Efficient Query Answering over Datalog with Existential Quantifiers

Nicola Leone, Marco Manna,
Giorgio Terracina and Pierfrancesco Veltri*

*Department of Mathematics, University of Calabria, IT

SEBD 2012 - 20th Italian Symposium on Advanced Database Systems

Outline

Outline

Context

Let $\mathcal{L}$ be an *ontology specification language*.

The Ontology-Based Query Answering Problem over $\mathcal{L}$

INPUT:

- A relational *database* D
- An ontological *theory* $\Sigma \in \mathcal{L}$
- A boolean conjunctive *query* q

QUESTION: Does $D \cup \Sigma \models q$ hold?

Context

Question

What are/should be the “shape” and the properties of $\mathcal{L}$?

Language $\mathcal{L}$ should balance *expressiveness* and *complexity*.

In particular it should possibly be:

1. intuitive and easy-to-understand;
2. QA-decidable (i.e., Query Answering should be decidable);
3. tractable for query answering;
4. powerful enough in terms of expressiveness;
5. suitable for an efficient implementation.

Context

The *Datalog*$^{\pm}$ family (Cali, Gottlob and Lukasiewicz 2008):

1. is based on *Datalog*$^{\exists}$;
2. generalizes some well known ontology specification languages;
3. is arousing increasing interest.

Example (1)

`∃Y father(X,Y) ← person(X)`	*Datalog*$^{\exists}$ rule
`person(Y) ← father(X,Y)`	*Datalog* rule

Example (2)

`A ⊑ R.B`	DL axiom
`∃Y R(X,Y) ∧ B(Y) ← A(X)`	*Datalog*$^{\pm}$ rule

Context

Common strengths

Known *Datalog*$^{\pm}$ classes are:

1. Intuitive and easy-to-understand (enjoy the simplicity of *Datalog*);
2. QA-decidable.

Local weaknesses/shortcomings

Currently, each *Datalog*$^{\pm}$ class misses at least one of the following properties:

3. Tractability;
4. Some useful expressive power (e.g. transitivity);
5. Suitability for an efficient implementation.

Main Contribution: *Shy*, a new *Datalog*$^{\exists}$ class

1. Intuitive and easy to understand
2. QA-Decidable
3. Tractable QA: P-Complete in data complexity
4. Expressive:
   - Includes both *Datalog* and *Linear-Datalog*$^{\exists}$
   - Supports the standard first-order semantics for unrestricted CQs with existential variables
   - Supports useful ontology properties
5. Suitable for an efficient implementation:
   - We implemented an efficient evaluator for *Shy* (DLV$^{\exists}$).
   - Experiments confirm the effectiveness for on-the-fly QA.

Outline

Datalog$^{\exists}$ Programs and BCQs

A *Datalog*$^{\exists}$ program *P* is a finite set of rules of the form:

$$\forall \mathbb{X} \exists \mathbb{Y} \ \mathbf{atom}_{[\mathbb{X}' \cup \mathbb{Y}]} \leftarrow \mathbf{conj}_{[\mathbb{X}]}$$

A *Boolean Conjunctive Query* (BCQ) *q* is a first-order expression of the form:

$$\exists \mathbb{Y} \ \mathbf{conj}_{[\mathbb{Y}]}$$

Query *q* is called *atomic* if **conj** consists of one atom.

Example

```
person(X) ← father(X,Y).
person(Y) ← father(X,Y).
∃Y father(X,Y) ← person(X).

∃Y person(Y)                                    atomic
∃X∃Y father(pierfrancesco,X), father(X,Y)       conjunctive
```

The CHASE (1)

Example

```
D = {run(gazelle), fastest(gazelle)}

r1 :  ∃Y pursues(Y,X) ← run(X).
r2 :  ∃Y slowerThan(Y,X) ← fastest(X).
r3 :  pursues(X,Z) ← pursues(X,Y), slowerThan(Z,Y).
r4 :  run(Y) ← pursues(X,Y).
```

The CHASE (2)

Example

D = {run(gazelle), fastest(gazelle)}

r_1 : ∃Y pursues(Y,**X**) ← run(**X**).
r_2 : ∃Y slowerThan(Y,X) ← fastest(X).
r_3 : pursues(X,Z) ← pursues(X,**Y**), slowerThan(Z,**Y**).
r_4 : run(Y) ← pursues(X,Y).

