Don't Repeat Yourself: Termination of the Skolem Chase on Disjunctive Existential Rules

Lukas Gerlach

Knowledge-Based Systems Group
Technische Universität Dresden, Germany

08.10.2020

Motivation

- Reasoning over *Knowledge Bases*

Motivation

- Reasoning over *Knowledge Bases*
- Expressivity of *Disjunctive Existential Rules*

MOTIVATION

- Reasoning over *Knowledge Bases*
- Expressivity of *Disjunctive Existential Rules*
- Efficiency of the *Skolem Chase* (ASP Solvers)

Definition

A *knowledge base* is a pair $\langle R, I\rangle$ of a rule set R and an instance I.

Definition

A *knowledge base* is a pair $\langle R, I \rangle$ of a rule set R and an instance I.

Definition

An *instance* is a set of function free *facts* (ground atoms).

Knowledge Bases

Definition

A *knowledge base* is a pair $\langle R, I\rangle$ of a rule set R and an instance I.

Definition

An *instance* is a set of function free *facts* (ground atoms).

Definition

A *(disjunctive existental) rule* ρ is an expression of the form

$$\forall\vec{x}\forall\vec{y}.[B_\rho(\vec{x},\vec{y}) \rightarrow \bigvee_{i=1}^{n} \exists\vec{z}_i.H^i_\rho(\vec{x}_i,\vec{z}_i)]$$

where B_ρ and H^i_ρ are conjunctions of atoms without function symbols or constants; $\vec{x}, \vec{y}$, and $\vec{z}_i$ are pairwise disjoint lists of variables; and $\bigcup_{i=1}^{n} \vec{x}_i = \vec{x}$.

Definition

A *boolean conjunctive query (BCQ)* is an expression of the form $\exists\vec{z}.\varphi(\vec{z})$ where φ is a conjunction of function free atoms.

Definition

A *boolean conjunctive query (BCQ)* is an expression of the form $\exists\vec{z}.\varphi(\vec{z})$ where φ is a conjunction of function free atoms.

Example

Consider the following instance I, rule set R and BCQ σ:

$$I := \{ \mathit{Pizza}(\mathit{myPizza}) \}$$

Definition

A *boolean conjunctive query (BCQ)* is an expression of the form $\exists\vec{z}.\varphi(\vec{z})$ where φ is a conjunction of function free atoms.

Example

Consider the following instance I, rule set R and BCQ σ:

$$I := \{\, Pizza(myPizza) \,\}$$
$$R := \{\, Pizza(x) \rightarrow InFridge(x) \vee \exists z.(Service(z) \wedge Delivers(z, x)) \,\}$$

Definition

A *boolean conjunctive query (BCQ)* is an expression of the form $\exists \vec{z}.\varphi(\vec{z})$ where φ is a conjunction of function free atoms.

Example

Consider the following instance I, rule set R and BCQ σ:

$$I := \{\, Pizza(myPizza) \,\}$$

$$R := \{\, Pizza(x) \rightarrow InFridge(x) \vee \exists z.(Service(z) \wedge Delivers(z, x)) \,\}$$

$$\sigma := \exists z.(Service(z) \wedge Delivers(z, myPizza))$$

Definition

A *boolean conjunctive query (BCQ)* is an expression of the form $\exists \vec{z}.\varphi(\vec{z})$ where φ is a conjunction of function free atoms.

Example

Consider the following instance I, rule set R and BCQ σ:

$$I := \{\, Pizza(myPizza) \,\}$$

$$R := \{\, Pizza(x) \rightarrow InFridge(x) \vee \exists z.(Service(z) \wedge Delivers(z, x)) \,\}$$

$$\sigma := \exists z.(Service(z) \wedge Delivers(z, myPizza))$$

Is σ entailed by $\langle R, I \rangle$?

How to solve this in general?

Definition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is *entailed* by a knowledge base $\mathcal{K}$ if, for each first order model M of $\mathcal{K}$, there exists a substitution θ such that $\varphi\theta \subseteq M$.

Definition

A BCQ $\sigma := \exists\vec{z}.\varphi(\vec{z})$ is *entailed* by a knowledge base $\mathcal{K}$ if, for each first order model M of $\mathcal{K}$, there exists a substitution θ such that $\varphi\theta \subseteq M$.

Problems:

1. The number of models may be infinite.
2. Individual models may be infinite in size.

How to solve this in general?

Definition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is *entailed* by a knowledge base $\mathcal{K}$ if, for each first order model M of $\mathcal{K}$, there exists a substitution θ such that $\varphi\theta \subseteq M$.

Problems:

1. The number of models may be infinite.
2. Individual models may be infinite in size.

Proposition

BCQ entailment is undecidable [Beeri and Vardi, 1981].

Do we really need every model?

Definition

A *universal model set* [Bourhis et al., 2016] of a knowledge base $\mathcal{K}$ is a set of models $\mathcal{U}$, such that for each model M for $\mathcal{K}$ there exists a homomorphism from some model in $\mathcal{U}$ to M.

Do we really need every model?

Definition

A *universal model set* [Bourhis et al., 2016] of a knowledge base $\mathcal{K}$ is a set of models $\mathcal{U}$, such that for each model M for $\mathcal{K}$ there exists a homomorphism from some model in $\mathcal{U}$ to M.

Proposition

A BCQ $\sigma := \exists\vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proposition

A BCQ $\sigma := \exists\vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.
"$\Rightarrow$":
If σ is entailed by $\mathcal{K}$, then it is entailed for every model of $\mathcal{K}$.

