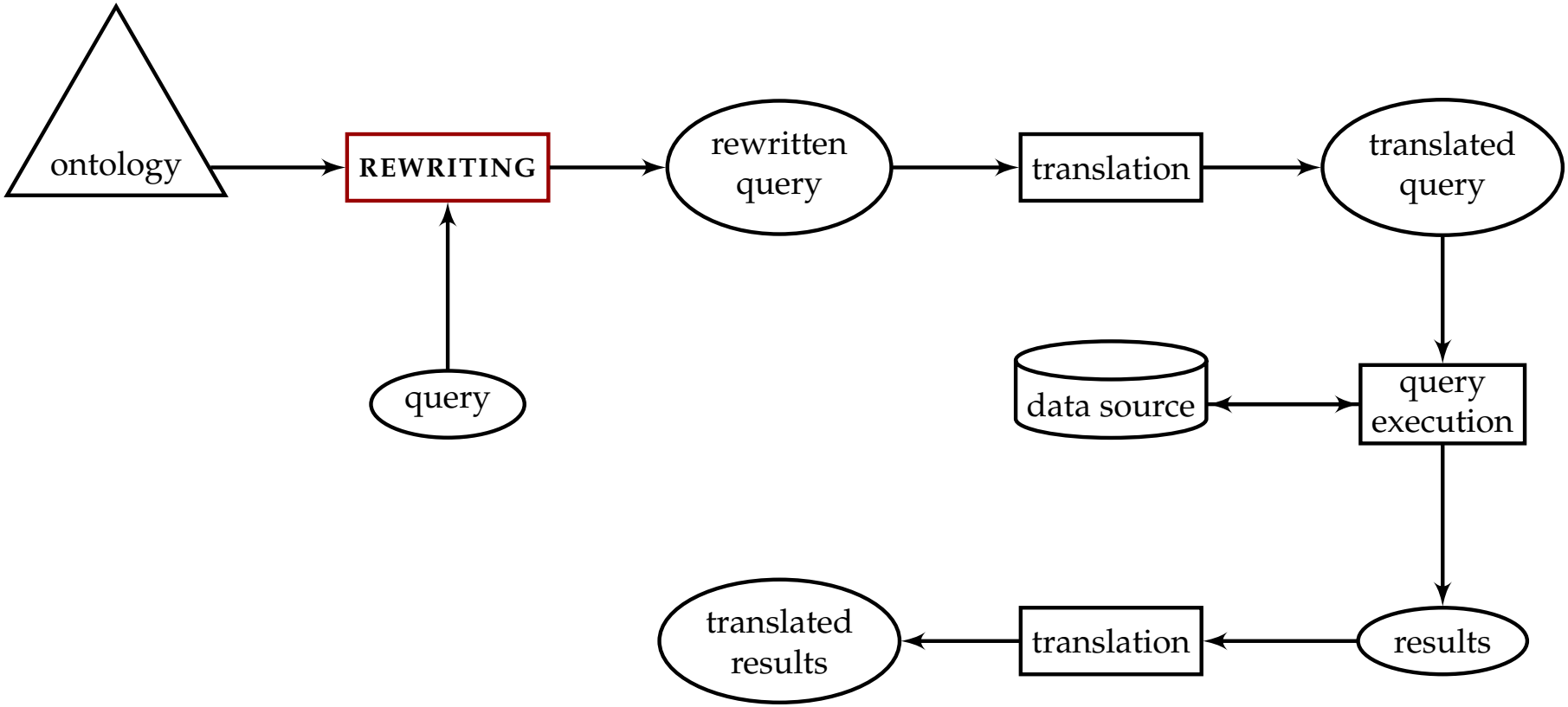
IDEA



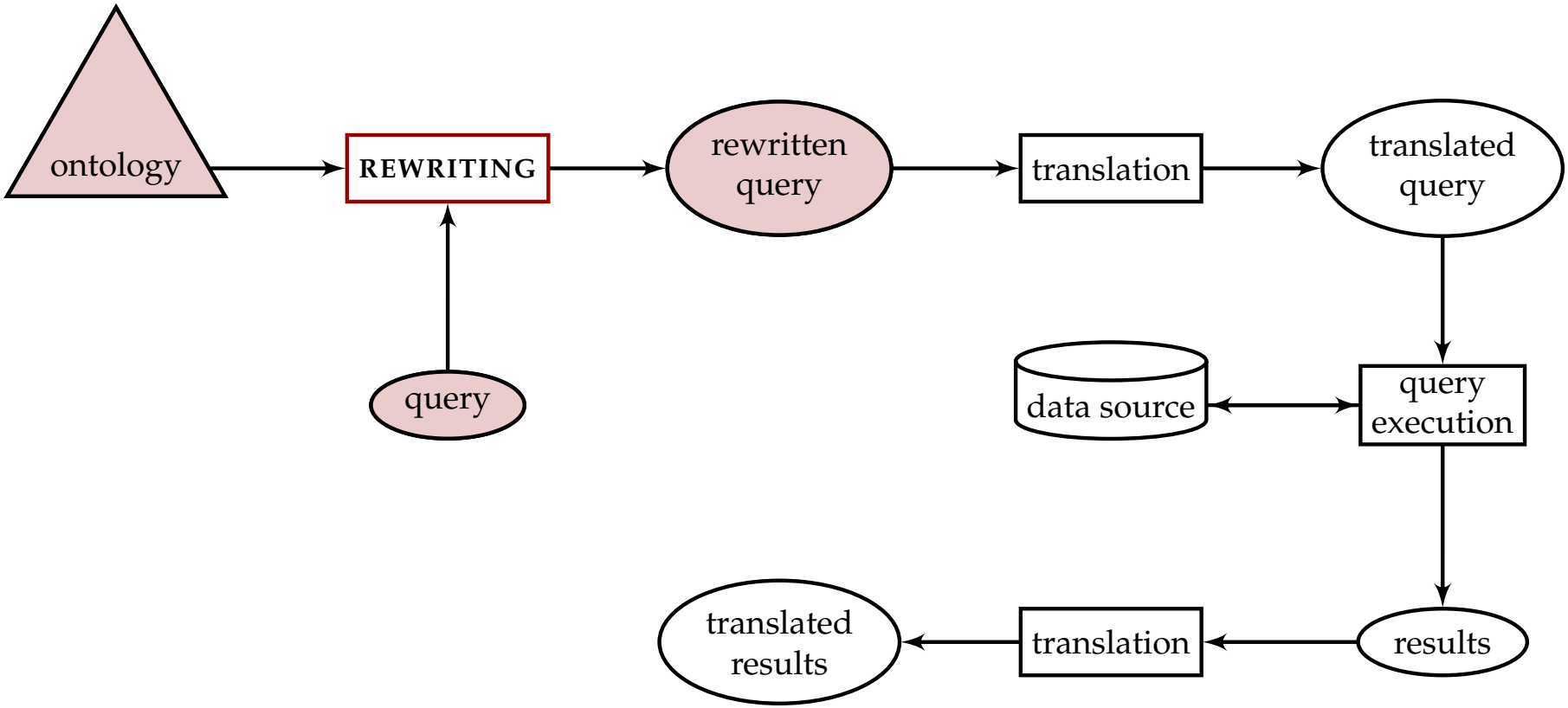
Approach



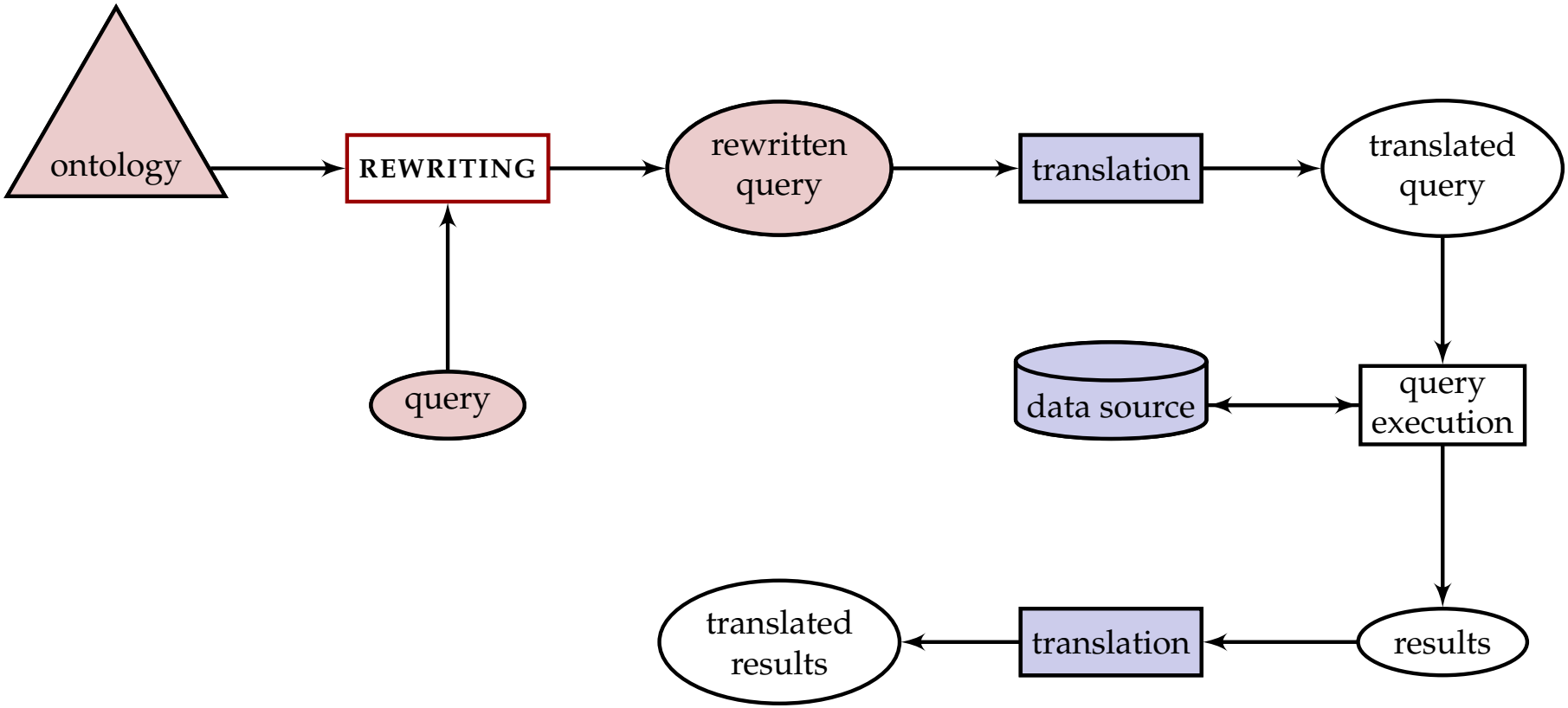
CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
data source
query execution
translated results
translation
results

