Reasoning over Existential Rules with Acyclicity Notions and the Datalog-first Restricted Chase

David Carral



Slides available at https://iccl.inf.tu-dresden.de/web/Existential-rules-acyclicity

Preliminaries

 $\forall x, y, z. \left(\mathsf{HasParent}(x, y) \land \mathsf{HasSister}(y, z) \to \mathsf{HasAunt}(x, z) \right)$ $\forall x. \left(\mathsf{Human}(x) \to \exists y. \left(\mathsf{HasParent}(x, y) \land \mathsf{Human}(y) \right) \right)$ $\forall x, y, w. \left(\mathsf{P}(x, a, y) \land \mathsf{R}(y, w) \land \mathsf{S}(w, x) \to \exists v. \left(\mathsf{R}(w, v) \land \mathsf{A}(v) \right) \right)$

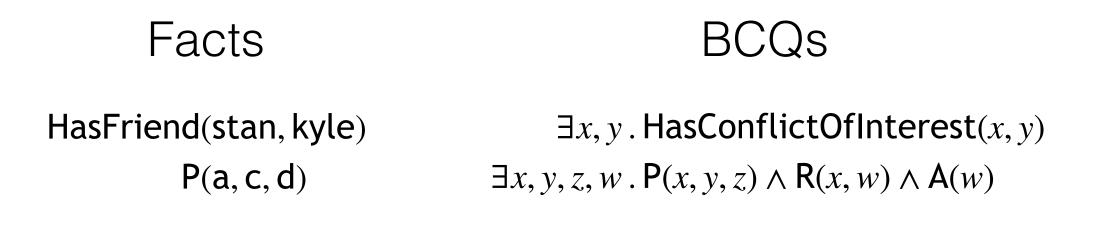
 $\begin{aligned} \mathsf{HasParent}(x,y) \wedge \mathsf{HasSister}(y,z) &\to \mathsf{HasAunt}(x,z) \\ \mathsf{Human}(x) \to \exists y \, . \, \mathsf{HasParent}(x,y) \wedge \mathsf{Human}(y) \\ \mathsf{P}(x,\mathsf{a},y) \wedge \mathsf{R}(y,w) \wedge \mathsf{S}(w,x) \to \exists v \, . \, \mathsf{R}(w,v) \wedge \mathsf{A}(v) \end{aligned}$

 $\begin{aligned} \mathsf{HasParent}(x,y) \wedge \mathsf{HasSister}(y,z) &\to \mathsf{HasAunt}(x,z) \\ \mathsf{Human}(x) \to \exists y \, . \, \mathsf{HasParent}(x,y) \wedge \mathsf{Human}(y) \\ \mathsf{P}(x,\mathsf{a},y) \wedge \mathsf{R}(y,w) \wedge \mathsf{S}(w,x) \to \exists v \, . \, \mathsf{R}(w,v) \wedge \mathsf{A}(v) \end{aligned}$

Facts

HasFriend(stan, kyle) P(a, c, d)

 $\begin{aligned} \mathsf{HasParent}(x,y) \wedge \mathsf{HasSister}(y,z) &\to \mathsf{HasAunt}(x,z) \\ \mathsf{Human}(x) \to \exists y \, . \, \mathsf{HasParent}(x,y) \wedge \mathsf{Human}(y) \\ \mathsf{P}(x,\mathsf{a},y) \wedge \mathsf{R}(y,w) \wedge \mathsf{S}(w,x) \to \exists v \, . \, \mathsf{R}(w,v) \wedge \mathsf{A}(v) \end{aligned}$



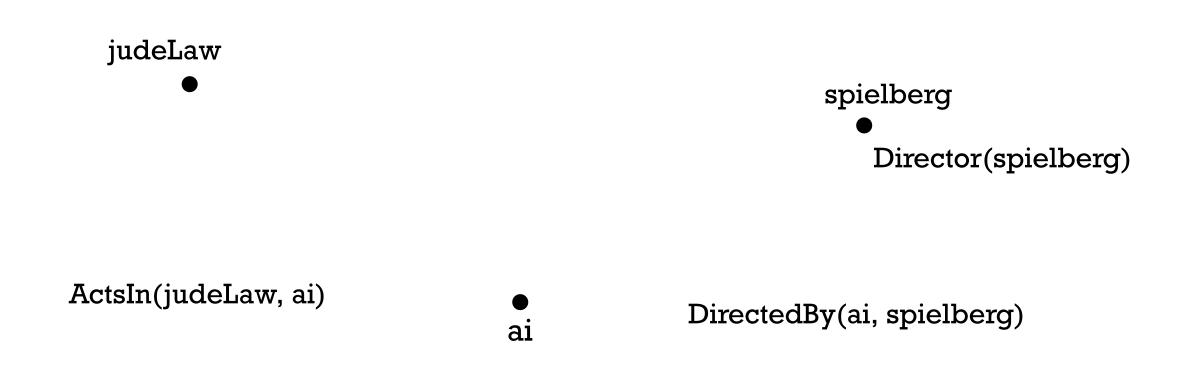
Features(x, y) \rightarrow Actor(y)DirectedBy(x, y) \rightarrow Directs(y, x)ActsIn(x, y) \rightarrow Features(y, x)Directs(x, y) \wedge Features(y, z) \rightarrow DirectsActor(x, z)

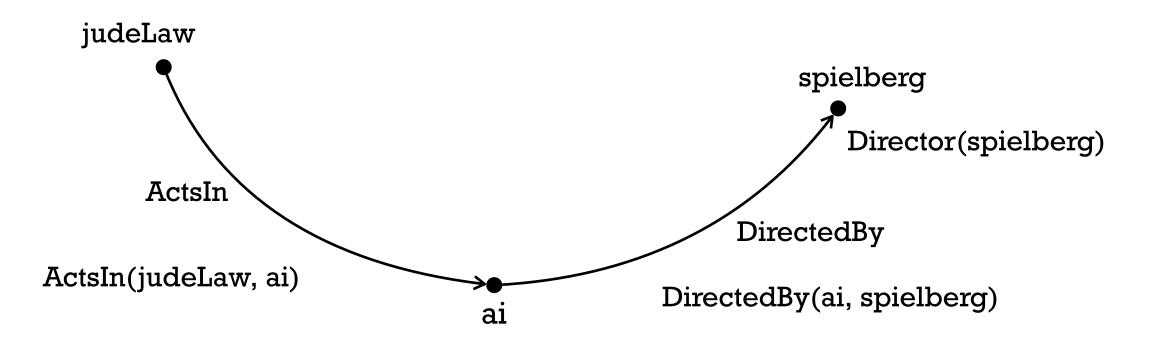
Director(spielberg)

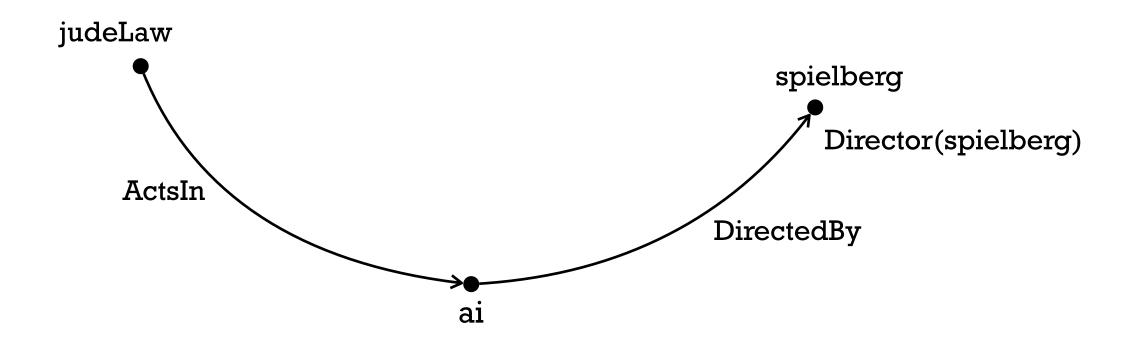
ActsIn(judeLaw, ai)

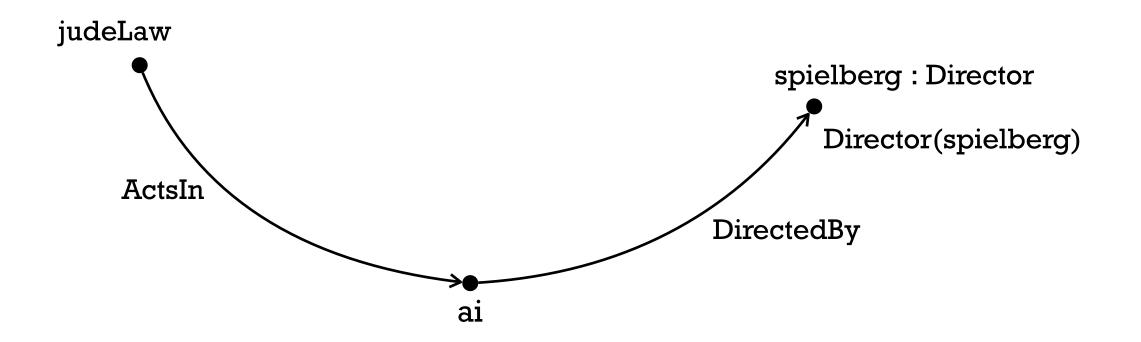
DirectedBy(ai, spielberg)

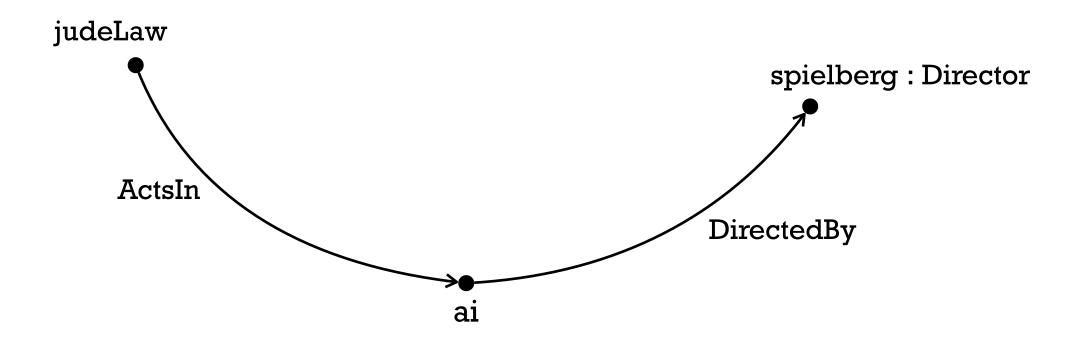
Features(x, y) \rightarrow Actor(y)DirectedBy(x, y) \rightarrow Directs(y, x)ActsIn(x, y) \rightarrow Features(y, x)Directs(x, y) \wedge Features(y, z) \rightarrow DirectsActor(x, z)