Do we really need every model?

Proposition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.
"$\Leftarrow$":
Consider a model M of $\mathcal{K}$.

Do we really need every model?

Proposition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.
"$\Leftarrow$":
Consider a model M of $\mathcal{K}$.

1. For each model U in some universal model set $\mathcal{U}$ of $\mathcal{K}$, there exists a substitution θ with $\varphi\theta \subseteq U$.

Proposition

A BCQ $\sigma := \exists\vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.

"$\Leftarrow$":

Consider a model M of $\mathcal{K}$.

1. For each model U in some universal model set $\mathcal{U}$ of $\mathcal{K}$, there exists a substitution θ with $\varphi\theta \subseteq U$.
2. By (1): There exists a model $U \in \mathcal{U}$ such that there exists a homomorphism τ from U to M.

Proposition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.
"$\Leftarrow$":
Consider a model M of $\mathcal{K}$.

1. For each model U in some universal model set $\mathcal{U}$ of $\mathcal{K}$, there exists a substitution θ with $\varphi\theta \subseteq U$.
2. By (1): There exists a model $U \in \mathcal{U}$ such that there exists a homomorphism τ from U to M.
3. By (1) and (2): $\tau \circ \theta$ is a substitution with $\varphi(\tau \circ \theta) \subseteq M$.

Proposition

A BCQ $\sigma := \exists\vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

Proof.
"$\Leftarrow$":
Consider a model M of $\mathcal{K}$.

1. For each model U in some universal model set $\mathcal{U}$ of $\mathcal{K}$, there exists a substitution θ with $\varphi\theta \subseteq U$.
2. By (1): There exists a model $U \in \mathcal{U}$ such that there exists a homomorphism τ from U to M.
3. By (1) and (2): $\tau \circ \theta$ is a substitution with $\varphi(\tau \circ \theta) \subseteq M$.
4. By (3): σ is entailed by $\mathcal{K}$.

□

Do we really need every model?

Definition

A *universal model set* [Bourhis et al., 2016] of a knowledge base $\mathcal{K}$ is a set of models $\mathcal{U}$, such that for each model M for $\mathcal{K}$ there exists a homomorphism from some model in $\mathcal{U}$ to M.

Proposition

A BCQ $\sigma := \exists \vec{z}.\varphi(\vec{z})$ is entailed by a knowledge base $\mathcal{K}$ iff it is entailed by each model in some universal model set of $\mathcal{K}$.

We study an algorithm that should compute a finite universal model set containing only finite models.

Chasing a Universal Model Set

General Chase Procedure

- Input: Knowledge Base $\mathcal{K}$
- Procedure: Apply rules in $\mathcal{K}$ exhaustively, until no new facts are obtained. Consider the head disjuncts individually by branching out on them.
- Output: Universal Model Set of $\mathcal{K}$

General Chase Procedure

- Input: Knowledge Base $\mathcal{K}$
- Procedure: Apply rules in $\mathcal{K}$ exhaustively, until no new facts are obtained. Consider the head disjuncts individually by branching out on them.
- Output: Universal Model Set of $\mathcal{K}$

Questions

Chasing a Universal Model Set

General Chase Procedure

- Input: Knowledge Base $\mathcal{K}$
- Procedure: Apply rules in $\mathcal{K}$ exhaustively, until no new facts are obtained. Consider the head disjuncts individually by branching out on them.
- Output: Universal Model Set of $\mathcal{K}$

Questions

1. How to apply rules?

General Chase Procedure

- Input: Knowledge Base $\mathcal{K}$
- Procedure: Apply rules in $\mathcal{K}$ exhaustively, until no new facts are obtained. Consider the head disjuncts individually by branching out on them.
- Output: Universal Model Set of $\mathcal{K}$

Questions

1. How to apply rules?
2. Do we indeed obtain a finite set of finite models?

How to apply rules?

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set *F*, the trigger λ is:

How to apply rules?

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set F, the trigger λ is:

- *active* if $B_\rho\theta \subseteq F$,

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set F, the trigger λ is:

- *active* if $B_\rho\theta \subseteq F$,
- *obsolete* if $sk(H^i_\rho)\theta \subseteq F$ for some $1 \leq i \leq branch(\rho)$,

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set F, the trigger λ is:

- *active* if $B_\rho\theta \subseteq F$,
- *obsolete* if $sk(H_\rho^i)\theta \subseteq F$ for some $1 \leq i \leq branch(\rho)$, and
- *applicable* to F if it is active and not obsolete.

How to apply rules?

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set F, the trigger λ is:

- *active* if $B_\rho \theta \subseteq F$,
- *obsolete* if $sk(H^i_\rho)\theta \subseteq F$ for some $1 \leq i \leq branch(\rho)$, and
- *applicable* to F if it is active and not obsolete.

If λ is applicable to F, then the *application* of λ on F is defined as the set of fact sets:

$$\lambda(F) := \{ F \cup sk(H^i_\rho)\theta \mid 1 \leq i \leq branch(\rho) \}$$

How to apply rules?

Definition

A *trigger* is a pair $\lambda := \langle \rho, \theta \rangle$ of a rule ρ and a substitution θ.
In the context of a fact set F, the trigger λ is:

- *active* if $B_\rho \theta \subseteq F$,
- *obsolete* if $sk(H^i_\rho)\theta \subseteq F$ for some $1 \leq i \leq branch(\rho)$, and
- *applicable* to F if it is active and not obsolete.