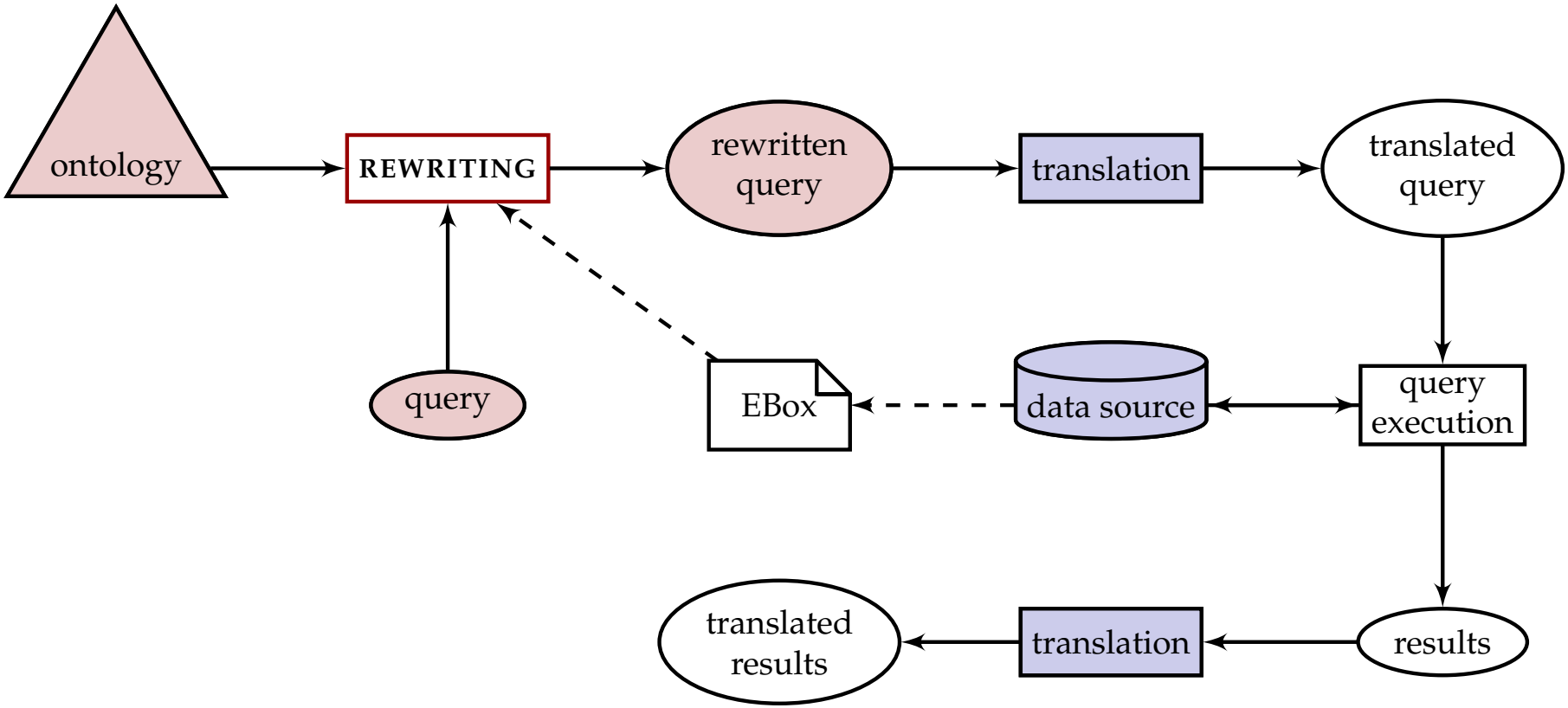
CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
data source
query execution
translated results
translation
results