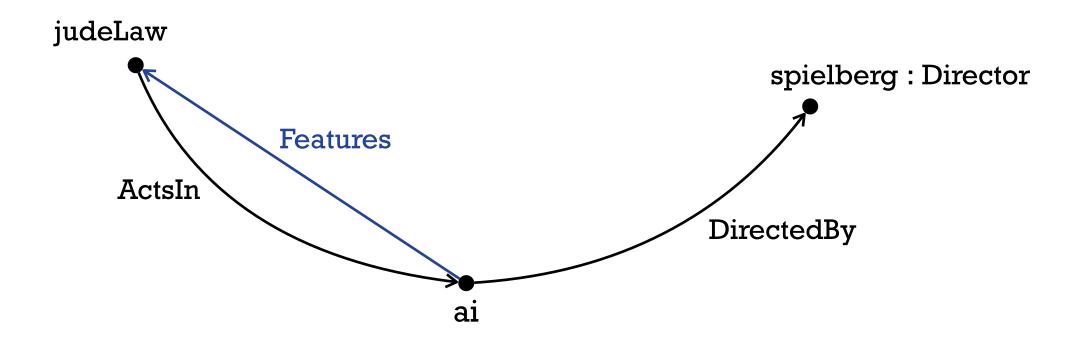


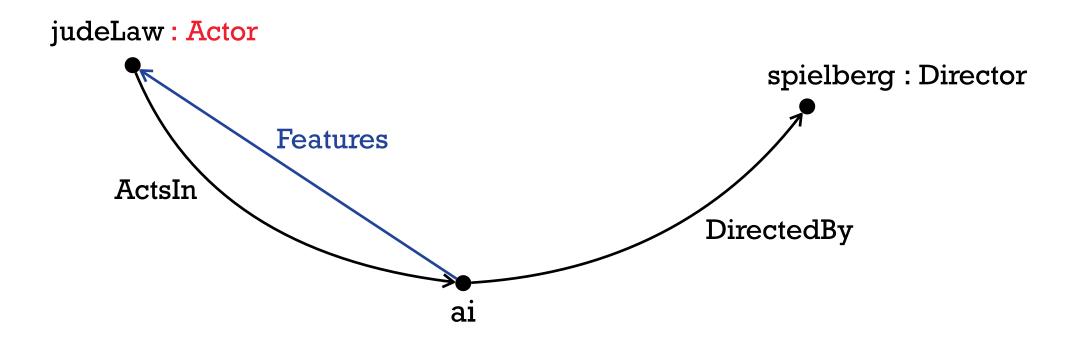


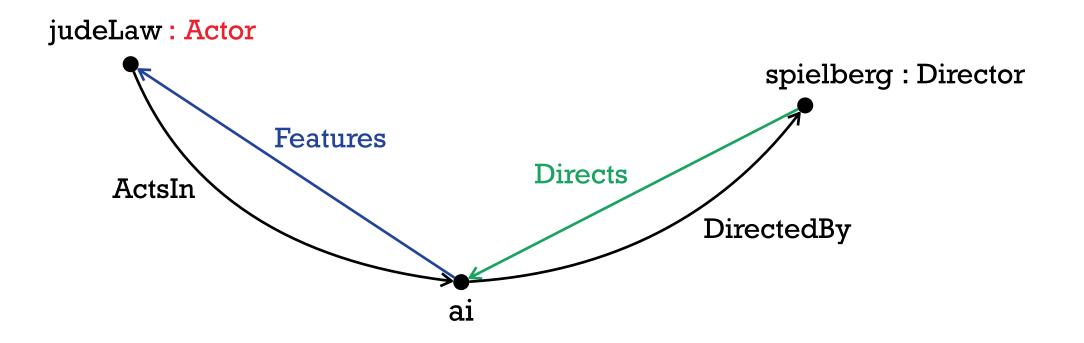


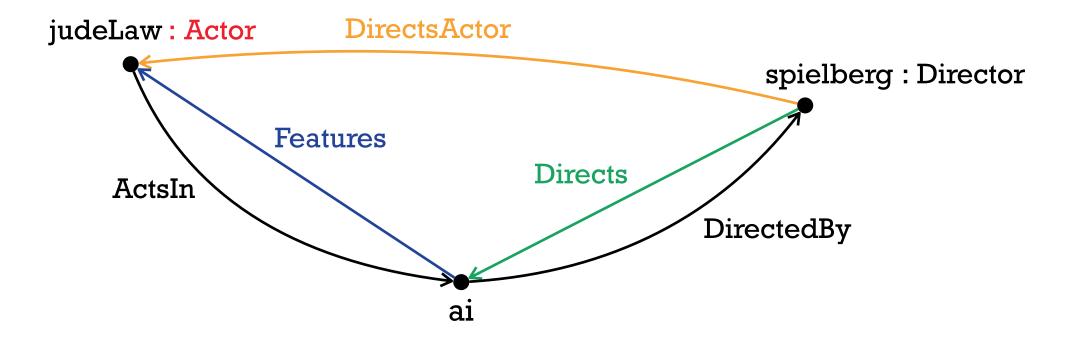










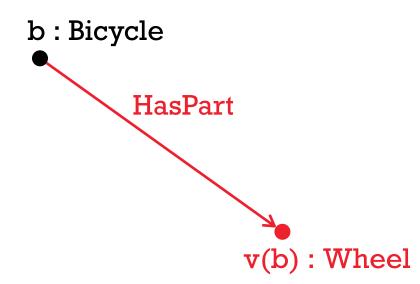


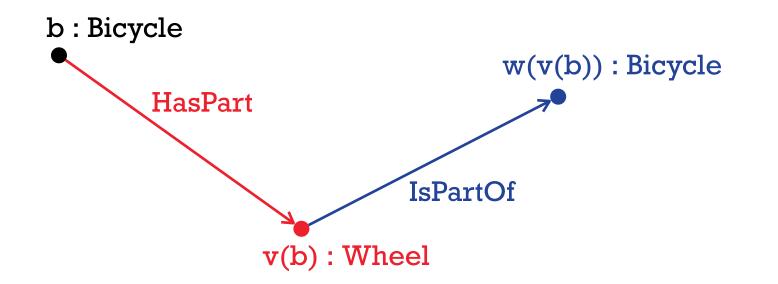
b:Bicycle

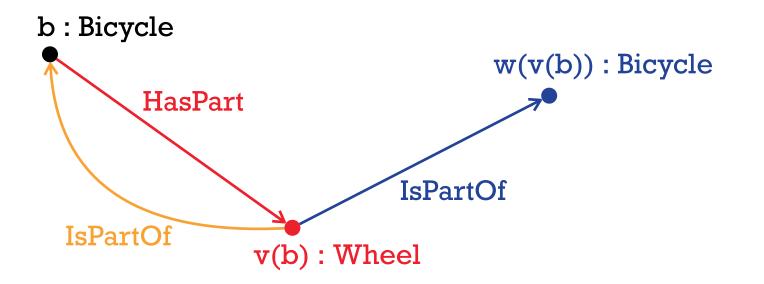
$$\begin{split} &\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, f_v(x)) \land \text{Wheel}(f_v(x)) & \text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x) \\ &\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, f_w(x)) \land \text{Bicycle}(f_w(x)) & \text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x) \end{split}$$

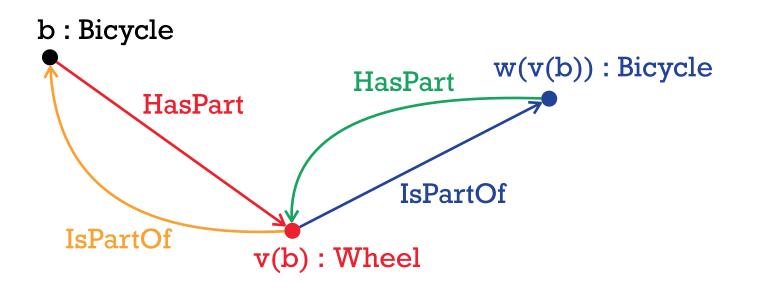
b: Bicycle

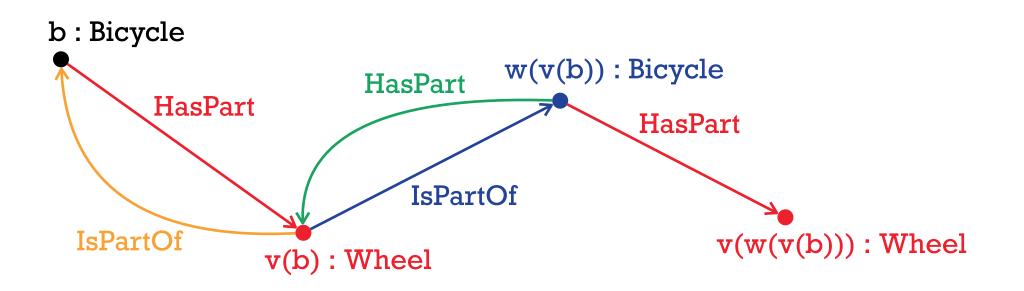
b: Bicycle

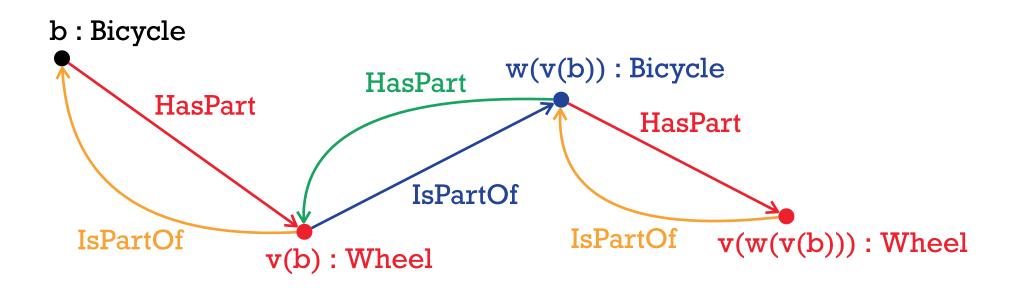


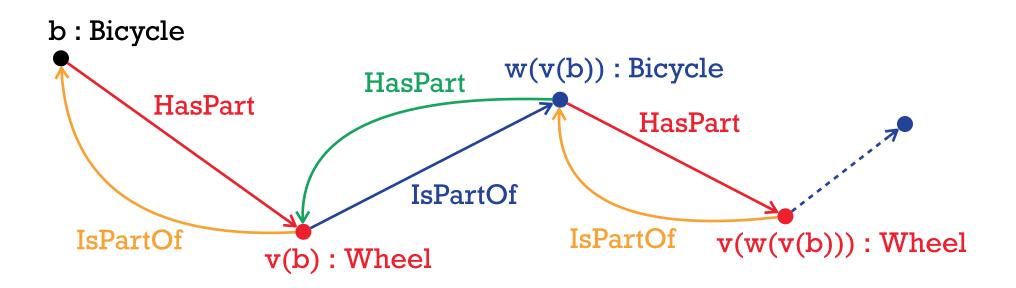


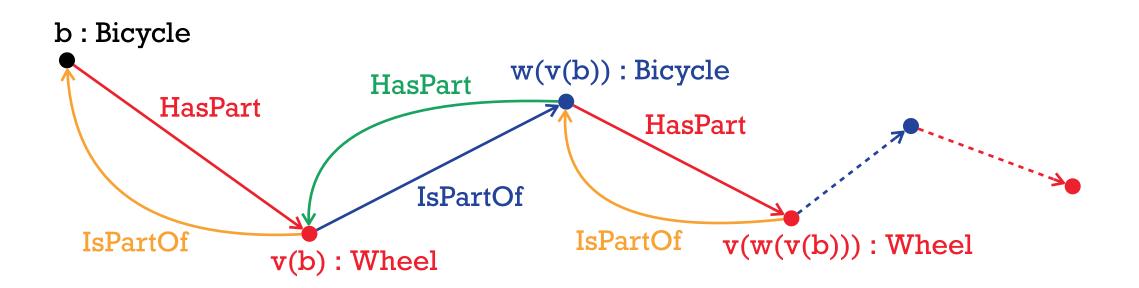


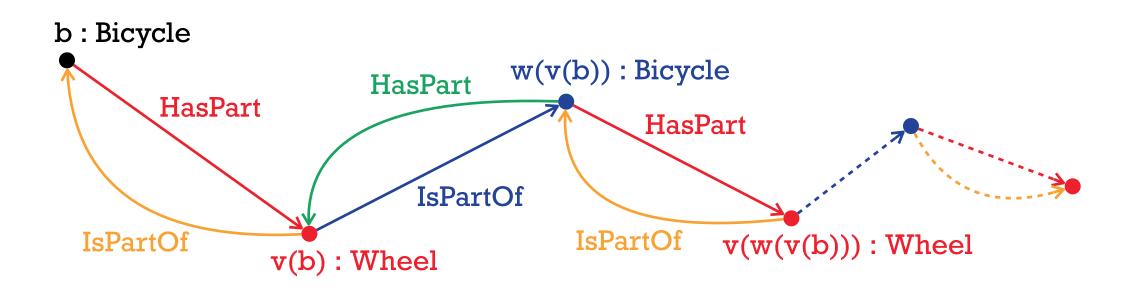


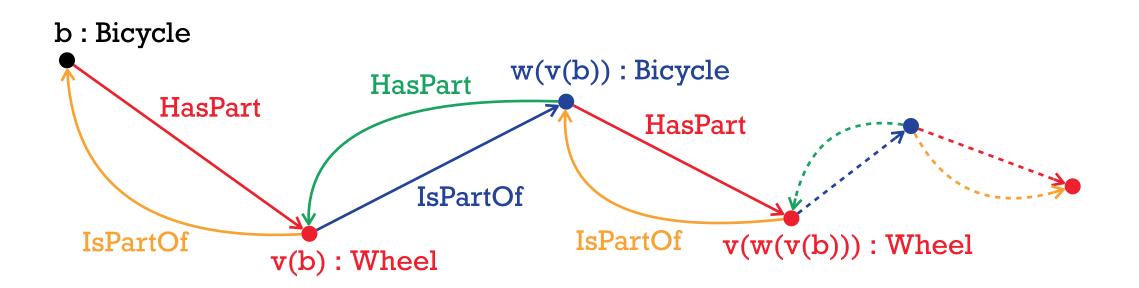












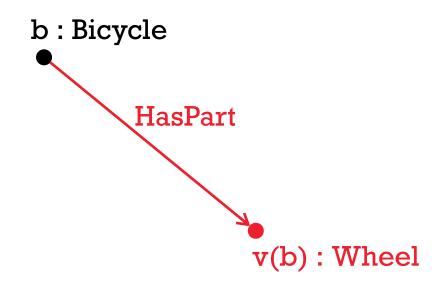
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

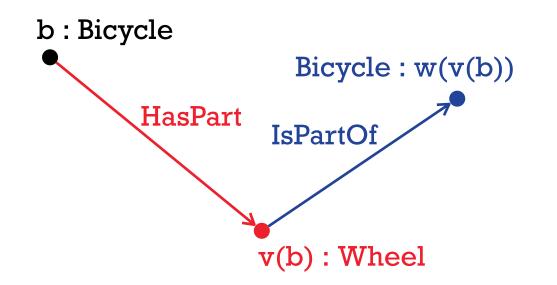
 $HasPart(x, y) \longrightarrow IsPartOf(y, x)$ $IsPartOf(x, y) \longrightarrow HasPart(y, x)$

b:Bicycle

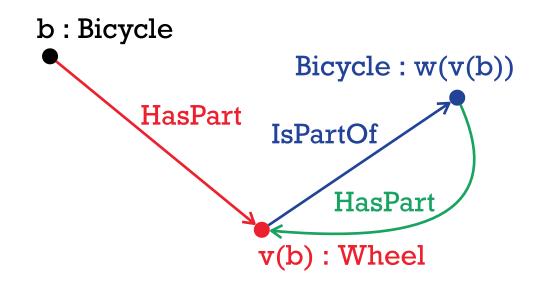
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$