If λ is applicable to F, then the *application* of λ on F is defined as the set of fact sets:

$$\lambda(F) := \{\, F \cup sk(H^i_\rho)\theta \mid 1 \leq i \leq branch(\rho) \,\}$$

Those applications can be implemented using ASP-solvers.

Example

Consider the following instance and rule set:

$$I := \{ \mathit{Pizza}(\mathit{myPizza}) \}$$

$$\rho := \mathit{Pizza}(x) \rightarrow \mathit{InFridge}(x) \vee (\mathit{Service}(f^z(x)) \wedge \mathit{Delivers}(f^z(x), x))$$

How to apply rules?

Example

Consider the following instance and rule set:

$$I := \{\, Pizza(myPizza) \,\}$$

$$\rho := Pizza(x) \to InFridge(x) \vee (Service(f^z(x)) \wedge Delivers(f^z(x), x))$$

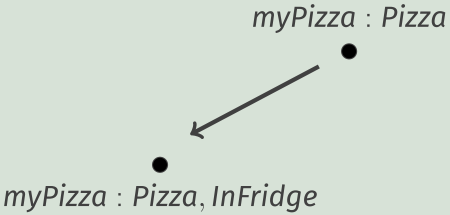
How to apply rules?

Example

Consider the following instance and rule set:

$$I := \{ \mathit{Pizza}(\mathit{myPizza}) \}$$

$$\rho := \mathit{Pizza}(x) \to \mathit{InFridge}(x) \vee (\mathit{Service}(f^z(x)) \wedge \mathit{Delivers}(f^z(x), x))$$

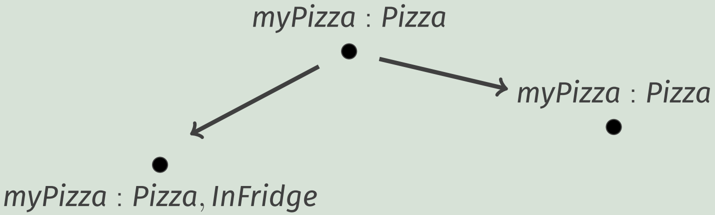


Example

Consider the following instance and rule set:

$$I := \{\, Pizza(myPizza) \,\}$$

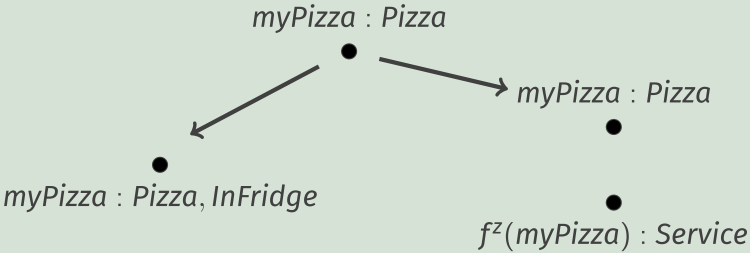
$$\rho := Pizza(x) \rightarrow InFridge(x) \vee (Service(f^z(x)) \wedge Delivers(f^z(x), x))$$



Example

Consider the following instance and rule set:

$$I := \{\, Pizza(myPizza) \,\}$$
$$\rho := Pizza(x) \rightarrow InFridge(x) \vee (Service(f^z(x)) \wedge Delivers(f^z(x), x))$$

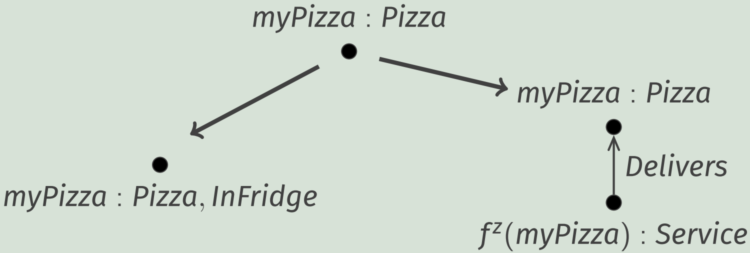


Example

Consider the following instance and rule set:

$$I := \{ \mathit{Pizza}(\mathit{myPizza}) \}$$

$$\rho := \mathit{Pizza}(x) \rightarrow \mathit{InFridge}(x) \vee (\mathit{Service}(f^z(x)) \wedge \mathit{Delivers}(f^z(x), x))$$

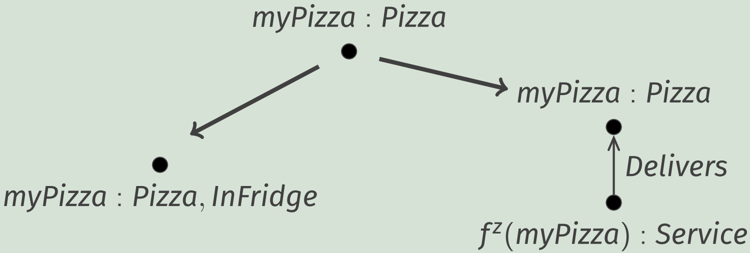


Example

Consider the following instance and rule set:

$$I := \{\, Pizza(myPizza) \,\}$$

$$\rho := Pizza(x) \to InFridge(x) \vee (Service(f^z(x)) \wedge Delivers(f^z(x), x))$$



The BCQ $\exists z.(Service(z) \wedge Delivers(z, myPizza))$ is not entailed.