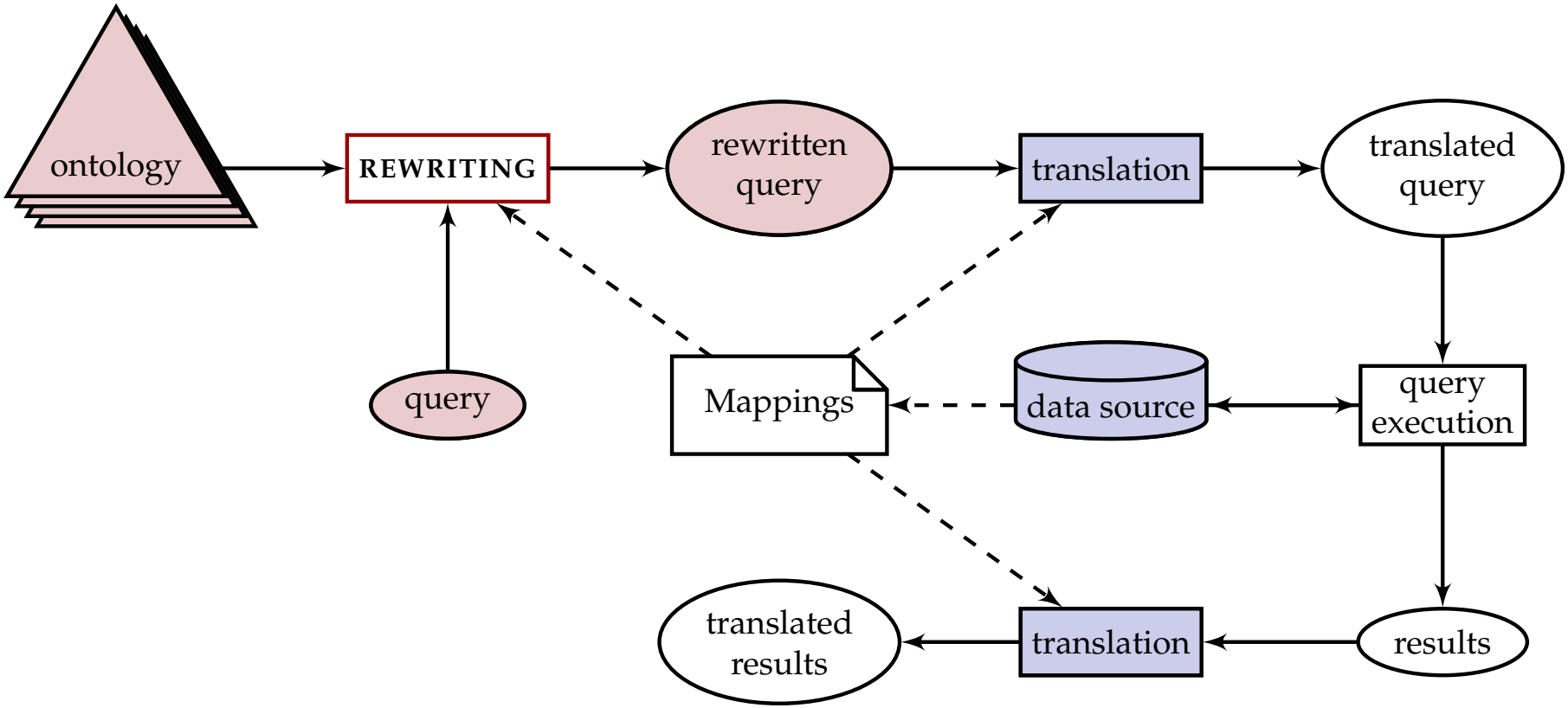
CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
data source
query execution
translated results
translation
results