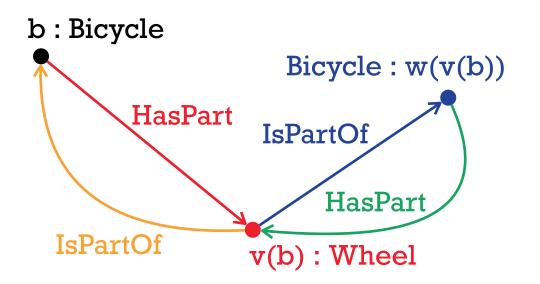
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

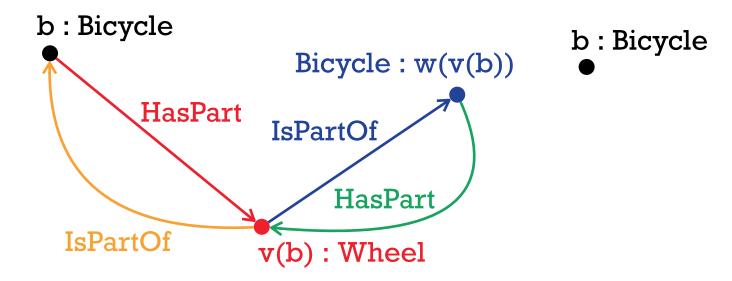


 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

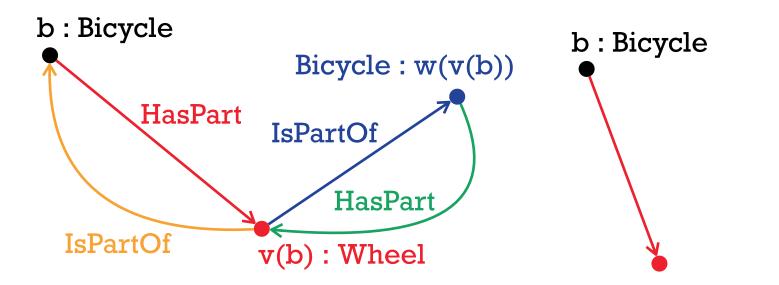


 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

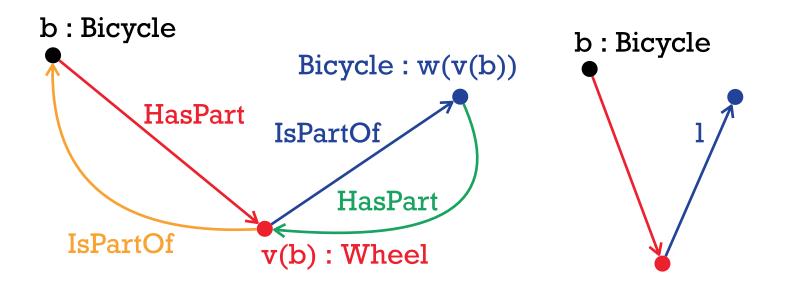




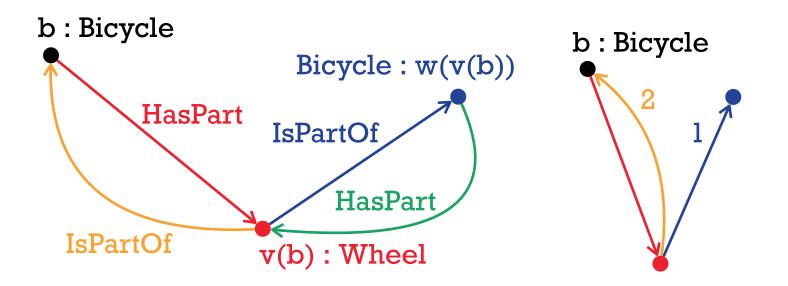
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

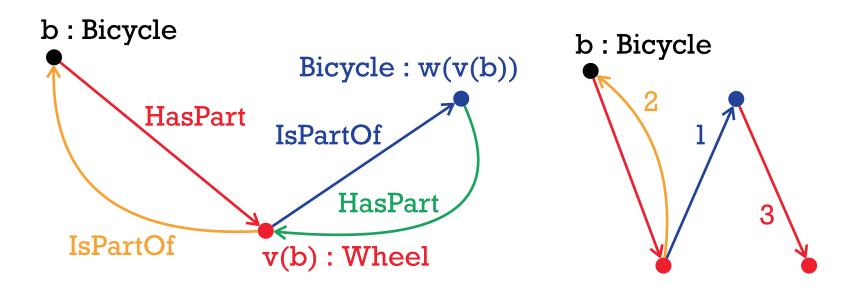


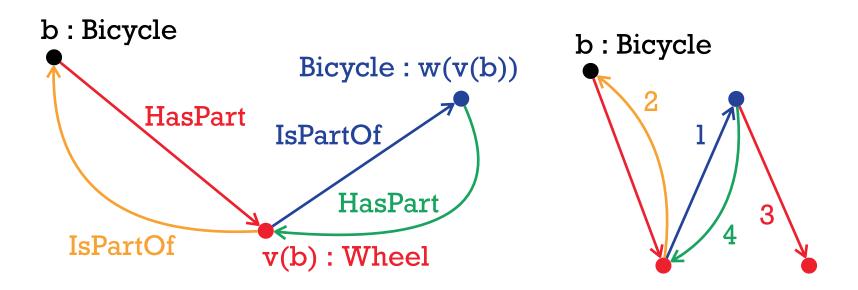
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

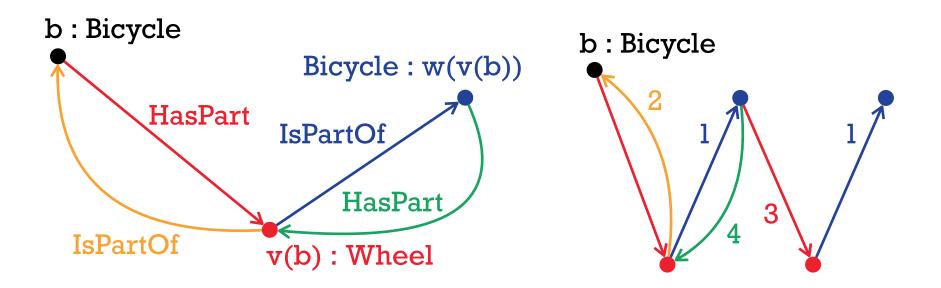


 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

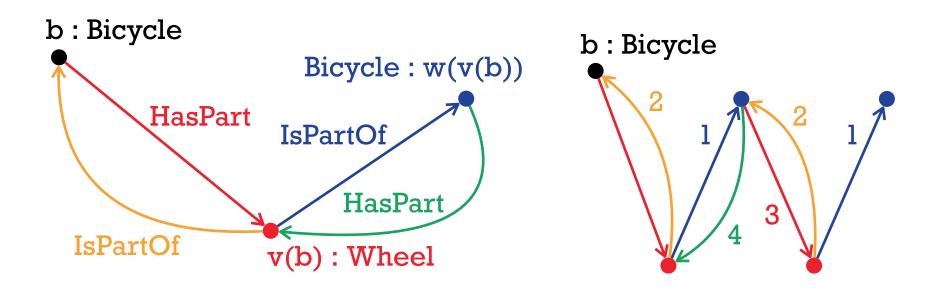


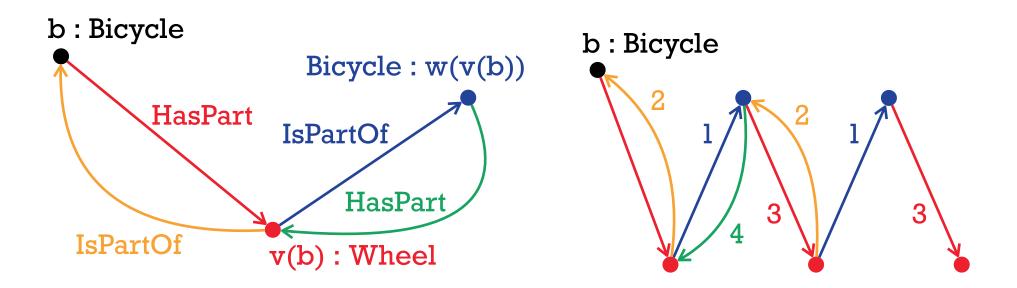




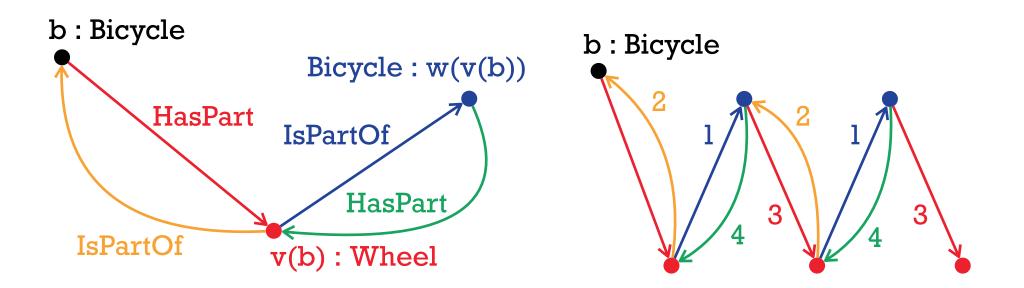


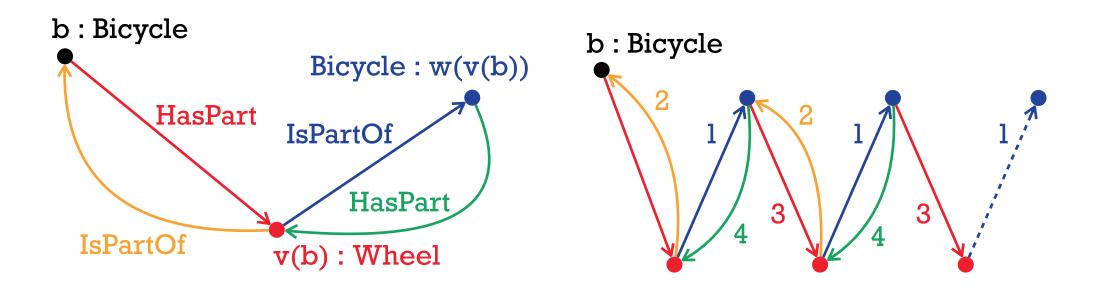
Bicycle(x) $\rightarrow \exists v . HasPart(x, v) \land Wheel(v)$ HasPart(x, y)Wheel(x) $\rightarrow \exists w . IsPartOf(x, w) \land Bicycle(w)$ IsPartOf(x, y)

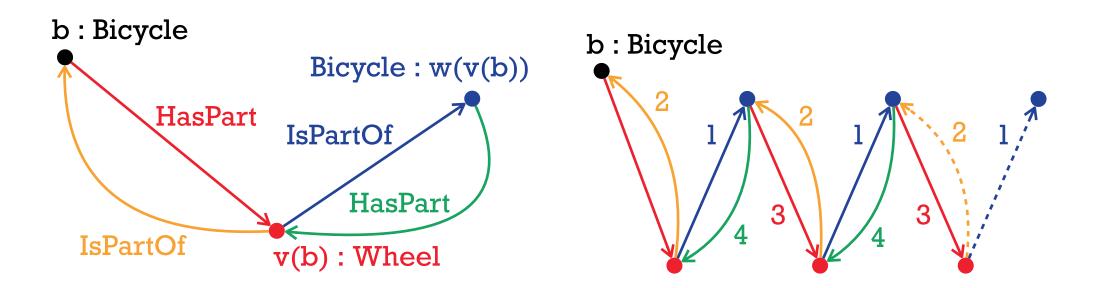




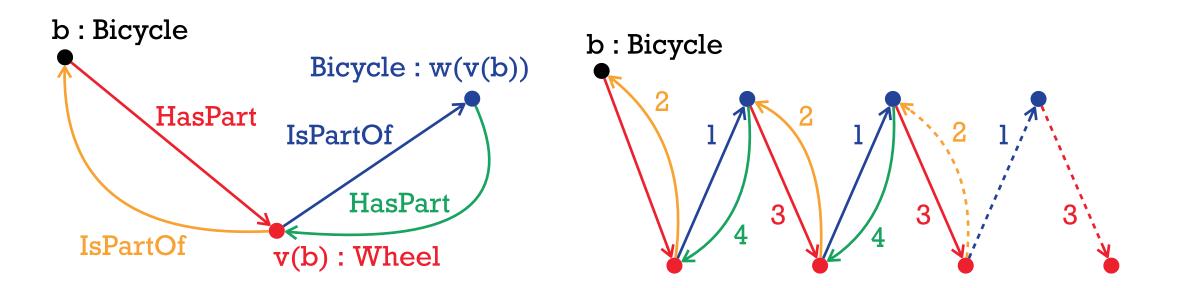
David Carral

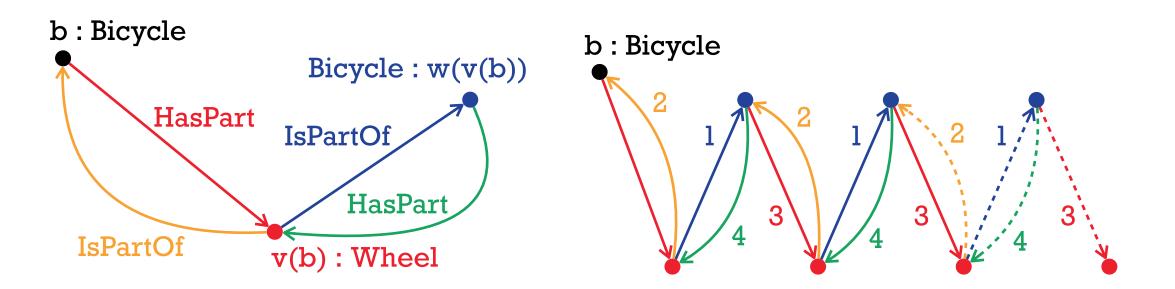






David Carral





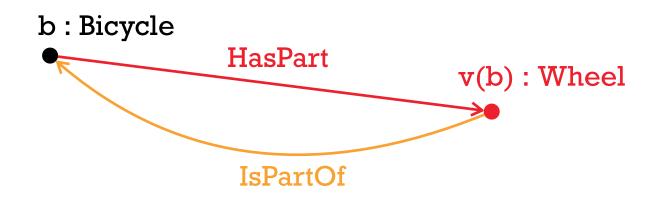
The Datalog-First Restricted Chase

b : Bicycle

The Datalog-First Restricted Chase



The Datalog-First Restricted Chase



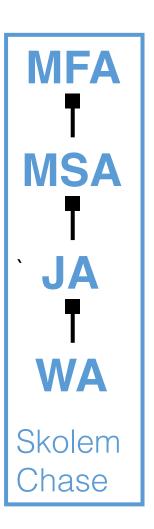
Acyclicity Notions

Acyclicity Notions

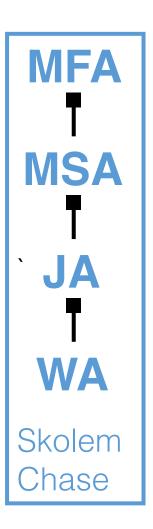
Restricted Chase (Non)Termination for Existential Rules with Disjunctions David Carral, Irina Dragoste, and Markus Krötzsch [IJCAI 2017]

- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (JA) [IJCAI 2011]
- Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]

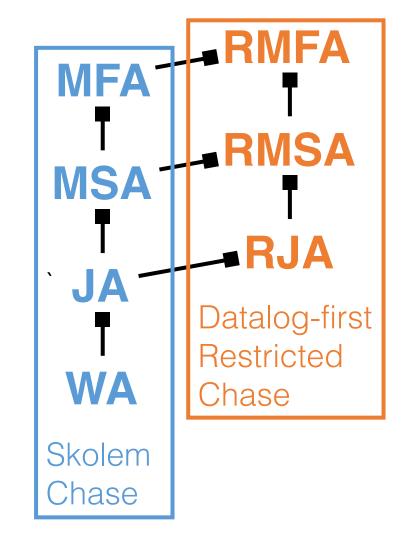
- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (JA) [IJCAI 2011]
- Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]



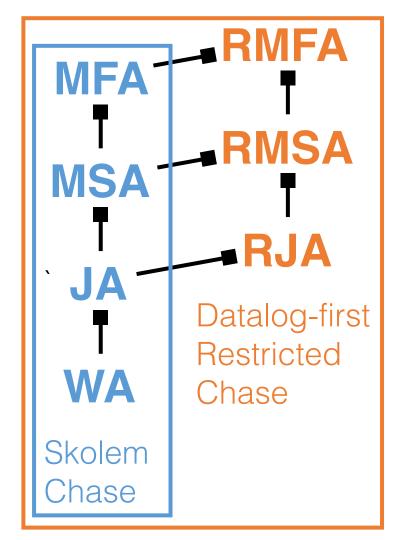
- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (**JA**) [IJCAI 2011]
- Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]
- * Restricted Joint Acyclicity (RJA), Restricted Model-Summarising Acyclicity (RMSA), and Restricted Model-Faithful Acyclicity (RMFA) [IJCAI 2017]



- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (JA) [IJCAI 2011]
- Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]
- * Restricted Joint Acyclicity (RJA), Restricted Model-Summarising Acyclicity (RMSA), and Restricted Model-Faithful Acyclicity (RMFA) [IJCAI 2017]

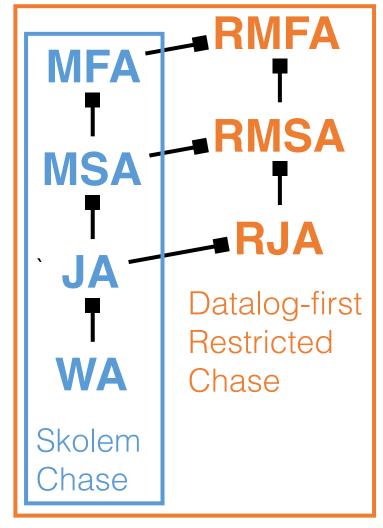


- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (JA) [IJCAI 2011]
- Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]
- * Restricted Joint Acyclicity (RJA), Restricted Model-Summarising Acyclicity (RMSA), and Restricted Model-Faithful Acyclicity (RMFA) [IJCAI 2017]



- * Weak Acyclicity (WA) [Theor. Comput. Sci. 2005]
- * Joint Acyclicity (JA) [IJCAI 2011]
- * Model-Summarising Acyclicity (MSA) and Model-Faithful Acyclicity (MFA) [J. Artif. Intell. Res. 2013]
- * Restricted Joint Acyclicity (RJA), Restricted Model-Summarising Acyclicity (RMSA), and Restricted Model-Faithful Acyclicity (RMFA) [IJCAI 2017]

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists v \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists w \ . \ IsPartOf(x, w) \land Bicycle(w) \\ HasPart(x, y) \longrightarrow IsPartOf(y, x) \\ IsPartOf(x, y) \longrightarrow HasPart(y, x) \end{array}$



* **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.

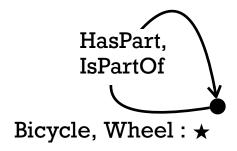
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

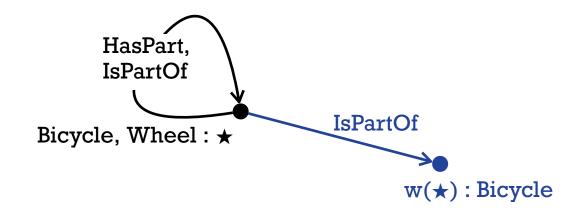
 $\begin{array}{ll} Bicycle(x) \longrightarrow \exists y \; . \; HasPart(x, \, v) & H \\ Wheel(x) \longrightarrow \exists y \; . \; IsPartOf(x, \, w) \; \land \; Bicycle(w) & Is \end{array}$

- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

Bicycle(x) $\rightarrow \exists y . HasPart(x, v)$ IWheel(x) $\rightarrow \exists y . IsPartOf(x, w) \land Bicycle(w)$ I

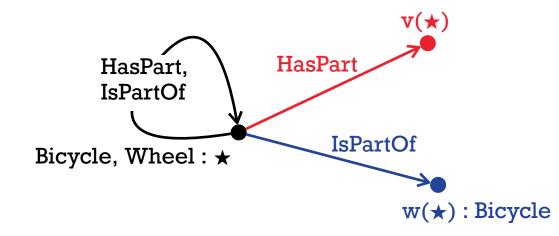


- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.



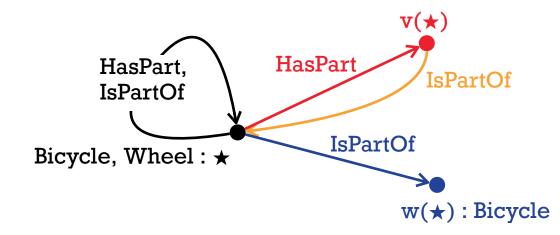
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

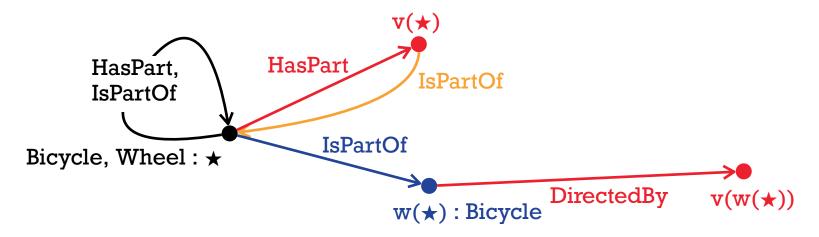


- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

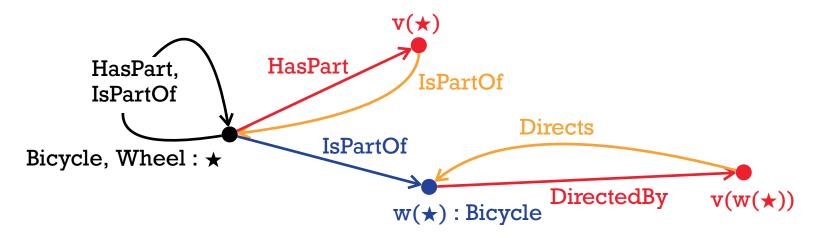
Bicycle(x) $\longrightarrow \exists y . HasPart(x, v)$ Wheel(x) $\longrightarrow \exists y . IsPartOf(x, w) \land Bicycle(w)$



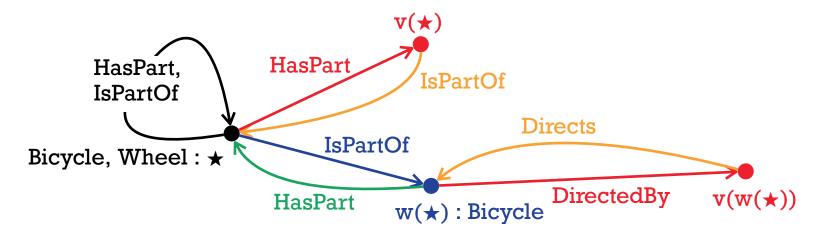
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.



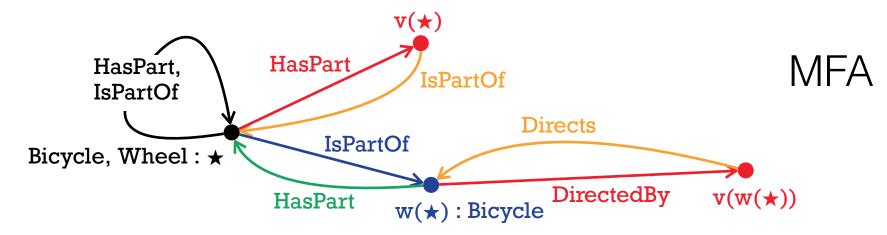
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.



- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.



- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

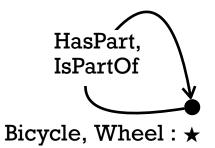


- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \;.\; HasPart(x,\,v) \;\land\; Wheel(v) \\ Wheel(x) \longrightarrow \exists y \;.\; IsPartOf(x,\,w) \;\land\; Bicycle(w) \end{array}$

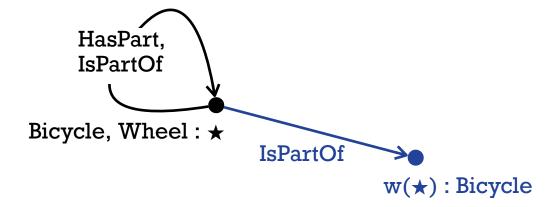
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \;.\; HasPart(x,\,v) \;\land\; Wheel(v) \\ Wheel(x) \longrightarrow \exists y \;.\; IsPartOf(x,\,w) \;\land\; Bicycle(w) \end{array}$



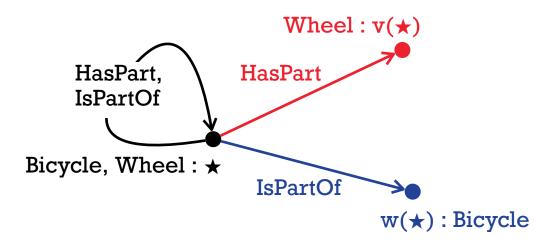
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



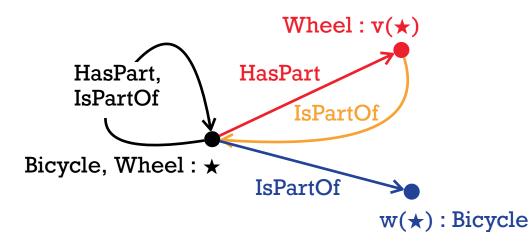
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



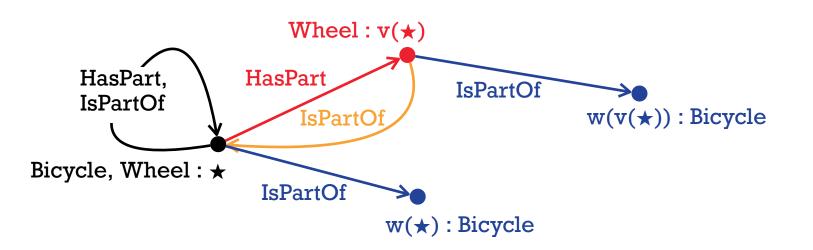
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



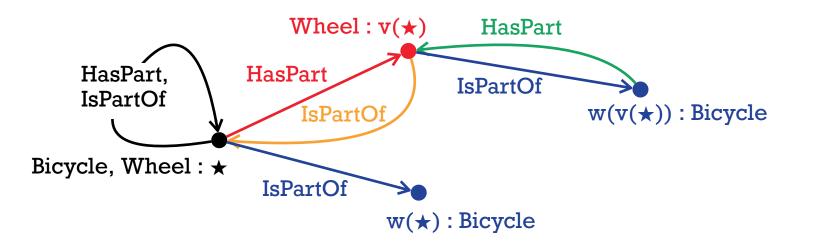
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