General Chase Procedure

- Input: Knowledge Base $\mathcal{K}$
- Procedure: Apply rules in $\mathcal{K}$ exhaustively, until no new facts are obtained. Consider the head disjuncts individually by branching out on them.
- Output: Universal Model Set of $\mathcal{K}$

Questions

1. How to apply rules?
2. Do we indeed obtain a finite set of finite models?

Do we indeed obtain a finite set of finite models?

Not always: The chase may not terminate on a knowledge base.

Do we indeed obtain a finite set of finite models?

Not always: The chase may not terminate on a knowledge base.

Definition

A knowledge base $\mathcal{K}$ is *terminating* if no new facts can be obtained at some point in the chase computation.

Do we indeed obtain a finite set of finite models?

Not always: The chase may not terminate on a knowledge base.

Definition

A knowledge base $\mathcal{K}$ is *terminating* if no new facts can be obtained at some point in the chase computation.
A rule set R is *terminating* if $\langle R, I \rangle$ is terminating for every instance I.

Do we indeed obtain a finite set of finite models?

Not always: The chase may not terminate on a knowledge base.

Definition

A knowledge base $\mathcal{K}$ is *terminating* if no new facts can be obtained at some point in the chase computation.
A rule set R is *terminating* if $\langle R, I \rangle$ is terminating for every instance I.

Problem: Knowledge base termination and rule set termination are undecidable [Beeri and Vardi, 1981, Deutsch et al., 2008, Gogacz and Marcinkowski, 2014].

Do we indeed obtain a finite set of finite models?

Not always: The chase may not terminate on a knowledge base.

Definition

A knowledge base $\mathcal{K}$ is *terminating* if no new facts can be obtained at some point in the chase computation.
A rule set R is *terminating* if $\langle R, I \rangle$ is terminating for every instance I.

Problem: Knowledge base termination and rule set termination are undecidable [Beeri and Vardi, 1981, Deutsch et al., 2008, Gogacz and Marcinkowski, 2014].

Example

The following rule set is not terminating.

$$Pizza(x) \to Last(x) \lor (NextOrder(x, f^z(x)) \land Pizza(f^z(x)))$$

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Existing notions:

- MFA [Cuenca Grau et al., 2013] - Skolem Chase
- RMFA [Carral et al., 2017] - Restricted Chase

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Existing notions:

- MFA [Cuenca Grau et al., 2013] - Skolem Chase
- RMFA [Carral et al., 2017] - Restricted Chase

Why not use MFA?

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Existing notions:

- MFA [Cuenca Grau et al., 2013] - Skolem Chase
- RMFA [Carral et al., 2017] - Restricted Chase

Why not use MFA?
MFA is not defined for rule sets with disjunctions.

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Existing notions:

- MFA [Cuenca Grau et al., 2013] - Skolem Chase
- RMFA [Carral et al., 2017] - Restricted Chase

Why not use MFA?
MFA is not defined for rule sets with disjunctions.

Why not use RMFA?

Definition

Acyclicity Notions are sufficient conditions for rule set termination.

Existing notions:

- MFA [Cuenca Grau et al., 2013] - Skolem Chase
- RMFA [Carral et al., 2017] - Restricted Chase

Why not use MFA?
MFA is not defined for rule sets with disjunctions.

Why not use RMFA?
RMFA is not sound for the disjunctive skolem chase. (We can still use some ideas.)

Better safe than sorry

General idea of MFA: Compute chase on critical instance $I_R^\star$ and check for cyclic terms.

Better safe than sorry

General idea of MFA: Compute chase on critical instance $I_R^\star$ and check for cyclic terms.

Theorem

A rule set R without disjunctions is terminating if and only if $\langle R, I_R^\star \rangle$ is terminating [Marnette, 2009].

Better safe than sorry

General idea of MFA: Compute chase on critical instance $I_R^\star$ and check for cyclic terms.

Theorem

A rule set R without disjunctions is terminating if and only if $\langle R, I_R^\star \rangle$ is terminating [Marnette, 2009].

This approach does not work for disjunctive existential rules.

General idea of MFA: Compute chase on critical instance $I_R^\star$ and check for cyclic terms.

Theorem

A rule set R without disjunctions is terminating if and only if $\langle R, I_R^\star\rangle$ is terminating [Marnette, 2009].

This approach does not work for disjunctive existential rules.

Example

Consider the rule set R and its critical instance $I_R^\star$:

$$R = \{\, Pizza(x) \to Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))\,\}$$
$$I_R^\star = \{\, Pizza(\star), Last(\star), NextOrder(\star, \star)\,\}$$

The knowledge base $\langle R, I_R^\star\rangle$ is terminating but R is not.

Naive Fix

Treat disjunctions as conjunctions.

Naive Fix

Treat disjunctions as conjunctions.

Proposition

Consider a rule set R and a rule set R' that results from R by replacing disjunctions with conjunctions. If R' is terminating, then R is terminating.

Naive Fix

Treat disjunctions as conjunctions.

Proposition

Consider a rule set R and a rule set R' that results from R by replacing disjunctions with conjunctions. If R' is terminating, then R is terminating.

Though, this is not satisfactory.

Naive Fix

Treat disjunctions as conjunctions.

Proposition

Consider a rule set R and a rule set R' that results from R by replacing disjunctions with conjunctions. If R' is terminating, then R is terminating.

Though, this is not satisfactory.

Example

The following rule set is terminating but it does not terminate if we replace the disjunction by a conjunction:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

Generalise Obsoleteness

How to improve upon the naive fix?