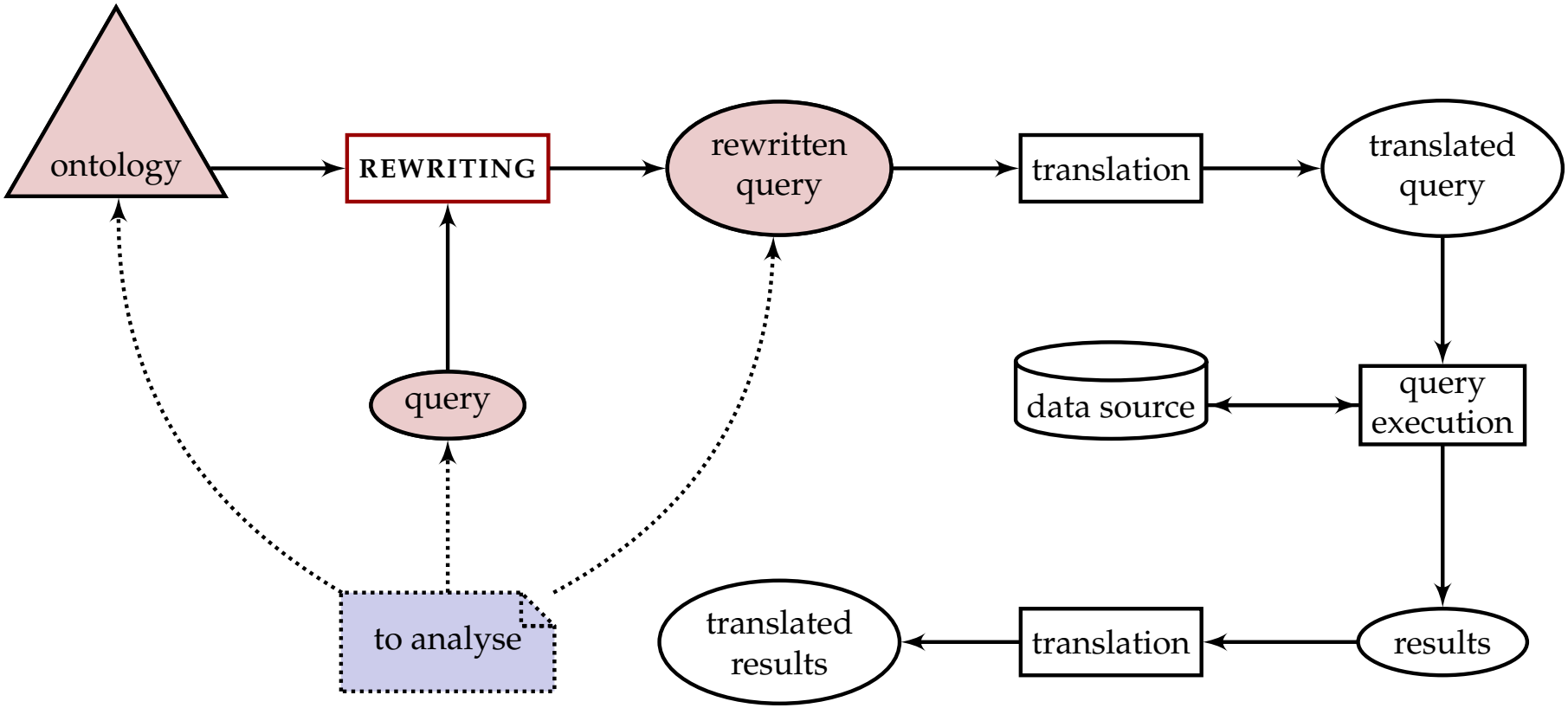
CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
EBox
data source
query execution
translated results
translation
results

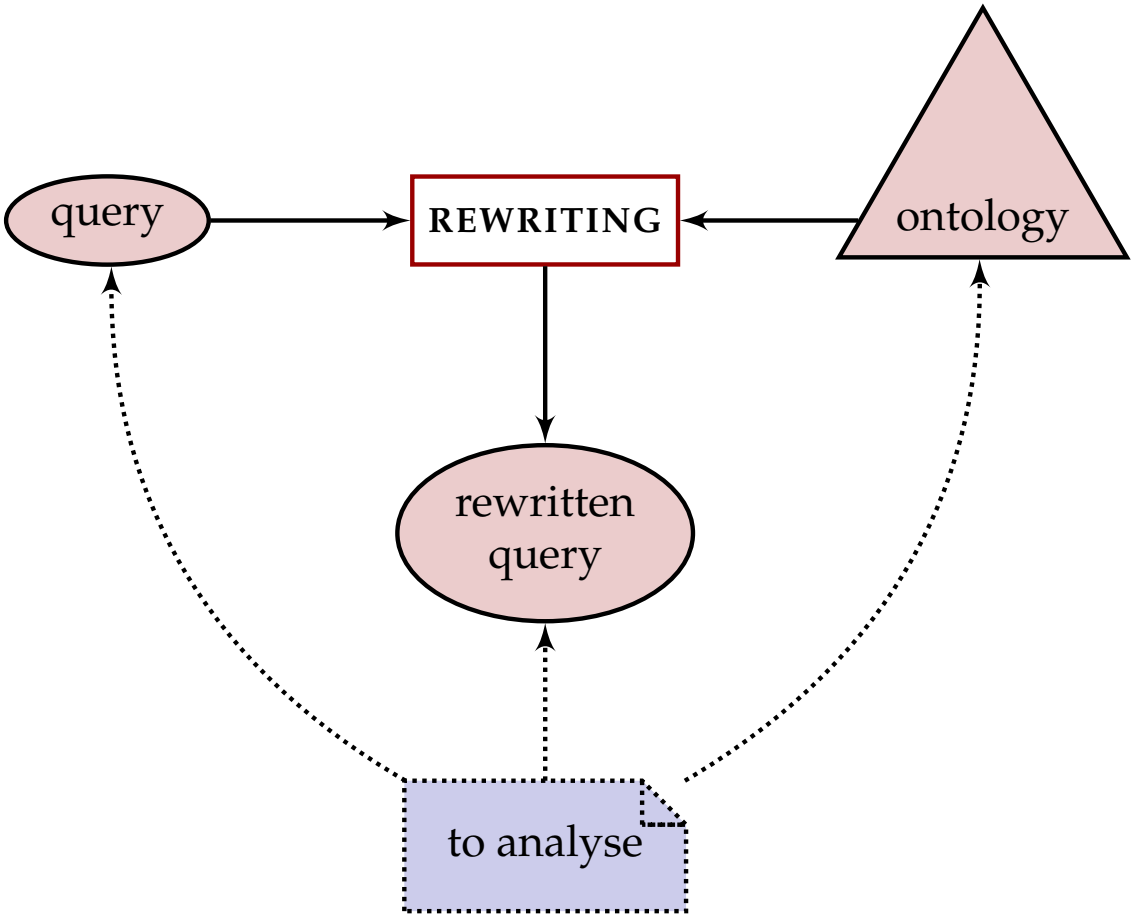
CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
Mappings
data source
query execution
translated results
translation
results