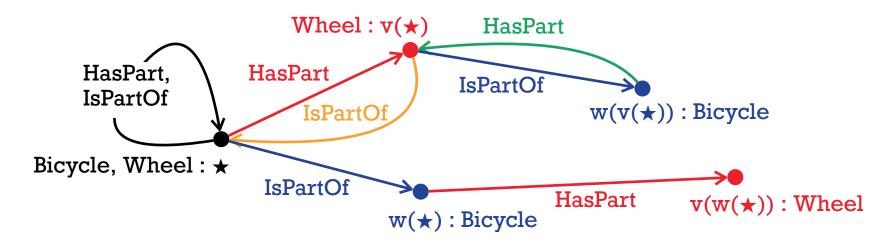
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



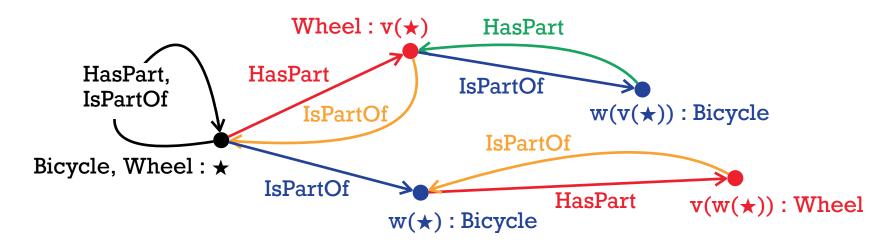
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$



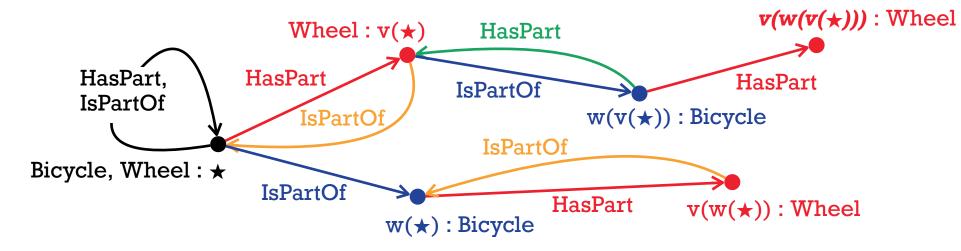
- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \;.\; HasPart(x,\,v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists y \;.\; IsPartOf(x,\,w) \land Bicycle(w) \end{array}$



- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



- * **Fact:** If the Skolem chase terminates on the critical instance (the set of all possible facts containing a single constant "★"), then it terminates on all sets of facts.
- * **MFA Check:** Perform chase with the critical instance, check if it stops; give up if a **cyclic skolem term** (with a repeated function symbol) appears.

 $Bicycle(x) \longrightarrow \exists y . HasPart(x, v) \land Wheel(v)$ $HasPart(x, y) \longrightarrow IsPartOf(y, x)$ Wheel(x) $\rightarrow \exists y \text{ . IsPartOf}(x, w) \land Bicycle(w)$ IsPartOf(x, y) \longrightarrow HasPart(y, x) **v(w(v(★)))** : Wheel Wheel: $v(\star)$ HasPart HasPart. **HasPart HasPart IsPartOf** IsPartOf **IsPartO** $w(v(\star))$: Bicycle Not MFA! **IsPartOf** Bicycle, Wheel : \star **IsPartOf** HasPart $v(w(\star))$: Wheel $w(\bigstar)$: Bicycle

* Problem: Datalog-first restricted chase termination is not monotone!

* Problem: Datalog-first restricted chase termination is not monotone!

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

* Problem: Datalog-first restricted chase termination is not monotone!

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

HasPart(x, y) \longrightarrow IsPartOf(y, x) IsPartOf(x, y) \longrightarrow HasPart(y, x)

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Bicycle(x) \rightarrow \exists y . HasPart(x, v) \land Wheel(v)HWheel(x) \rightarrow \exists y . IsPartOf(x, w) \land Bicycle(w)Is

 $HasPart(x, y) \longrightarrow IsPartOf(y, x)$ $IsPartOf(x, y) \longrightarrow HasPart(y, x)$

v(*w*(*t*)) : *Wheel*

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

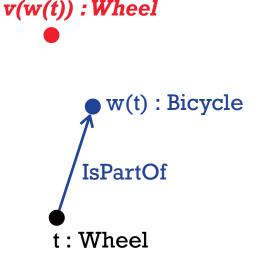
* Such a fact may only be derived via application of the red rule on Bicycle(w(t)) which in turn may only be derived if the blue rule is applied. Hence, Wheel(t) and IsPartOf(t, w(t)) and are also part of the chase before Wheel(v(w(t))) is derived.

v(*w*(*t*)) : *Wheel*

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

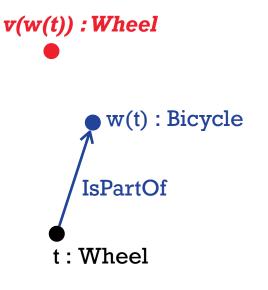
Such a fact may only be derived via application of the **red rule** on **Bicycle(w(t))** which in turn may only be derived if the **blue rule** is applied. Hence, **Wheel(t)** and **IsPartOf(t, w(t))** and are also part of the chase before **Wheel(v(w(t)))** is derived.



- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

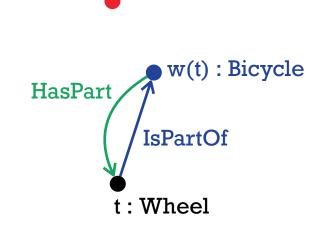
- Such a fact may only be derived via application of the **red rule** on **Bicycle(w(t))** which in turn may only be derived if the **blue rule** is applied. Hence, **Wheel(t)** and **IsPartOf(t, w(t))** and are also part of the chase before **Wheel(v(w(t)))** is derived.
- * Because the **green rule** is Datalog, **DirectedBy(v(t), t)** is also part of the chase.



- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

- Such a fact may only be derived via application of the **red rule** on **Bicycle(w(t))** which in turn may only be derived if the **blue rule** is applied. Hence, **Wheel(t)** and **IsPartOf(t, w(t))** and are also part of the chase before **Wheel(v(w(t)))** is derived.
- * Because the **green rule** is Datalog, **DirectedBy(v(t), t)** is also part of the chase.

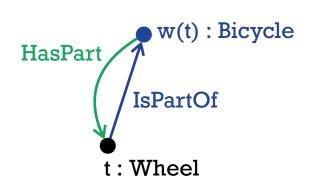


v(*w*(*t*)) : *Wheel*

- * Problem: Datalog-first restricted chase termination is not monotone!
- * In particular, it always terminates on the critical instance.
- * Idea: for each fact that occurs in the chase sequence, we can re-trace a necessary fact set the must have been derived to derive this fact. By checking these facts we can in some cases determine that the application of the rule and substitution that generates this fact is **blocked**.

Example: Suppose for a contradiction that the fact Wheel(v(w(t))) with t some term is derived during the computation of a chase sequence.

- Such a fact may only be derived via application of the **red rule** on **Bicycle(w(t))** which in turn may only be derived if the **blue rule** is applied. Hence, **Wheel(t)** and **IsPartOf(t, w(t))** and are also part of the chase before **Wheel(v(w(t)))** is derived.
- * Because the green rule is Datalog, DirectedBy(v(t), t) is also part of the chase.
- * The red rule may not be applied to introduce Director(v(w(t))) since its application with respect to the substitution {x / w(t)} is restricted.



v(*w*(*t*)) : *Wheel*

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

* Perform a chase like construction on the critical instance.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, v) \land Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, w) \land Bicycle(w) \end{array}$

- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.

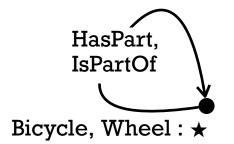
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

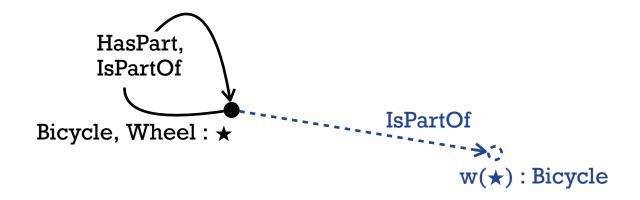
```
\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}
```



- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

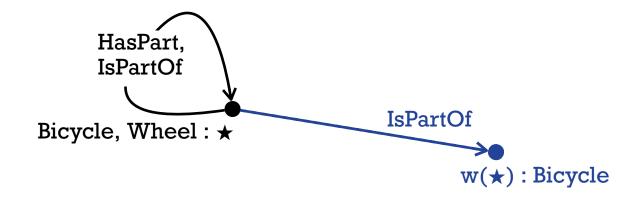
```
\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}
```

```
HasPart(x, y) \longrightarrow IsPartOf(y, x)IsPartOf(x, y) \longrightarrow HasPart(y, x)
```



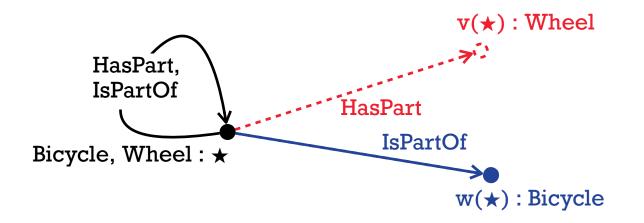
- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

```
\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}
```



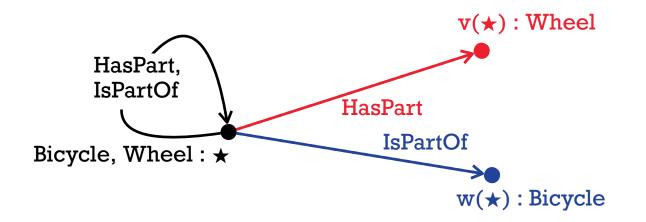
- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



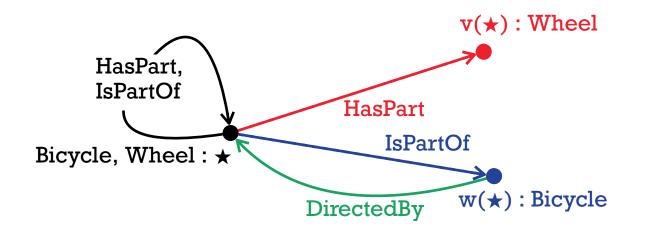
- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

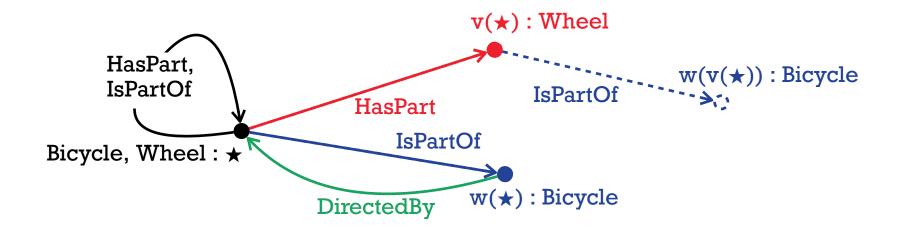


- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

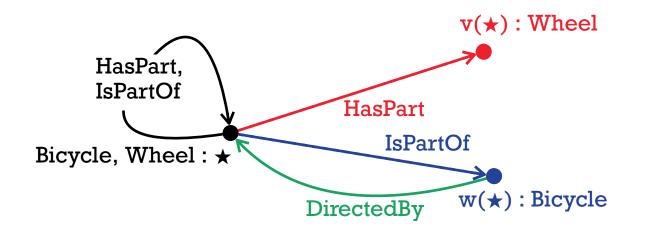


- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

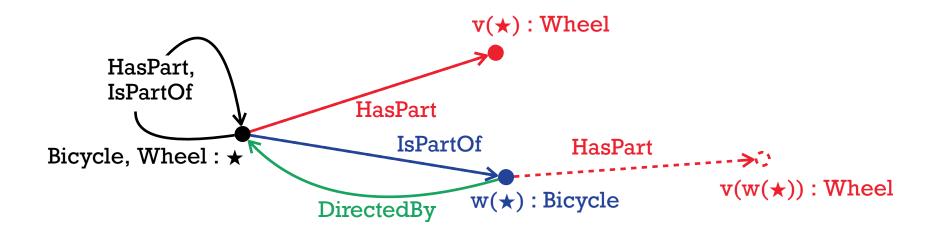


- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$

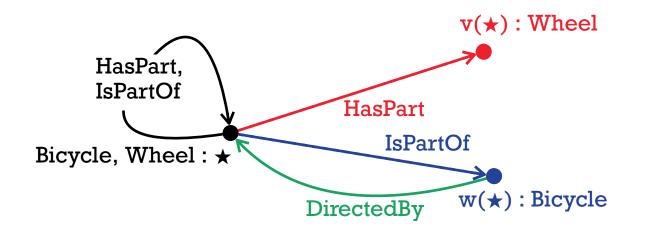


- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.