Generalise Obsoleteness

How to improve upon the naive fix?

Problem: Obsoleteness is too restrictive.
Can we identify triggers that are obsolete independent of the starting instance?

Generalise Obsoleteness

How to improve upon the naive fix?

Problem: Obsoleteness is too restrictive.
Can we identify triggers that are obsolete independent of the starting instance?

Definition

A trigger $\langle \rho, \theta \rangle$ is *blocked* if it is obsolete w.r.t. to the set of facts that are "necessarily involved" in the derivation of $B_\rho\theta$.

Generalise Obsoleteness

How to improve upon the naive fix?

Problem: Obsoleteness is too restrictive.
Can we identify triggers that are obsolete independent of the starting instance?

Definition

A trigger $\langle \rho, \theta \rangle$ is *blocked* if it is obsolete w.r.t. to the set of facts that are "necessarily involved" in the derivation of $B_\rho\theta$.

Theorem

In the context of a rule set: If a trigger λ is blocked, then λ is not applicable to any fact set occurring in the chase on any instance.

Generalise Obsoleteness

What does "necessarily involved" mean?

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \to Last(x) \lor (NextOrder(x, f^z(x)) \land Pizza(f^z(x)))$$
$$NextOrder(y, x) \to Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.
To derive $f^z(\star)$, $\langle \rho_1, \{ x \mapsto \star \} \rangle$ was applied before.

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.
To derive $f^z(\star)$, $\langle \rho_1, \{ x \mapsto \star \} \rangle$ was applied before.
The following fact set has necessarily been involved:

$$\{ Pizza(\star),$$

Generalise Obsoleteness

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \to Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \to Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.
To derive $f^z(\star)$, $\langle \rho_1, \{ x \mapsto \star \} \rangle$ was applied before.
The following fact set has necessarily been involved:

$$\{ Pizza(\star), NextOrder(\star, f^z(\star)),$$

Generalise Obsoleteness

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.
To derive $f^z(\star)$, $\langle \rho_1, \{ x \mapsto \star \} \rangle$ was applied before.
The following fact set has necessarily been involved:

$$\{ Pizza(\star), NextOrder(\star, f^z(\star)), Pizza(f^z(\star)),$$

Generalise Obsoleteness

What does "necessarily involved" mean?

Example

Consider the following rule set:

$$\rho_1 : Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

Consider the trigger $\langle \rho_1, \{ x \mapsto f^z(\star) \} \rangle$.
To derive $f^z(\star)$, $\langle \rho_1, \{ x \mapsto \star \} \rangle$ was applied before.
The following fact set has necessarily been involved:

$$\{ Pizza(\star), NextOrder(\star, f^z(\star)), Pizza(f^z(\star)), Last(f^z(\star)) \}$$

Definition

For a rule set R, we define DMFA(R) to be the smallest fact set such that $I^{\star}_{R} \subseteq$ DMFA(R) and, for every trigger $\langle \rho, \theta \rangle$ with $\rho \in R$ that is active w.r.t. DMFA(R) and not blocked, we have $sk(H^{i}_{\rho})\theta \subseteq$ DMFA(R) for all $1 \leq i \leq branch(\rho)$.

Compute a generalized Chase result

Definition

For a rule set R, we define DMFA(R) to be the smallest fact set such that $I^{\star}_{R} \subseteq$ DMFA(R) and, for every trigger $\langle \rho, \theta \rangle$ with $\rho \in R$ that is active w.r.t. DMFA(R) and not blocked, we have $sk(H^{i}_{\rho})\theta \subseteq$ DMFA(R) for all $1 \leq i \leq branch(\rho)$.

Definition

R is DMFA if DMFA(R) does not contain a cyclic term.

Compute a generalized Chase result

Definition

For a rule set R, we define DMFA(R) to be the smallest fact set such that $I^{\star}_{R} \subseteq$ DMFA(R) and, for every trigger $\langle \rho, \theta \rangle$ with $\rho \in R$ that is active w.r.t. DMFA(R) and not blocked, we have $sk(H^{i}_{\rho})\theta \subseteq$ DMFA(R) for all $1 \leq i \leq branch(\rho)$.

Definition

R is DMFA if DMFA(R) does not contain a cyclic term.

Theorem

If a rule set R is DMFA, then R is terminating.

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.
Let σ be a mapping over constants with $\sigma(c) := \star$ for all constants c.

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.
Let σ be a mapping over constants with $\sigma(c) := \star$ for all constants c.

1. There exists an instance I such that $\langle R, I\rangle$ is not terminating.

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.
Let σ be a mapping over constants with $\sigma(c) := \star$
for all constants c.

1. There exists an instance I such that $\langle R, I \rangle$ is not terminating.
2. By (1): The chase sequence of $\langle R, I \rangle$ contains a cyclic term t.

Compute a generalized Chase result

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.
Let σ be a mapping over constants with $\sigma(c) := \star$
for all constants c.

1. There exists an instance I such that $\langle R, I \rangle$ is not terminating.
2. By (1): The chase sequence of $\langle R, I \rangle$ contains a cyclic term t.
3. By (2): The cyclic term $\sigma(t)$ is in DMFA(R) (via induction over the chase sequence).

Theorem

If a rule set R is DMFA, then R is terminating.

Proof.
We show that R is not DMFA if R is not terminating.
Let σ be a mapping over constants with $\sigma(c) := \star$ for all constants c.