CONTEXT


ontology
REWRITING
rewritten query
translation
translated query
query
data source
query execution
to analyse
translated results
translation
results

FOCUSING



Logics

axiom[1] \ logic	DL-Lite$_{core}$	DL-Lite$_F$	DL-Lite$_R$	Rapid's	$\mathcal{ELHIO}^{\neg}$	Datalog$^{\pm 2}$	Horn-$\mathcal{SHIQ}$
$B_1 \sqsubseteq B_2, B_1 \sqsubseteq \neg B_2$[3]	✓	✓	✓	✓	✓	✓	✓
$\geq 2R_1 \sqsubseteq \bot$	✗	✓	✗	✗	✗	✗	✓
$R_1 \sqsubseteq R_2, R_1 \sqsubseteq \neg R_2$	✗	✗	✓	✓	✓	✓	✓
$B_1 \sqsubseteq \exists R_1, \exists R_1 \sqsubseteq B_1$	✗	✗	✓	✓	✓	✓	✓
$B_1 \sqsubseteq \exists R_1.B_2$	✗	✗	✗	✓	✓	✓	✓
$\exists R_1.B_1 \sqsubseteq B_2$	✗	✗	✗	✗	✓	✗	✓
$B_1 \sqcap B_2 \sqsubseteq B_3$	✗	✗	✗	✗	✓	✗	✓
$\{a\} \sqsubseteq B, B \sqsubseteq \{a\}, B(a)$	✗	✗	✗	✗	✓	✗	✓
n-ary predicates	✗	✗	✗	✗	✗	✓	✗
$trans(R_1)$	✗	✗	✗	✗	✗	✗	✓
$B_1 \sqsubseteq \forall R_1.B_2, B_1 \sqsubseteq \leq 1R_1.B_2$	✗	✗	✗	✗	✗	✗	✓

[1] here B_i represents a basic concept and R_j a role that may be basic or inverted.
[2] as implemented in Nyaya
[3] note that some systems implementing negation assume a consistent ABox

SYSTEMS

System	Input	Output	Reference
Quonto	DL-Lite$_R$	UCQ	Calvanese et al. (2007)
REQUIEM	$\mathcal{ELHIO}^\neg$	Datalog or UCQ	Pérez-Urbina et al. (2009)
Presto	DL-Lite$_R$	Datalog	Rosati and Almatelli (2010)
Rapid	DL-Lite$_R$[4]	Datalog or UCQ	Chortaras et al. (2011)
Nyaya	*Datalog*$^\pm$	UCQ	Gottlob et al. (2011)
Venetis'	DL-Lite$_R$	UCQ	Venetis et al. (2011)
Prexto	DL-Lite$_R$ and EBox	Datalog or UCQ	Rosati (2012)
Clipper	Horn-$\mathcal{SHIQ}$	Datalog	Eiter et al. (2012)
kyrie	$\mathcal{ELHIO}^\neg$	Datalog or UCQ	Mora and Corcho (2013)

[4] Close to OWL2 QL, $B_1 \sqsubseteq \exists R.B_2$ axioms are supported

Previous efforts

- Imprialou et al. (2012) is the most notable previous effort
- Automatic and exhaustive coverage of the ontologies with queries
- Detected flaws in the implementations of most systems
- Still focused on qualitative results (soundness and completeness)

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

DIMENSIONS

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality™

Dimensions

- Expressiveness in tests: traditionally DL-Lite$_R$
  - Most expressive logic in the intersection of all systems
  - Systems that handle more expressive logics cannot show their full potential
- Expressiveness lost in translation to Datalog
  - How expressive are the ontologies for OBDA?
  - Consequences of not covered expressiveness on precision and completeness
- Output complexity: apples, oranges and pears
  - How to compare UCQs and Datalog programs?
  - Characteristics of the system that is going to execute them
- Input complexity: treat with care
  - What kind of queries can be processed?
  - What kind of queries do we want to process?
- Additional inputs: to each one its own
  - EBox, cache of previous queries, etc.
  - Comparison among systems and comparison with Reality^{TM}

ASSETS

- Several assets used for evaluation[5]
- Usual ontologies (A, AX, P1, P5, P5X, S, U, UX, V)
- Not so usual ontologies (core, galen-lite)
- New ontologies (AXE, AXEb, P5XE, UXE)
- Usually with 5 queries, but up to 9 in some cases