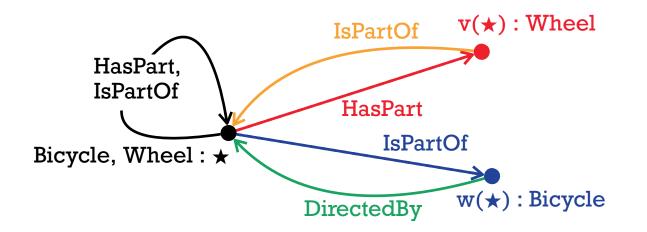
- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



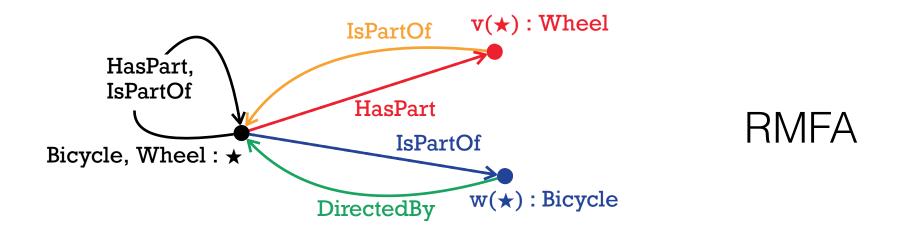
- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



- * Perform a chase like construction on the critical instance.
- * Only apply an existential rule with respect to a substitution if this pair is not **blocked**.
- * Give up if the procedure does not stop before the occurrence of a cyclic term.

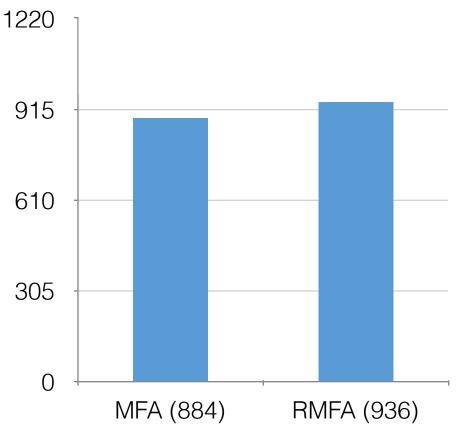
 $\begin{array}{l} Bicycle(x) \longrightarrow \exists y \ . \ HasPart(x, \, v) \ \land \ Wheel(v) \\ Wheel(x) \longrightarrow \exists y \ . \ IsPartOf(x, \, w) \ \land \ Bicycle(w) \end{array}$



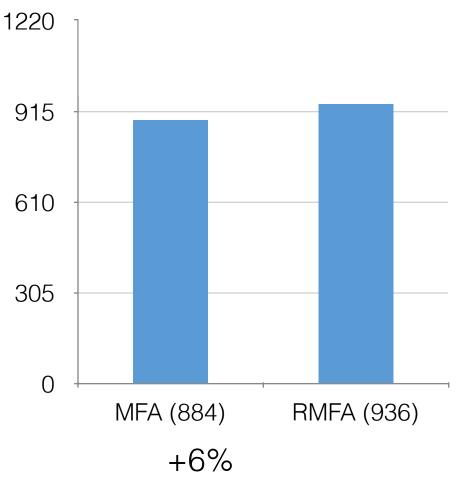
* We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$

- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.

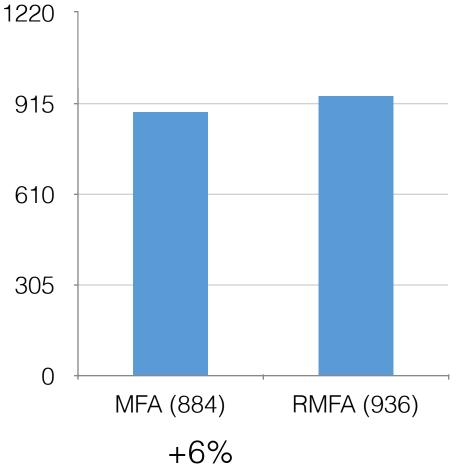
- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.



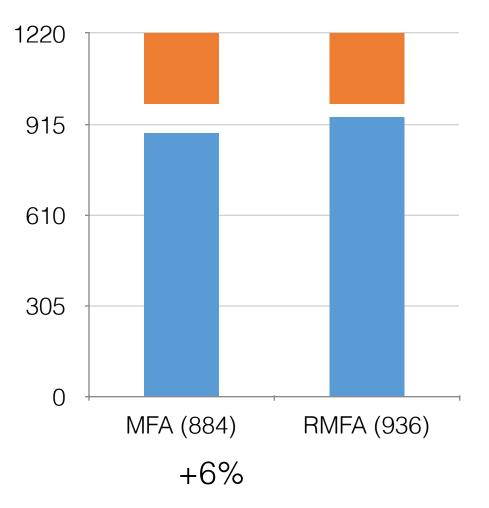
- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.



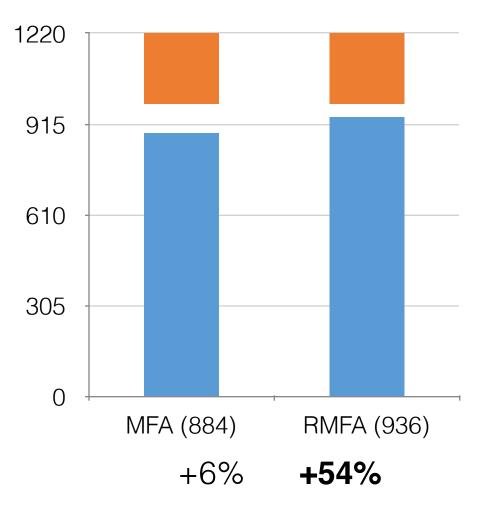
- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.
- * We developed a cyclicity notion, i.e., sufficient condition for chase nontermination: Restricted Model-Faithful Cyclicity (RMFC)



- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.
- * We developed a cyclicity notion, i.e., sufficient condition for chase nontermination: Restricted Model-Faithful Cyclicity (RMFC)



- * We selected all rule sets from the MOWLCorp with less than 1000 rules of the form $A(x) \rightarrow \exists y . R(x, y) \land B(y)$
- * We also considered (all) the rule sets in the Oxford Ontology Library.
- * We developed a cyclicity notion, i.e., sufficient condition for chase nontermination: Restricted Model-Faithful Cyclicity (RMFC)



Membership Checks

| | RJA | RMSA | RMFA |
|-------------------------------|---------|---------|-----------|
| No restrictions | ExpTime | ExpTime | 2-ExpTime |
| Variables per rule bounded | Р | Р | 2-ExpTime |

Membership Checks

| | RJA | RMSA | RMFA |
|-------------------------------|---------|---------|-----------|
| No restrictions | ExpTime | ExpTime | 2-ExpTime |
| Variables per rule bounded | Р | Р | 2-ExpTime |

BCQ entailment: 2-ExpTime

Ensuring Tractability of the Chase Ensuring Tractability of the Chase

Tractable Query Answering for Expressive Ontologies and Existential Rules <u>David Carral</u>, Irina Dragoste, and Markus Krötzsch [ISWC 2017]

$$A(x) \rightarrow \exists y . S(x, y) \land B(y)$$
$$B(x) \rightarrow \exists z . R(x, z) \land D(z)$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow \exists w . R(x, w)$$
$$B(x) \land C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$

W

y

Ζ

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

$$B(x) \wedge C(x) \rightarrow E(x)$$

$$S(x, y) \rightarrow C(x)$$

W

y

Ζ

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

$$B(x) \wedge C(x) \rightarrow E(x)$$

$$S(x, y) \rightarrow C(x) \qquad A(c)$$

$$S(c, y(c)), B(y(c))$$

$$R(y(c), z(y(c))), D(z(y(c)))$$

W

y

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

$$B(x) \wedge C(x) \rightarrow E(x)$$

$$S(x, y) \rightarrow C(x) \qquad A(c)$$

$$S(c, y(c)), B(y(c))$$

$$R(y(c), z(y(c))), D(z(y(c)))$$

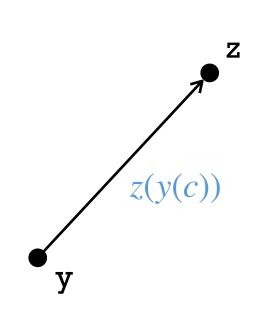
W

Ζ

z(y(c))

V

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$



W

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

$$B(x) \wedge C(x) \rightarrow E(x)$$

$$S(x, y) \rightarrow C(x)$$

Z *z(y(c))* y

W

$$\begin{array}{l} \mathsf{A}(x) \to \mathsf{S}(x, y(x)) \land \mathsf{B}(y(x)) \\ & \mathsf{B}(x) \to \mathsf{R}(x, z(x)) \land \mathsf{D}(z(x)) \\ & \mathsf{D}(x) \to \mathsf{E}(x) \\ & \mathsf{D}(x) \to \mathsf{E}(x) \\ & \mathsf{E}(x) \to \mathsf{R}(x, w(x)) \end{array}$$
$$\begin{array}{l} \mathsf{B}(x) \land \mathsf{C}(x) \to \mathsf{E}(x) \\ & \mathsf{S}(x, y) \to \mathsf{C}(x) \end{array} \qquad \begin{array}{l} \mathsf{B}(c) \\ & \mathsf{R}(c, z(c)), D(z(c)), \\ & \mathsf{E}(z(c)), \\ & \mathsf{R}(z(c), w(z(c))) \end{array}$$

y

z(y(c))

 \mathbf{Z}

W

Reasoning over Existential Rules with Acyclicity Notions

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

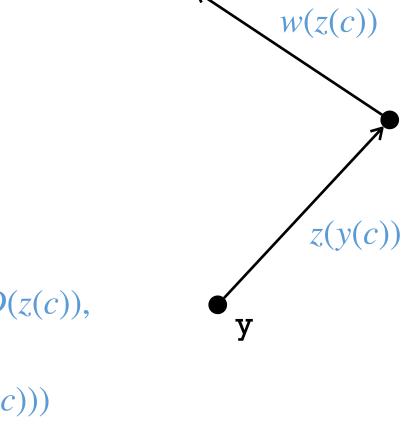
$$B(x) \wedge C(x) \rightarrow E(x)$$

$$S(x, y) \rightarrow C(x)$$

$$B(c)$$

$$R(c, z(c)), D(z(c))$$

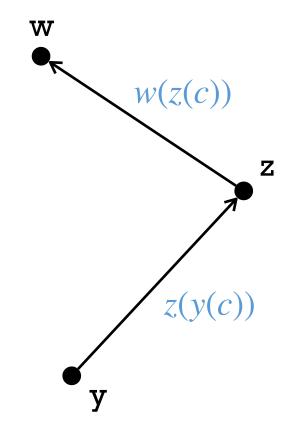
$$R(z(c), w(z(c)))$$



W

 \mathbf{Z}

$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$



$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$

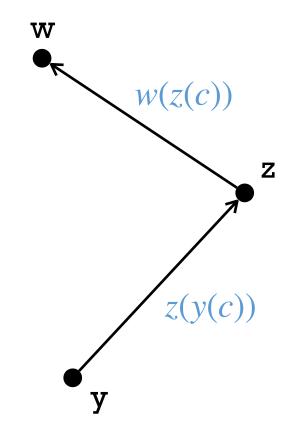
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$

$$D(x) \rightarrow E(x)$$

$$E(x) \rightarrow R(x, w(x))$$

$$B(x) \wedge C(x) \rightarrow E(x)$$

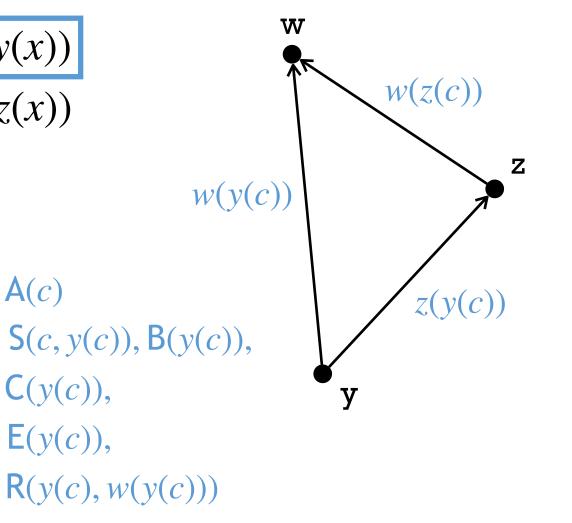
$$S(x, y) \rightarrow C(x)$$



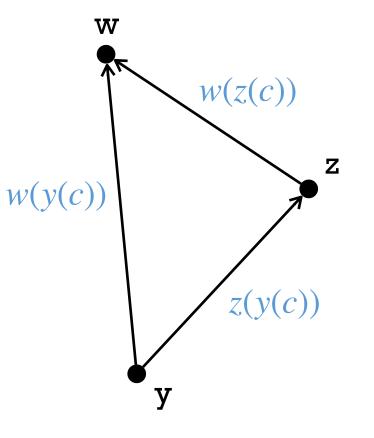
$$\begin{array}{c} \mathsf{A}(x) \to \mathsf{S}(x, y(x)) \land \mathsf{B}(y(x))) \\ \mathsf{B}(x) \to \mathsf{R}(x, z(x)) \land \mathsf{D}(z(x)) \\ \mathsf{D}(x) \to \mathsf{E}(x) \\ \mathsf{E}(x) \to \mathsf{R}(x, w(x)) \\ \mathsf{B}(x) \land \mathsf{C}(x) \to \mathsf{E}(x) \\ \mathsf{S}(x, y) \to \mathsf{C}(x) \end{array} \qquad \begin{array}{c} \mathsf{A}(c) \\ \mathsf{S}(c, y(c)), \mathsf{B}(y(c)), \\ \mathsf{C}(y(c)), \\ \mathsf{E}(y(c)), \end{array}$$