1. There exists an instance I such that $\langle R, I \rangle$ is not terminating.
2. By (1): The chase sequence of $\langle R, I \rangle$ contains a cyclic term t.
3. By (2): The cyclic term $\sigma(t)$ is in DMFA(R) (via induction over the chase sequence).
4. By (3): R is not DMFA.

□

Compute a generalized Chase result

Example

R:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

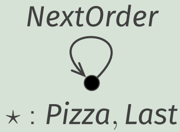
Compute a generalized Chase result

Example

R:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

DMFA(R):

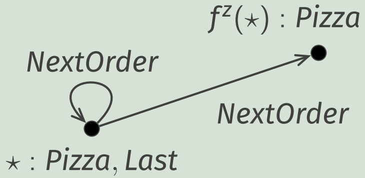


Example

R:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

DMFA(R):

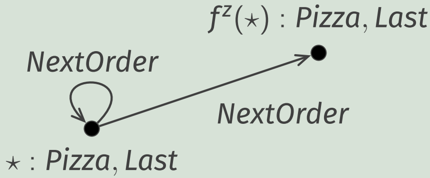


Example

R:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

DMFA(R):

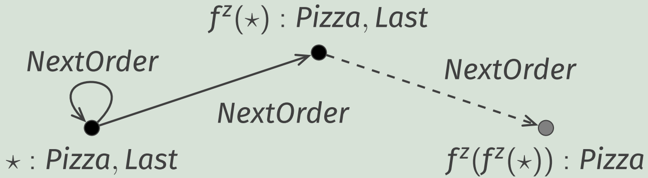


Example

R:

$$Pizza(x) \rightarrow Last(x) \vee (NextOrder(x, f^z(x)) \wedge Pizza(f^z(x)))$$
$$NextOrder(y, x) \rightarrow Last(x)$$

DMFA(R):



Hierarchy of Acyclicity Notions

Disjunctive Existential Rule Sets

Disjunctive Existential Rule Sets

Terminating w.r.t. restricted chase

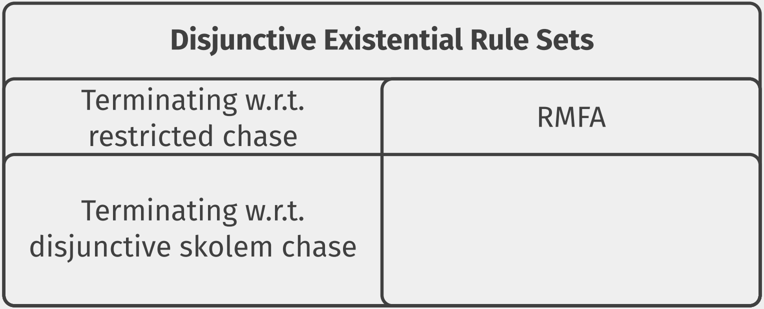
Hierarchy of Acyclicity Notions

Disjunctive Existential Rule Sets

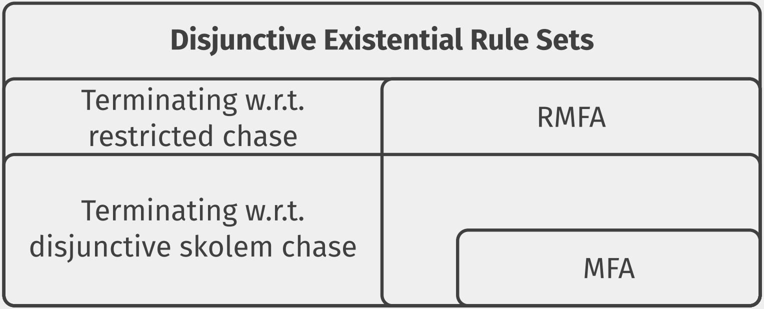
Terminating w.r.t.
restricted chase

Terminating w.r.t.
disjunctive skolem chase

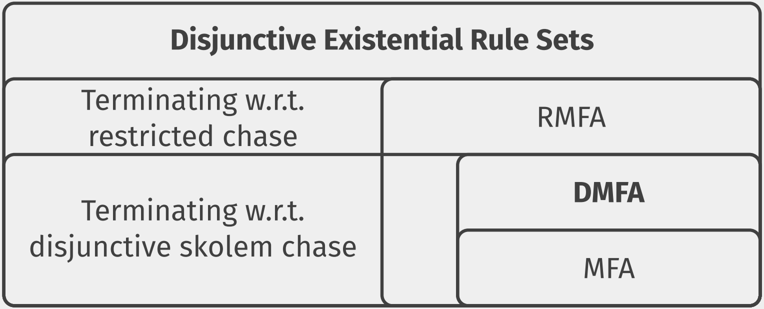
Hierarchy of Acyclicity Notions



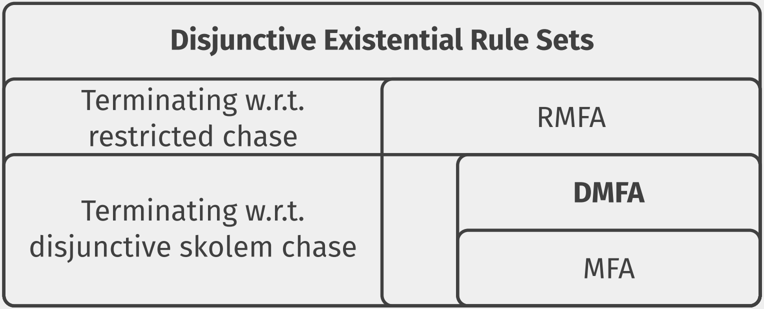
Hierarchy of Acyclicity Notions



Hierarchy of Acyclicity Notions



Hierarchy of Acyclicity Notions



Example

The following singleton rule set is contained in the blank space:

$$P(y, x) \wedge Q(y) \rightarrow \exists z.(P(x, z) \wedge P(z, x))$$

Complexity of the DMFA Check

Theorem

Checking if a rule set R is DMFA is 2ExpTime-complete.