[5] available here: `http://j.mp/benchmarkqueryrewriting`

Assets

- Several assets used for evaluation[5]
- Usual ontologies (A, AX, P1, P5, P5X, S, U, UX, V)
- Not so usual ontologies (core, galen-lite)
- New ontologies (AXE, AXEb, P5XE, UXE)
- Usually with 5 queries, but up to 9 in some cases

[5] available here: `http://j.mp/benchmarkqueryrewriting`

ASSETS

- Several assets used for evaluation[5]
- Usual ontologies (A, AX, P1, P5, P5X, S, U, UX, V)
- Not so usual ontologies (core, galen-lite)
- New ontologies (AXE, AXEb, P5XE, UXE)
- Usually with 5 queries, but up to 9 in some cases

[5] available here: `http://j.mp/benchmarkqueryrewriting`

ASSETS

- Several assets used for evaluation[5]
- Usual ontologies (A, AX, P1, P5, P5X, S, U, UX, V)
- Not so usual ontologies (core, galen-lite)
- New ontologies (AXE, AXEb, P5XE, UXE)
- Usually with 5 queries, but up to 9 in some cases

[5] available here: `http://j.mp/benchmarkqueryrewriting`

ASSETS

- Several assets used for evaluation[5]
- Usual ontologies (A, AX, P1, P5, P5X, S, U, UX, V)
- Not so usual ontologies (core, galen-lite)
- New ontologies (AXE, AXEb, P5XE, UXE)
- Usually with 5 queries, but up to 9 in some cases

[5] available here: `http://j.mp/benchmarkqueryrewriting`

RESULTS I

		REQUIEM(F)		Presto		Rapid		Clipper		kyrie	
$\mathcal{O}$	q	size	time	size	time	size	time	size	time	size	time
U	1	19	12	4	7	4	3	2	21	2	0
	2	47	16	2	9	2	9	49	19	47	3
	3	20	9	8	16	8	12	21	24	20	3
	4	64	15	3	12	3	3	63	18	64	3
	5	53	12	8	15	8	12	53	15	53	0
	6	20	12	19	6	21	15	16	18	16	9
	7	49	25	22	15	22	18	44	18	45	12
	8	10	9	13	6	13	9	10	20	10	3
	9	29	15	24	12	24	17	19	20	21	7
UX	1	22	6	7	10	7	3	5	27	5	6
	2	52	15	2	15	2	15	54	27	52	3
	3	24	15	10	28	10	9	25	28	24	3
	4	70	15	6	19	6	12	69	22	70	0
	5	56	15	11	15	11	15	56	21	56	12
	6	24	18	28	15	27	22	20	19	20	3
	7	55	28	29	21	27	21	50	28	51	12
	8	11	6	14	12	14	13	11	26	11	1
	9	32	15	30	21	30	15	22	46	24	4

TABLE : Results of the execution to obtain Datalog (time in ms)

RESULTS I

		REQUIEM(F)		Presto		Rapid		Clipper		kyrie	
$\mathcal{O}$	q	size	time	size	time	size	time	size	time	size	time
U	1	19	12	4	7	4	3	2	21	2	0
	2	47	16	2	9	2	9	49	19	47	3
	3	20	9	8	16	8	12	21	24	20	3
	4	64	15	3	12	3	3	63	18	64	3
	5	53	12	8	15	8	12	53	15	53	0
	6	20	12	19	6	21	15	16	18	16	9
	7	49	25	22	15	22	18	44	18	45	12
	8	10	9	13	6	13	9	10	20	10	3
	9	29	15	24	12	24	17	19	20	21	7
UX	1	22	6	7	10	7	3	5	27	5	6
	2	52	15	2	15	2	15	54	27	52	3
	3	24	15	10	28	10	9	25	28	24	3
	4	70	15	6	19	6	12	69	22	70	0
	5	56	15	11	15	11	15	56	21	56	12
	6	24	18	28	15	27	22	20	19	20	3
	7	55	28	29	21	27	21	50	28	51	12
	8	11	6	14	12	14	13	11	26	11	1
	9	32	15	30	21	30	15	22	46	24	4

TABLE : Results of the execution to obtain Datalog (time in ms)

RESULTS II

		REQUIEM(F)		Rapid		Prexto		Nyaya		kyrie	
$\mathcal{O}$	q	size	time	size	time	size	time	size	time	size	time
U	1	2	15	2	3	2	9	2	5	2	0
	2	1	103	1	15	1	15	1	1	1	34
	3	4	212	4	9	4	18	4	34	4	18
	4	2	3762	2	12	2	15	2	4	2	50
	5	10	13034	10	18	10	15	10	33	10	37
	6	29	47	29	28	28	12	40	1595	29	28
	7	42	797	42	37	70	18	54	670	42	43
	8	10	15	10	18	10	6	10	63	10	3
	9	960	1893	960	209	960	928	960	75135	960	1107
UX	1	5	15	5	12	5	12	5	22	5	9
	2	1	172	1	12	1	16	1	3	1	37
	3	12	2062	12	15	12	28	12	55	12	21
	4	5	31422	5	15	5	23	5	6	5	47
	5	25	91878	25	27	25	23	25	39	25	46
	6	323	468	323	106	448	178	348	2685	323	187
	7	1456	37212	224	81	280	75	264	852	224	121
	8	20	21	20	23	20	15	20	61	20	9
	9	4200	30506	4200	739	4200	21181	4200	366673	4200	16875