 $\mathsf{R}(y(c), w(y(c)))$

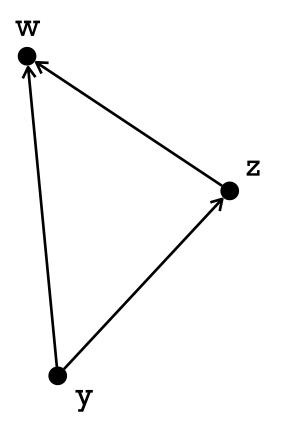
$$\begin{array}{l} \mathsf{A}(x) \to \mathsf{S}(x, y(x)) \land \mathsf{B}(y(x)) \\ \mathsf{B}(x) \to \mathsf{R}(x, z(x)) \land \mathsf{D}(z(x)) \\ \mathsf{D}(x) \to \mathsf{E}(x) \\ \mathsf{E}(x) \to \mathsf{R}(x, w(x)) \\ \mathsf{B}(x) \land \mathsf{C}(x) \to \mathsf{E}(x) \\ \mathsf{S}(x, y) \to \mathsf{C}(x) \end{array} \xrightarrow{\mathsf{A}(c)} \\ \begin{array}{l} \mathsf{A}(c) \\ \mathsf{S}(c, y(c)) \\ \mathsf{C}(y(c)), \end{array}$$

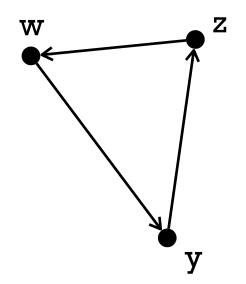


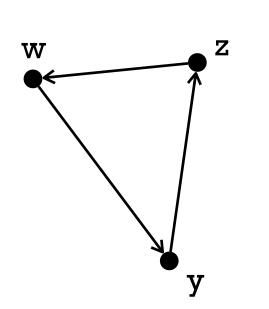
$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$



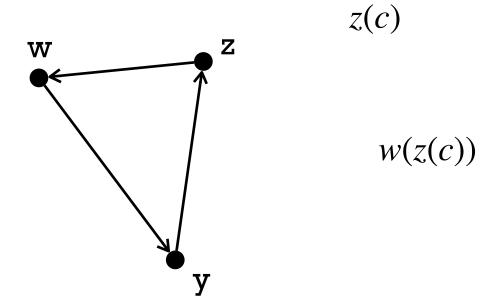
$$A(x) \rightarrow S(x, y(x)) \wedge B(y(x))$$
$$B(x) \rightarrow R(x, z(x)) \wedge D(z(x))$$
$$D(x) \rightarrow E(x)$$
$$E(x) \rightarrow R(x, w(x))$$
$$B(x) \wedge C(x) \rightarrow E(x)$$
$$S(x, y) \rightarrow C(x)$$

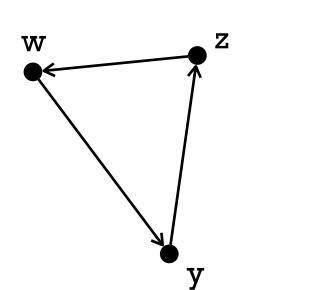






z(c)

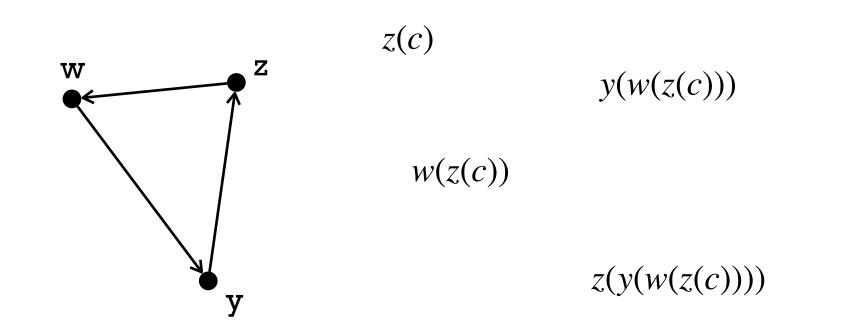




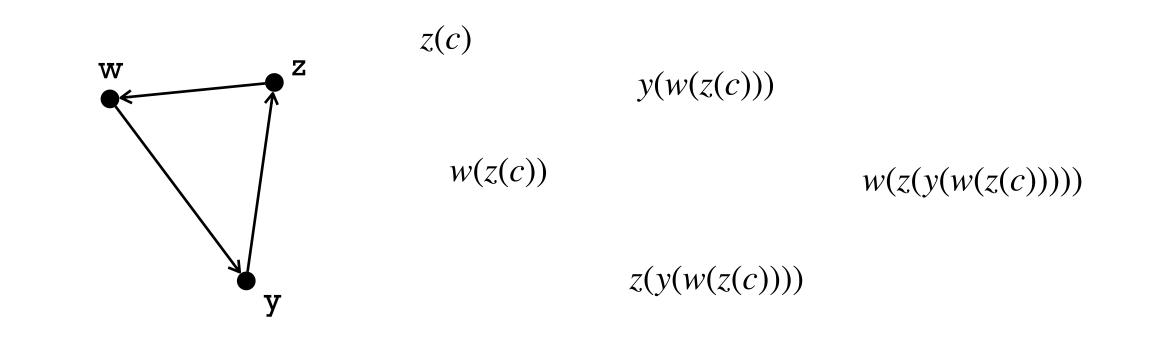
z(c)

y(w(z(c)))

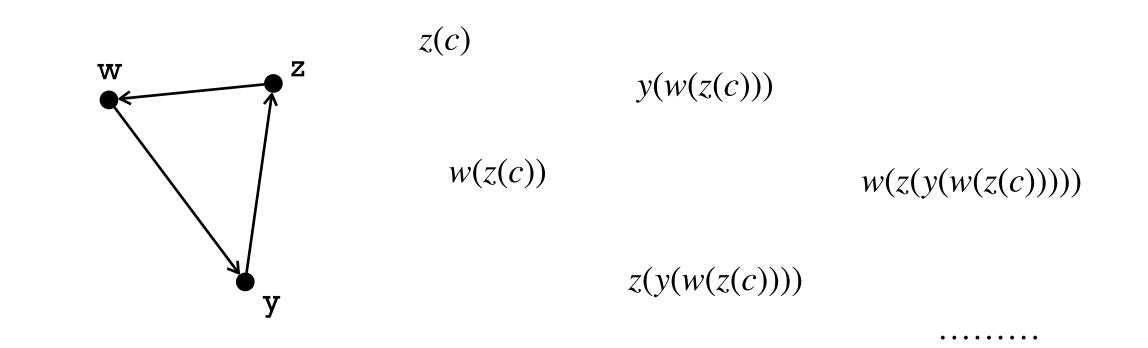
w(z(c))



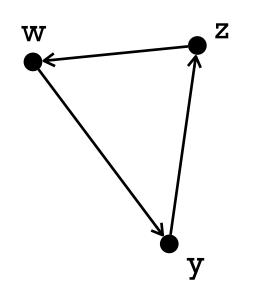
David Carral



David Carral



David Carral



Remark. If the existential dependency graph of a given set of rules is acyclic, then the set of terms introduced during the computation of the chase is finite.

Film(x) $\rightarrow \exists y$. IsFilmDirectedBy(x, y) \wedge Director(y) A(x) \wedge B(x, w) \wedge C(x, z) $\rightarrow \exists z . R(x, w, z)$

Film(x) → ∃y.IsFilmDirectedBy(x, y) ∧ Director(y) Film(x) → IsFilmDirectedBy(x, y(x)) ∧ Director(y(x)) $A(x) \land B(x, y) \land C(x, z) \rightarrow \exists z . R(x, y, z)$ $A(x) \land B(x, w) \land C(x, z) \rightarrow R(x, w, z(x, w))$

Film(x) $\rightarrow \exists y$. IsFilmDirectedBy(x, y) \wedge Director(y) Film(x) \rightarrow IsFilmDirectedBy(x, y(x)) \wedge Director(y(x)) A(x) \wedge B(x, y) \wedge C(x, z) $\rightarrow \exists z$. R(x, y, z) A(x) \wedge B(x, w) \wedge C(x, z) \rightarrow R(x, w, z(x, w))

Remark. If the arity of every function symbol in the skolemisation of a program is at most 1, then every term in the chase is of the form xl(...xn(c)...) with c constant.

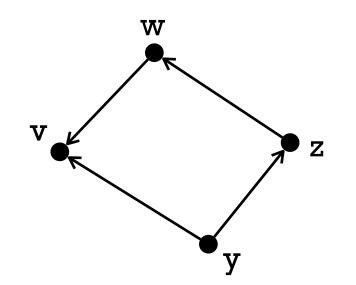
Remark. If the arity of every function symbol in the skolemisation of a program is at most 1, then every term in the chase is of the form xl(...xn(c)...) with c constant.

Remark. If the arity of every function symbol in the skolemisation of a program is at most 1, then every term in the chase is of the form xl(...xn(c)...) with c constant.

Corollary. Every term occurring in the chase corresponds to a path in the dependency graph and a constant.

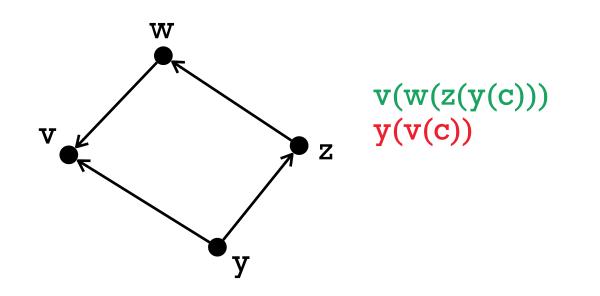
Remark. If the arity of every function symbol in the skolemisation of a program is at most 1, then every term in the chase is of the form xl(...xn(c)...) with c constant.

Corollary. Every term occurring in the chase corresponds to a path in the dependency graph and a constant.



Remark. If the arity of every function symbol in the skolemisation of a program is at most 1, then every term in the chase is of the form xl(...xn(c)...) with c constant.

Corollary. Every term occurring in the chase corresponds to a path in the dependency graph and a constant.



Reasoning over Existential Rules with Acyclicity Notions

(a) The dependency graph is acyclic.

(a) The dependency graph is acyclic.

(f) The arity of all function symbols in the skolemisation of the program is at most 1.

(a) The dependency graph is acyclic.

(f) The arity of all function symbols in the skolemisation of the program is at most 1. All skolem terms correspond to some path in the dependency graph and some constant

(a) The dependency graph is acyclic.

(f) The arity of all function symbols in the skolemisation of the program is at most 1. All skolem terms correspond to some path in the dependency graph and some constant

(w) The number of variables per rule is bounded.

(a) The dependency graph is acyclic.

(f) The arity of all function symbols in the skolemisation of the program is at most 1. All skolem terms correspond to some path in the dependency graph and some constant

Rules can be applied

(w) The number of variables per rule is bounded.

The number of facts is polynomial in the number of terms

in polynomial time

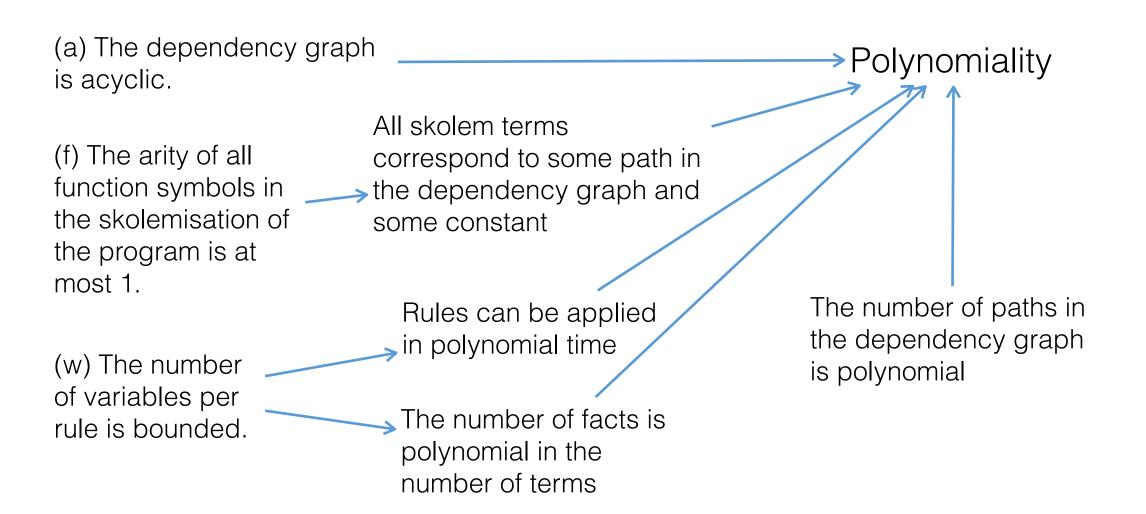
(a) The dependency graph is acyclic.