Theorem

Checking if a rule set R is DMFA is 2ExpTime-complete.

Proof. (Sketch)
(Membership): We compute DMFA(R) step by step.

Complexity of the DMFA Check

Theorem

Checking if a rule set R is DMFA is 2ExpTime-complete.

Proof. (Sketch)
(Membership): We compute DMFA(R) step by step. The number of facts without cyclic terms is at most doubly exponential.

Complexity of the DMFA Check

Theorem

Checking if a rule set R is DMFA is 2ExpTime-complete.

Proof. (Sketch)
(Membership): We compute DMFA(R) step by step. The number of facts without cyclic terms is at most doubly exponential. Therefore, there are at most doubly exponentially many steps of which each is computable in 2ExpTime.

Complexity of the DMFA Check

Theorem

Checking if a rule set R is DMFA is 2ExpTime-complete.

Proof. (Sketch)
(Membership): We compute DMFA(R) step by step. The number of facts without cyclic terms is at most doubly exponential. Therefore, there are at most doubly exponentially many steps of which each is computable in 2ExpTime.
(Hardness): Reduction from MFA (MFA and DMFA coincide for rule sets without disjunctions)
[Cuenca Grau et al., 2013, Calì et al., 2010].

Complexity of BCQ entailment for DMFA Ruleset

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime*-complete.*

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime*-complete.*

Proof. (Sketch)
(Membership): We consider BCQ non-entailment.

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime-complete.

Proof. (Sketch)
(Membership): We consider BCQ non-entailment. The number of sets of fact sets is at most triply exponential.

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime*-complete.*

Proof. (Sketch)
(Membership): We consider BCQ non-entailment. The number of sets of fact sets is at most triply exponential. Since it suffices to guess one fact set in each chase step, we end up in N2ExpTime by using a similar step by step computation as for the DMFA check.

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime-complete.

Proof. (Sketch)
(Membership): We consider BCQ non-entailment. The number of sets of fact sets is at most triply exponential. Since it suffices to guess one fact set in each chase step, we end up in N2ExpTime by using a similar step by step computation as for the DMFA check. Thus, BCQ entailment is in coN2ExpTime.

Theorem

Let R be a rule set that is DMFA and let $\mathcal{K}$ be a knowledge base featuring R. BCQ entailment for $\mathcal{K}$ is coN2ExpTime-complete.

Proof. (Sketch)
(Membership): We consider BCQ non-entailment. The number of sets of fact sets is at most triply exponential. Since it suffices to guess one fact set in each chase step, we end up in N2ExpTime by using a similar step by step computation as for the DMFA check. Thus, BCQ entailment is in coN2ExpTime.
(Hardness): Reduction from the word problem of N2ExpTime-bounded turing machines similar to the proof for RMFA [Carral et al., 2017, Calì et al., 2010].

Results

- DMFA is a novel acyclicity notions tailored towards the disjunctive skolem chase.

Results

- DMFA is a novel acyclicity notions tailored towards the disjunctive skolem chase.
- DMFA is in between MFA and RMFA (as expected).

RESULTS

- DMFA is a novel acyclicity notions tailored towards the disjunctive skolem chase.
- DMFA is in between MFA and RMFA (as expected).
- Checking DMFA is 2EXPTIME-complete and checking BCQ entailment for a rule set that is DMFA is coN2EXPTIME-complete.

Future Work

In theory:

1. Develop a cyclicity notion for the disjunctive skolem chase.

In theory:

1. Develop a cyclicity notion for the disjunctive skolem chase.
2. Refine notions to capture as many rule sets as possible.

Future Work

In theory:

1. Develop a cyclicity notion for the disjunctive skolem chase.
2. Refine notions to capture as many rule sets as possible.

In practice:

1. Evaluate DMFA on real world knowledge bases.

Future Work

In theory:

1. Develop a cyclicity notion for the disjunctive skolem chase.
2. Refine notions to capture as many rule sets as possible.

In practice:

1. Evaluate DMFA on real world knowledge bases.
2. Use an ASP based implementation of the disjunctive skolem chase for reasoning with description logics.

References I

Abiteboul, S., Hull, R., and Vianu, V. (1995).
Foundations of Databases.
Addison-Wesley.

Adrian, W. T., Alviano, M., Calimeri, F., Cuteri, B., Dodaro, C., Faber, W., Fuscà, D., Leone, N., Manna, M., Perri, S., Ricca, F., Veltri, P., and Zangari, J. (2018).
The ASP system DLV: advancements and applications.
Künstliche Intell., 32(2-3):177–179.

Aho, A. V., Beeri, C., and Ullman, J. D. (1979).
The theory of joins in relational databases.
ACM Trans. Database Syst., 4(3):297–314.

Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., and Ricca, F. (2019).
Evaluation of disjunctive programs in WASP.
In [Balduccini et al., 2019], pages 241–255.