TABLE : Results of the execution to obtain UCQ (time in ms)

RESULTS II

		REQUIEM(F)		Rapid		Prexto		Nyaya		kyrie	
$\mathcal{O}$	q	size	time	size	time	size	time	size	time	size	time
U	1	2	15	2	3	2	9	2	5	2	0
	2	1	103	1	15	1	15	1	1	1	34
	3	4	212	4	9	4	18	4	34	4	18
	4	2	3762	2	12	2	15	2	4	2	50
	5	10	13034	10	18	10	15	10	33	10	37
	6	29	47	29	28	28	12	40	1595	29	28
	7	42	797	42	37	70	18	54	670	42	43
	8	10	15	10	18	10	6	10	63	10	3
	9	960	1893	960	209	960	928	960	75135	960	1107
UX	1	5	15	5	12	5	12	5	22	5	9
	2	1	172	1	12	1	16	1	3	1	37
	3	12	2062	12	15	12	28	12	55	12	21
	4	5	31422	5	15	5	23	5	6	5	47
	5	25	91878	25	27	25	23	25	39	25	46
	6	323	468	323	106	448	178	348	2685	323	187
	7	1456	37212	224	81	280	75	264	852	224	121
	8	20	21	20	23	20	15	20	61	20	9
	9	4200	30506	4200	739	4200	21181	4200	366673	4200	16875

TABLE : Results of the execution to obtain UCQ (time in ms)

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions I

A benchmark should consider how the input represents reality wrt:

- queries
  - syntax (`SELECT`, `COUNT`, `MAX`, ...)
  - expressiveness (CQ, UCQ, comparisons, arithmetic operations, ...)
  - shape (star shaped, linear, cyclic, ...)
  - size (number of atoms, number of clauses, triples...)
- ontologies
  - expressiveness (from DL-Lite$_R$ to...)
  - shape (flat, hierarchical, cyclic ...)
  - size (number of concepts, properties, individuals, ...)
- Additional information
  - mappings
  - ABox dependencies / EBox
  - caching for several queries

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

Conclusions II

A benchmark should consider what the output means wrt:

- Shape of rewritten queries
  - expressiveness
  - types of clauses
  - syntax with special characteristics
- Size of rewritten queries
  - number of clauses
  - number of atoms (and distinct atoms)
  - number of joins

REFERENCES I

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, October 2007. doi: 10.1007/s10817-007-9078-x. URL http://dx.doi.org/10.1007/s10817-007-9078-x.

Alexandros Chortaras, Despoina Trivela, and Giorgos Stamou. Optimized query rewriting for OWL 2 QL. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction – CADE-23*, volume 6803, pages 192–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-22437-9. URL http://www.springerlink.com/content/g8153m783k638210/.

Thomas Eiter, Magdalena Ortiz, Mantas \vSimkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for horn-SHIQ plus rules. In *Proc. of the 26th AAAI Conference on Artificial Intelligence, AAAI*, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewPDFInterstitial/4931/5263.

Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: Rewriting and optimization (extended version). *arXiv:1112.0343*, December 2011. URL http://arxiv.org/abs/1112.0343.

Martha Imprialou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking ontology-based query rewriting systems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewPDFInterstitial/4910/5270.

Jose Mora and Oscar Corcho. Engineering optimisations in query rewriting for OBDA. In *Proceedings of the 9th International Conference on Semantic Systems*, ICPS, pages 41–48, Graz, Austria, September 2013. ACM.

Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik. Efficient query answering for OWL 2. In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 489–504. Springer, 2009. URL http://dx.doi.org/10.1007/978-3-642-04930-9_31.

Riccardo Rosati. Prexto: Query rewriting under extensional constraints in DL-Lite. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, volume 7295 of *Lecture Notes in Computer Science*, pages 360–374. Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-30283-1. URL http://www.springerlink.com/content/1j61g52j78525175/abstract/.

Riccardo Rosati and Alessandro Almatelli. Improving query answering over DL-Lite ontologies. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*. AAAI Press, 2010. URL http://dblp.uni-trier.de/db/conf/kr/kr2010.html.

T. Venetis, G. Stoilos, and G. Stamou. Query rewriting under query extensions for OWL 2 QL ontologies. In *The 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011)*, page 59, 2011. URL http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/Workshops/SSWS/SSWS2011-Proceedings.pdf.

TOWARDS A SYSTEMATIC BENCHMARKING OF ONTOLOGY-BASED QUERY REWRITING SYSTEMS

José Mora and Óscar Corcho

{jmora, ocorcho}@fi.upm.es

Boadilla - October 10, 2013