0 run(**gazelle**) fastest(**gazelle**)

r_1

1 pursues(φ_1,**gazelle**)

The CHASE (2)

Example

D = {run(gazelle), fastest(gazelle)}

r_1 : ∃Y pursues(Y,X) ← run(X).
r_2 : ∃Y slowerThan(Y,**X**) ← fastest(**X**).
r_3 : pursues(X,Z) ← pursues(X,**Y**), slowerThan(Z,**Y**).
r_4 : run(Y) ← pursues(X,Y).

0 run(**gazelle**) fastest(**gazelle**)

r_1 r_2

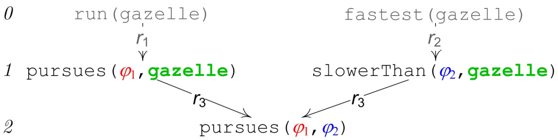
1 pursues(φ_1,**gazelle**) slowerThan(φ_2,**gazelle**)

The CHASE (2)

Example

```
D = {run(gazelle), fastest(gazelle)}
```

r_1 : ∃Y pursues(Y,X) ← run(X).
r_2 : ∃Y slowerThan(Y,X) ← fastest(X).
r_3 : pursues(X,Z) ← pursues(X,**Y**), slowerThan(Z,**Y**).
r_4 : run(Y) ← pursues(X,Y).

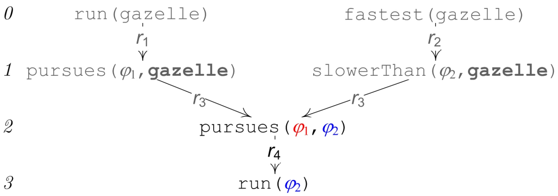


The CHASE (2)

Example

D = {run(gazelle), fastest(gazelle)}

r_1 : ∃Y pursues(Y,X) ← run(X).
r_2 : ∃Y slowerThan(Y,X) ← fastest(X).
r_3 : pursues(X,Z) ← pursues(X,**Y**), slowerThan(Z,**Y**).
r_4 : run(Y) ← pursues(X,Y).

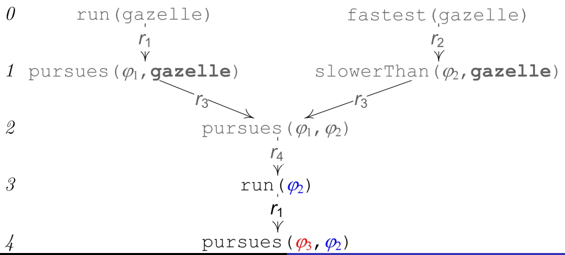


The CHASE (2)

Example

D = {run(gazelle), fastest(gazelle)}

r_1 : ∃Y pursues(Y,X) ← run(X).
r_2 : ∃Y slowerThan(Y,X) ← fastest(X).
r_3 : pursues(X,Z) ← pursues(X,**Y**), slowerThan(Z,**Y**).
r_4 : run(Y) ← pursues(X,Y).



The CHASE (3)

Question

Does the CHASE terminate on the following program?

```
person(pierfrancesco)
∃Y father(X,Y) ← person(X)
person(Y) ← father(X,Y)
```

Theorem

Query Answering over *Datalog*$^{\exists}$ is undecidable, in the general case.

Questions

CHASE may generate infinitely many (duplicate) homomorphic atoms

- Can we avoid duplicate generation obtaining *Datalog* fixpoint efficiency?
- Under which assumptions duplicate-free CHASE ensures sound and complete query answering?

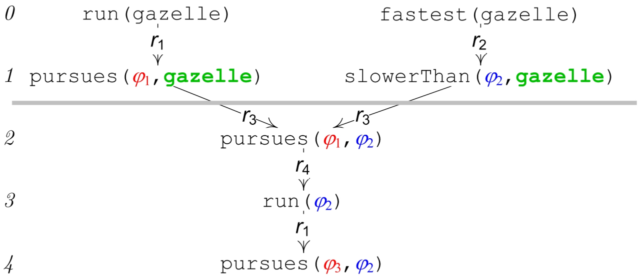
Outline

The PARSIMONIOUS-CHASE

Definition

For any *Datalog*$^{\exists}$ program P, PARSIMONIOUS-CHASE is the procedure resulting by forcing the CHASE to stop as soon as each atom produced in a level can be mapped homomorphically to someone else in previous levels.