References II

Baget, J., Leclère, M., Mugnier, M., and Salvat, E. (2011).
On rules with existential variables: Walking the decidability line.
Artif. Intell., 175(9-10):1620–1654.

Balduccini, M., Lierler, Y., and Woltran, S., editors (2019).
***Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019*, volume 11481 of *Lecture Notes in Computer Science*. Springer.**

Beeri, C. and Vardi, M. Y. (1981).
The implication problem for data dependencies.
In [Even and Kariv, 1981], pages 73–85.

Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., and Tsamoura, E. (2017).
Benchmarking the chase.
In [Sallinger et al., 2017], pages 37–52.

References III

Bourhis, P., Leclère, M., Mugnier, M., Tison, S., Ulliana, F., and Gallois, L. (2019).
Oblivious and semi-oblivious boundedness for existential rules.
In [Kraus, 2019], pages 1581–1587.

Bourhis, P., Manna, M., Morak, M., and Pieris, A. (2016).
Guarded-based disjunctive tuple-generating dependencies.
ACM Trans. Database Syst., 41(4):27:1–27:45.

Bourhis, P., Morak, M., and Pieris, A. (2013).
The impact of disjunction on query answering under guarded-based existential rules.
In [Eiter et al., 2013], pages 539–551.

Calì, A., Gottlob, G., and Pieris, A. (2010).
Query answering under non-guarded rules in Datalog+/-.
In [Hitzler and Lukasiewicz, 2010], pages 1–17.

References IV

Calvanese, D., Lenzerini, M., and Motwani, R., editors (2002).
***Database Theory - ICDT 2003, 9th International Conference*, volume 2572 of *Lecture Notes in Computer Science*. Springer.**

Carral, D., Dragoste, I., and Krötzsch, M. (2017).
Restricted chase (non)termination for existential rules with disjunctions.
In [Sierra, 2017], pages 922–928.

Carro, M., King, A., Saeedloei, N., and Vos, M. D., editors (2016).
***Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs*, volume 52 of *OASICS*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.**

Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., and Wang, Z. (2013).
Acyclicity notions for existential rules and their application to query answering in ontologies.
J. Artif. Intell. Res., 47:741–808.

References V

Deutsch, A., Nash, A., and Remmel, J. B. (2008).
The chase revisited.
In [Lenzerini and Lembo, 2008], pages 149–158.

Deutsch, A. and Tannen, V. (2003).
Reformulation of XML queries and constraints.
In [Calvanese et al., 2002], pages 225–241.

Eiter, T., Glimm, B., Kazakov, Y., and Krötzsch, M., editors (2013).
***Informal Proceedings of the 26th International Workshop on Description Logics*, volume 1014 of *CEUR Workshop Proceedings*.**
CEUR-WS.org.

Esparza, J., Fraigniaud, P., Husfeldt, T., and Koutsoupias, E., editors (2014).
***Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, volume 8573 of *Lecture Notes in Computer Science*. Springer.**

References VI

- **Even, S. and Kariv, O., editors (1981).**
  ***Automata, Languages and Programming, 8th Colloquium*, volume 115 of *Lecture Notes in Computer Science*. Springer.**

- **Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005).**
  Data exchange: semantics and query answering.
  Theor. Comput. Sci., 336(1):89–124.

- Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Wanko, P. (2016).
  Theory solving made easy with clingo 5.
  In [Carro et al., 2016], pages 2:1–2:15.

- Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2012a).
  ***Answer Set Solving in Practice*.**
  Synthesis Lectures on Artificial Intelligence and Machine Learning.
  Morgan & Claypool Publishers.

References VII

Gebser, M., Kaufmann, B., and Schaub, T. (2012b).
Multi-threaded ASP solving with clasp.
Theory Pract. Log. Program., 12(4-5):525–545.

Gogacz, T. and Marcinkowski, J. (2014).
All-instances termination of chase is undecidable.
In [Esparza et al., 2014], pages 293–304.

Grahne, G. and Onet, A. (2018).
Anatomy of the chase.
Fundam. Inform., 157(3):221–270.

Hitzler, P. and Lukasiewicz, T., editors (2010).
***Web Reasoning and Rule Systems - Fourth International Conference, RR 2010*, volume 6333 of *Lecture Notes in Computer Science*. Springer.**

Kraus, S., editor (2019).
***Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*. ijcai.org.**

Lenzerini, M. and Lembo, D., editors (2008).
***Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008*. ACM.**

Maier, D., Mendelzon, A. O., and Sagiv, Y. (1979).
Testing implications of data dependencies.
ACM Trans. Database Syst., 4(4):455–469.

Marnette, B. (2009).
Generalized schema-mappings: from termination to tractability.
In [Paredaens and Su, 2009], pages 13–22.

Paredaens, J. and Su, J., editors (2009).
***Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009*. ACM.**

Rudolph, S., Krötzsch, M., and Hitzler, P. (2012).
Type-elimination-based reasoning for the description logic SHIQbs using decision diagrams and disjunctive datalog.
Logical Methods in Computer Science, 8(1).

References IX

Sallinger, E., den Bussche, J. V., and Geerts, F., editors (2017).
Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017. ACM.

Sierra, C., editor (2017).
Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017. ijcai.org.

Technical References

The sources of the presentation can be found on Gitlab:
https://gitlab.com/m0nstR/defence-grosser-beleg

The presentation uses the Focus Theme for Latex Beamer
https://github.com/elauksap/focus-beamertheme
which was obtained from Latex Templates
http://www.latextemplates.com/template/focus-presentation