The output of the PARSIMONIOUS-CHASE is denoted by $\boldsymbol{pChase}(P)$.



The PARSIMONIOUS-CHASE

Parsimonious Programs and Parsimonious Sets

A $\mathit{Datalog}^{\exists}$ program P is called *Parsimonious* if, for each $a \in \mathit{chase}(P)$, $\mathbf{pChase}(P)$ homomorphically entails a.

Parsimonious-Sets denotes the class of parsimonious programs.

Theorem

Let D be a database, P be a *parsimonious* program, and q be a Boolean *atomic* query. Then,

$$D \cup P \models q \ \ \mathit{iff} \ \ \mathbf{pChase}(D \cup P) \models q$$

Positive and Negative Results

Theorem

Atomic query answering against *Parsimonious-Sets* programs is decidable

Theorem

Checking whether a program is parsimonious is not decidable. In particular it is **coRE**-complete

Need for a recognizable class

Outline

Recognizable Parsimonious Programs: Shy

Shy is a subclass of *Datalog*$^\exists$ relying on the following intuition:

- *During a* CHASE*-run on a Shy program, nulls propagated body-to-head must not meet each other to join.*

Theorem

Shy is recognizable. Membership is polynomial-time doable.

Theorem

Shy is a subclass of *Parsimonious*

Theorem

Atomic Query Answering over *Shy* is decidable

Conjunctive queries over *Shy*

Question

Can we answer also conjunctive queries over *Shy*?

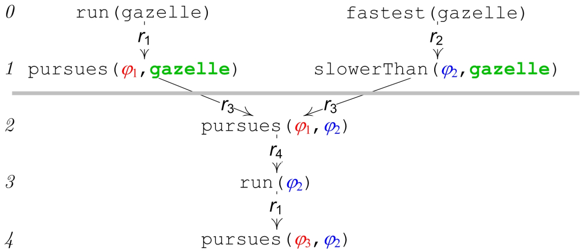
PARSIMONIOUS-CHASE alone doesn't work!

Conjunctive Queries over *Shy* (1)

Example

Using PARSIMONIOUS-CHASE to answer the following BCQ

$\exists$X$\exists$Y pursues(X,Y), slowerThan(Y,gazelle)

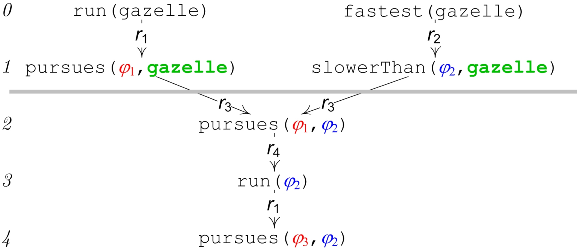


The answer to the query should be Yes!

Conjunctive Queries over *Shy* (2)

$\exists X \exists Y$ `pursues(X,Y), slowerThan(Y,gazelle)`

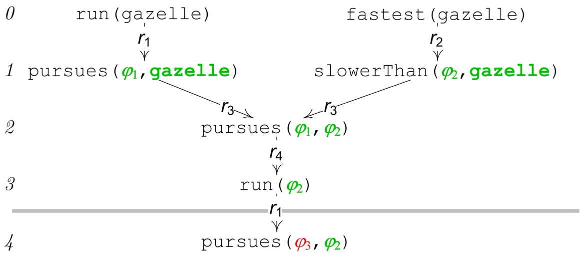
Let us both "promote" φ_1 and φ_2 (the nulls introduced in the first level) to constants, and "**resume**" the PARSIMONIOUS-CHASE execution.



Conjunctive Queries over *Shy* (3)

$\exists X \exists Y$ `pursues(X,Y), slowerThan(Y,gazelle)`

Let us both "promote" φ_1 and φ_2 (the nulls introduced in the first level) to constants, and "resume" the PARSIMONIOUS-CHASE execution.



Conjunctive Queries over *Shy* (4)

Definition

Let $\boldsymbol{pChase}(D \cup P, k)$ denote the output of PARSIMONIOUS-CHASE after k resumptions.

Theorem

Let D be a database, $P \in Shy$ and q be a BCQ with n different ($\exists$-quantified) variables. Then,

$$D \cup P \models q \;\; \textit{iff} \;\; \boldsymbol{pChase}(D \cup P, n) \models q$$

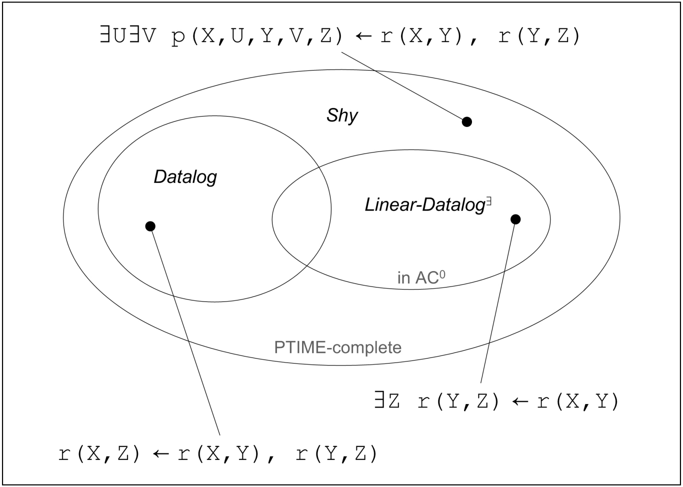
Decidability and Complexity

Theorem

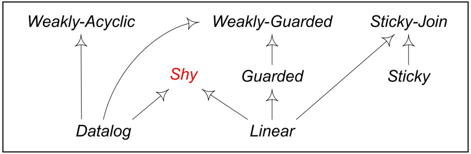
QA over *Shy* is decidable. In particular, it is

- **P**-complete in *data-complexity*
- **EXP**-complete in *combined-complexity*

Expressive Power (1)



Expressive Power (2)



Results

Shy includes *Datalog*

Shy includes *Linear-Datalog*$^{\exists}$

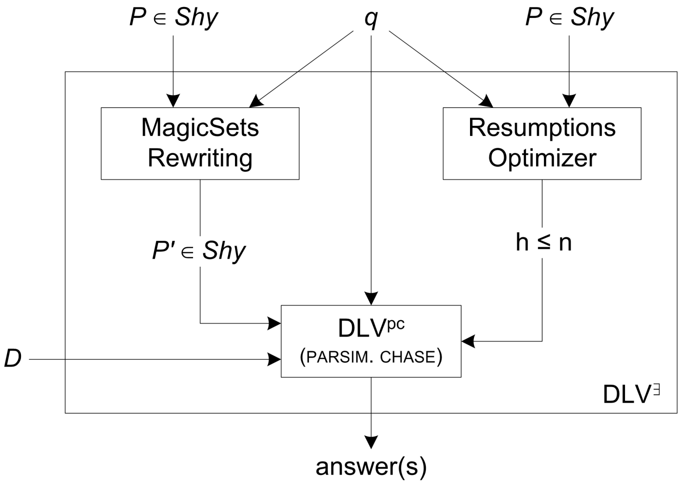
Shy and *Weakly-Acyclic* are uncomparable

Shy and *Sticky*/*Sticky-Join* are uncomparable

Shy and *Guarded*/*Weakly-Guarded* are uncomparable

Outline

DLV$^{\exists}$: Architecture



Comparison 1 (DLV$^\exists$ vs more expressive systems)

Data set	System	Full Infer.	# solved	Geom. Avg time
LUBM x10	DLV$^\exists$	17	14	2.87
	Pellet	27	14	84.48
	OWLIM-Lite	33	12	53.31
	OWLIM-SE	105	14	105.14
LUBM x30	DLV$^\exists$	55	14	9.70
	Pellet	–	0	–
	OWLIM-Lite	106	11	123.18
	OWLIM-SE	323	14	323.57
LUBM x50	DLV$^\exists$	93	14	16.67
	Pellet	–	0	–
	OWLIM-Lite	187	11	223.79
	OWLIM-SE	536	14	537.35

Running times for LUBM queries (sec) — Two hours timeout

Comparison 2 (DLV$^\exists$ vs less expressive systems)

Query #	Requiem+DLV			DLV^E		
	Load	Ans	Tot	Load	Ans	Tot
1	13,0	0,5	13,5	13,5	4,9	18,5
3	6,6	0,4	7,0	9,2	2,4	11,6
4	19,2	1,4	20,6	19,7	1,6	21,3
5	4,8	0,3	5,1	29,4	9,3	38,7
6	2,6	2,9	5,6	2,8	3,0	5,9
7	15,9	5,2	21,0	18,0	10,7	28,7
9	17,3	1,9	19,2	20,9	25,2	46,1
10	14,9	1,0	15,9	15,6	5,3	21,0
13	1,8	0,2	1,9	29,4	5,1	34,4
14	2,6	2,7	5,3	2,5	2,7	5,2
Arith. Avg	9,9	1,6	11,5	16,1	7,0	23,1
Geom. Avg	7,2	1,0	9,0	12,4	5,0	18,7

Running times for LUBM queries (sec) on the data set x50

The End

Thanks!

Context

Known *Datalog*$^{\pm}$ languages rely on three main paradigms:

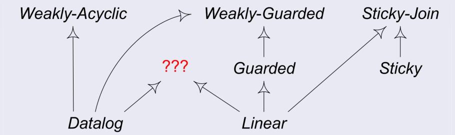
- *weak-acyclicity* $\rightsquigarrow$ "finite-model property"
  (Fagin, Kolaitis, Miller and Popa TCS05);
- *guardedness* $\rightsquigarrow$ "tree-model property"
  (Cali, Gottlob and Kifer KR08);
- *stickiness* $\rightsquigarrow$ "bounded-recursion property"
  (Cali, Gottlob, Pieris RR10).

Motivation (Complexity, Expressiveness, Implementations)

Tractability?

Datalog$^{\exists}$ class	*Data Complexity*
Weakly-Guarded	**EXP**-complete
Guarded, *Weakly-Acyclic*	**P**-complete
???	**P**-complete
Sticky, *Sticky-Join*	in $\mathbf{AC}_0$
Linear	in $\mathbf{AC}_0$

Expressive Power?



Implementation?

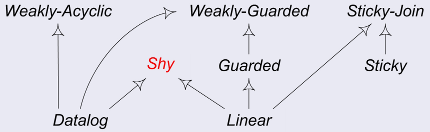
Currently, there in no system that implements Query Answering over *Weakly-Guarded* or *Guarded* programs.

Contribution (Complexity, Expressiveness, Implementations)

Tractability?

Datalog$^{\exists}$ class	*Data Complexity*
Weakly-Guarded	**EXP**-complete
Guarded, *Weakly-Acyclic*	**P**-complete
Shy	**P**-complete
Sticky, *Sticky-Join*	in $\mathbf{AC}_0$
Linear	in $\mathbf{AC}_0$

Expressive Power?



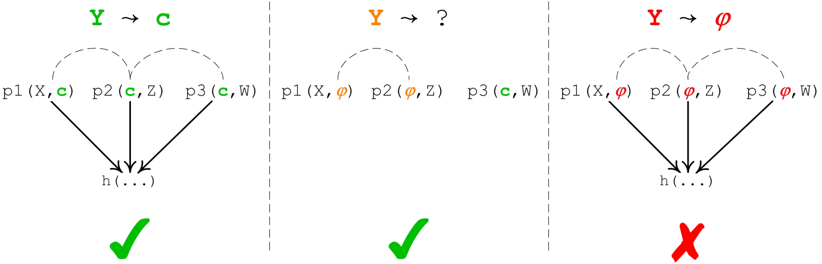
Implementation?

Query Answering over *Shy* programs can already be performed by DLV$^{\exists}$.

Shy Property 1

If a variable **Y** occurs in more than one body atom of a rule $r \in P$, then any σ mapping $\boldsymbol{body}(r)$ into $\boldsymbol{chase}(D \cup P)$ never maps **Y** into a null.

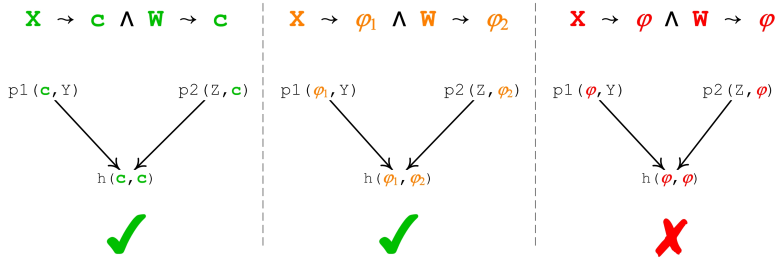
h(...) ← p1(X,**Y**), p2(**Y**,Z), p3(**Y**,W)



Shy Property 2

If two variables **X**,**W** occur in two different body atoms of a rule $r \in P$ and also in $\textbf{\textit{head}}(r)$, then any σ mapping $\textbf{\textit{body}}(r)$ into $\textbf{\textit{chase}}(D \cup P)$ never maps **X**,**W** into the same null.

`h(X,W) ← p1(X,Y), p2(Z,W)`



Recognizability

Proposition

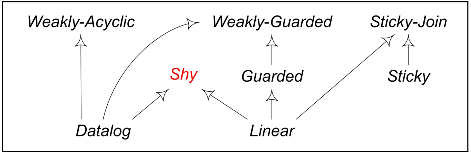
Checking whether a *Datalog*$^{\exists}$ program is *Shy* is doable in polynomial-time.

Proof (Sketch)

The marking procedure:

1. propagates polynomially many “representative” nulls; *and*
2. reaches a fixpoint in polynomially many steps.

Expressive Power (2)



Question

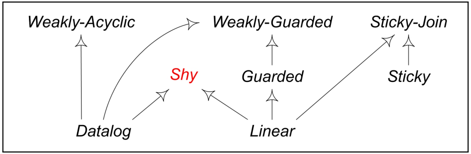
Why are *Shy* and *Weakly-Acyclic* uncomparable?

Intuitively, the following *Shy* program is not *Weakly-Acyclic* since its universal models have infinite size.

Example

```
∃Y father(X,Y) ← person(X).
person(Y) ← father(X,Y).
```

Expressive Power (3)



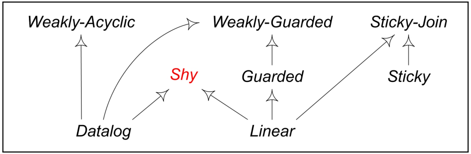
Question

Why are *Shy* and *Sticky*/*Sticky-Join* uncomparable?

Intuitively,

1. *Sticky-Join* does not capture transitivity.
2. *Sticky* programs allow some joins on nulls.

Expressive Power (4)



Question

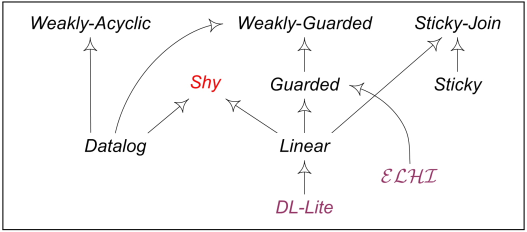
Why are *Shy* and *Guarded*/*Weakly-Guarded* uncomparable?

Intuitively,

1. *Shy* does not enjoy the tree-model property (There exists a shy program whose chase hypergraph has infinite treewidth.) .
2. *Guarded* programs allow some joins on nulls.

Expressive Power (5)

DLs VS *Shy*



Expressive Power (6)

DL Axiom	Shy Rule
$A \sqsubseteq B$	$p_A(X) \rightarrow p_B(X)$
$A \sqcap B \sqsubseteq C$	$p_A(X), p_B(X) \rightarrow p_C(X)$
$A \sqsubseteq \exists R.B$	$p_A(X) \rightarrow \exists Y\, p_R(X,Y), p_B(Y)$
$\exists R.A \sqsubseteq B$	$p_R(X,Y), p_A(Y) \rightarrow p_B(X)$
$R \sqsubseteq S$	$p_R(X,Y) \rightarrow p_S(X,Y)$
$R \sqsubseteq S^-$	$p_R(X,Y) \rightarrow p_S(Y,X)$
$R+$	$p_R(X,Y), p_R(Y,Z) \rightarrow p_R(X,Z)$

DLs VS *Shy*; A, B, C are concept names, R, S are role names.