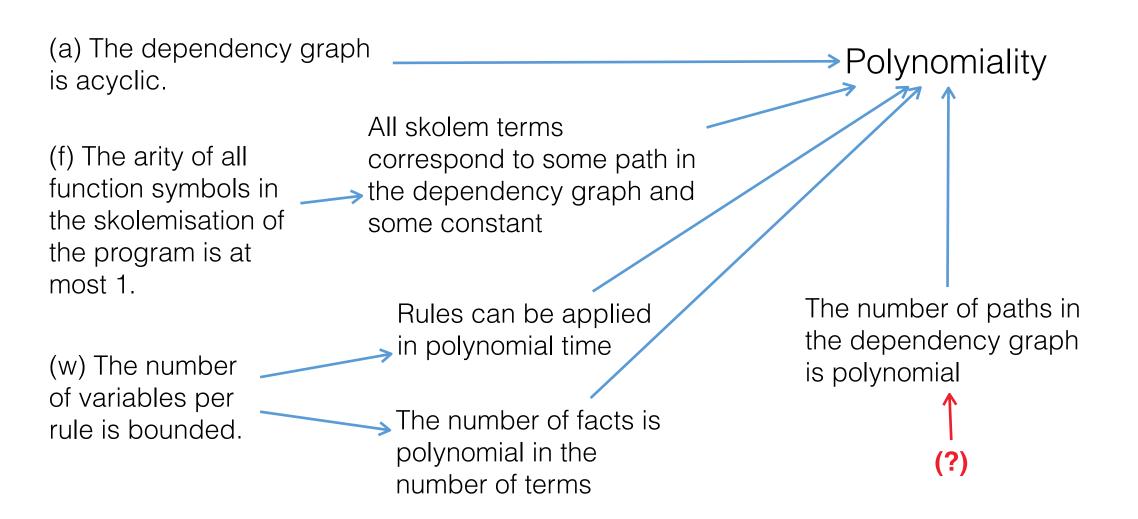
(f) The arity of all function symbols in the skolemisation of the program is at most 1. All skolem terms correspond to some path in the dependency graph and some constant

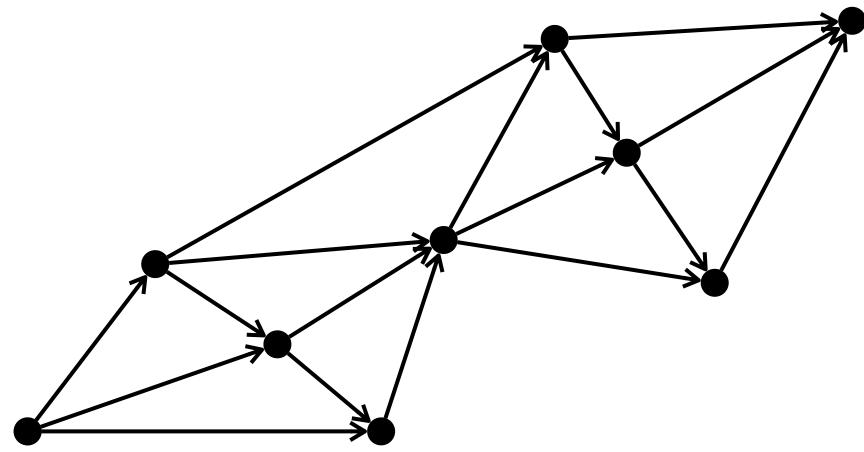
(w) The number of variables per rule is bounded. Rules can be applied in polynomial time

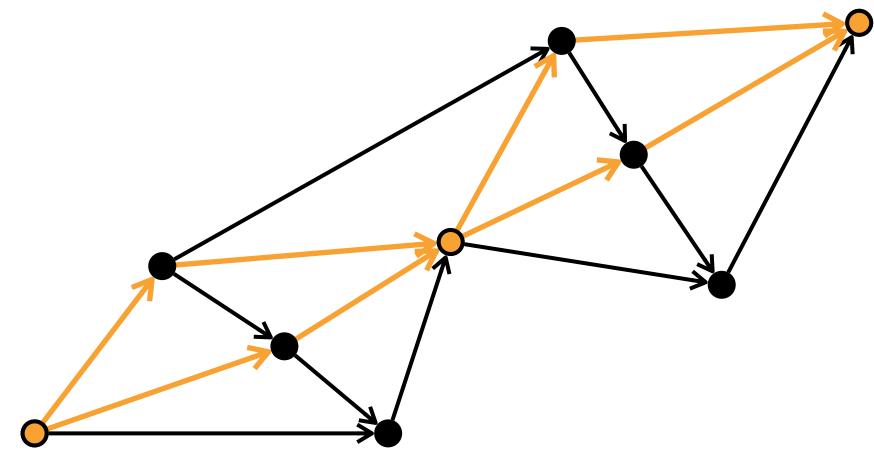
The number of facts is polynomial in the number of terms

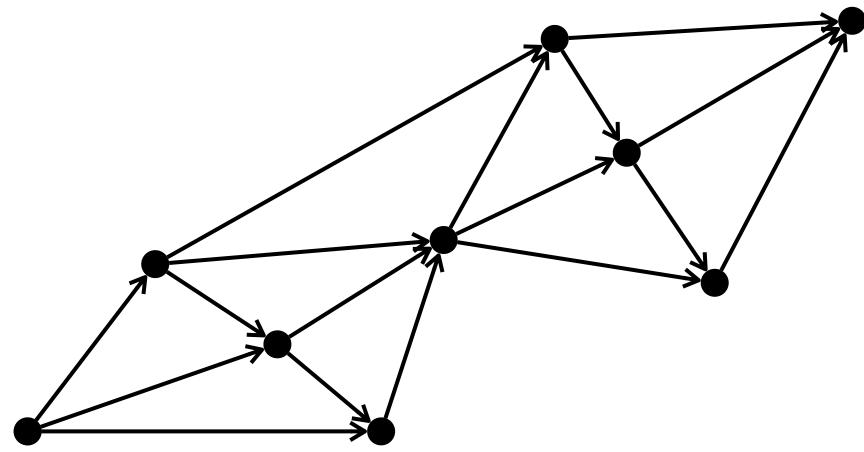
The number of paths in the dependency graph is polynomial

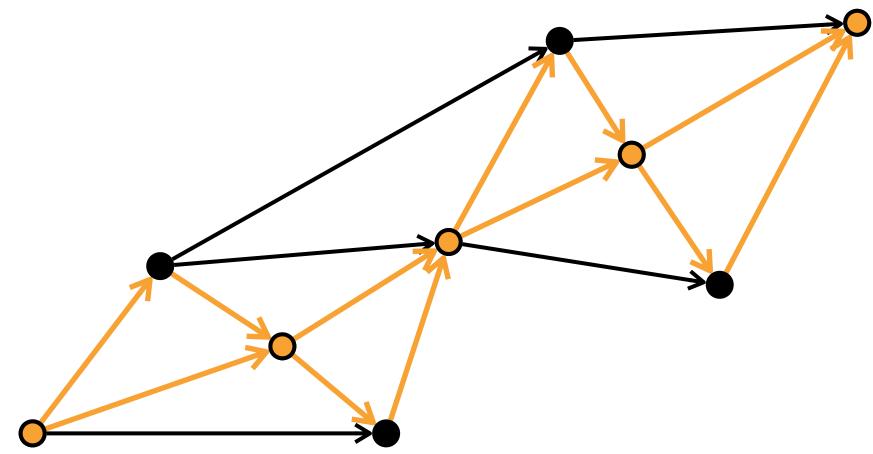


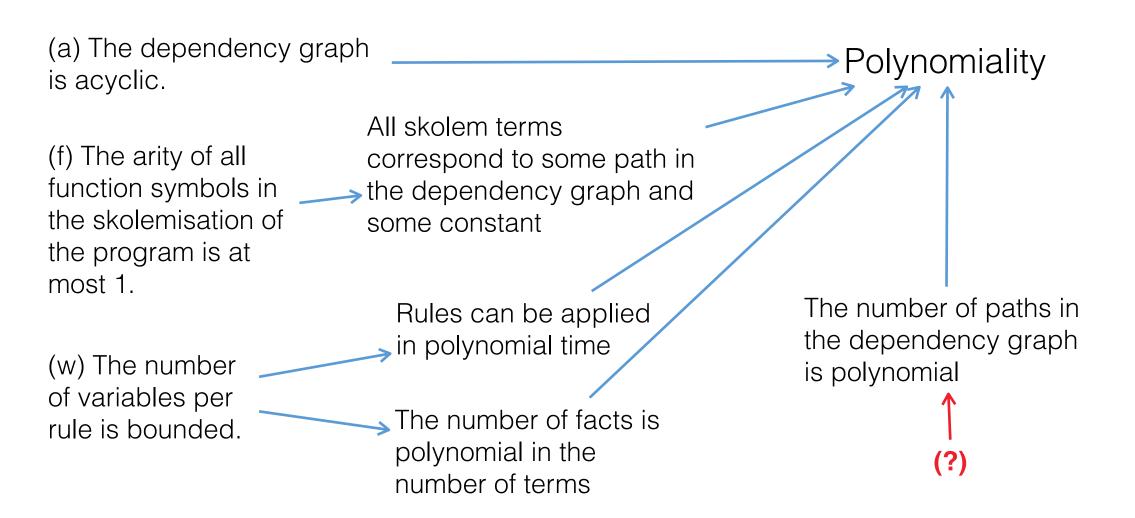


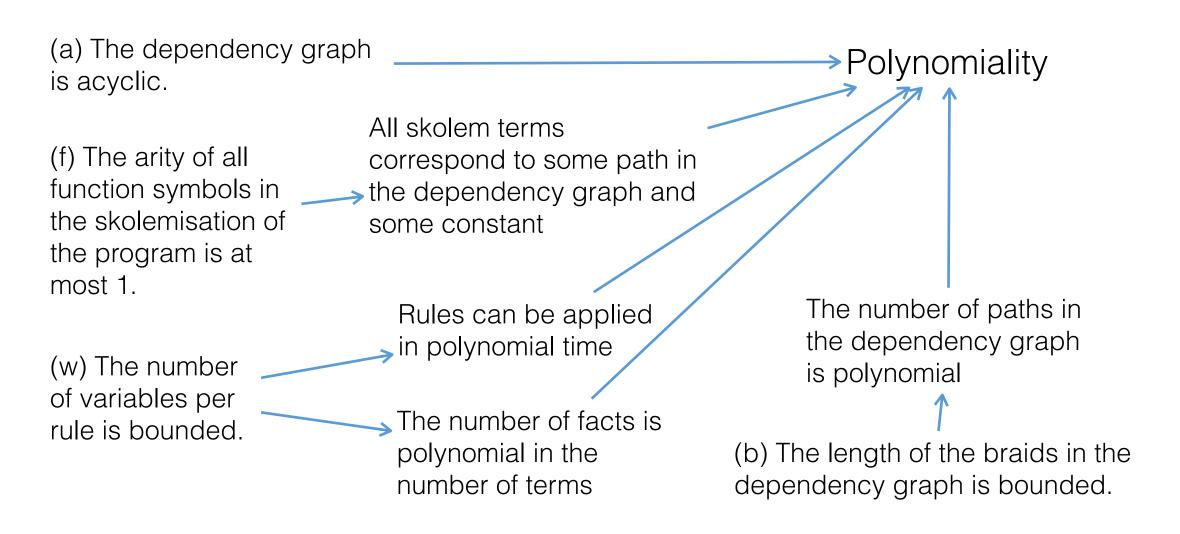


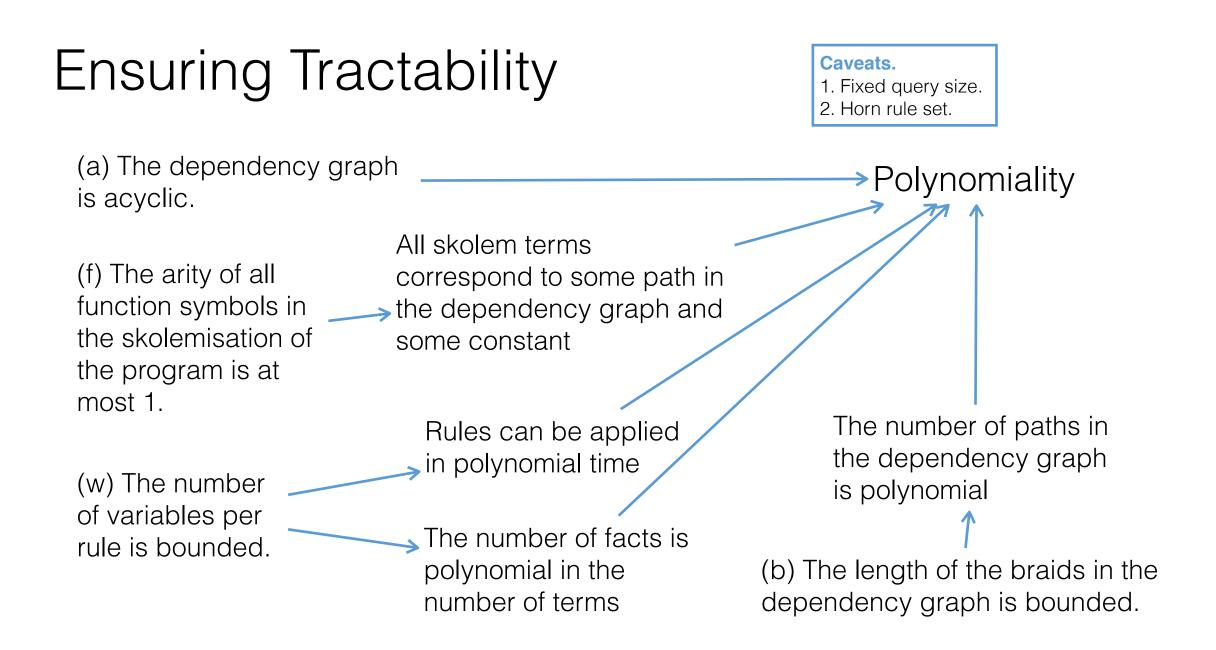












$$\begin{array}{rcl} A_{1}\sqcap\ldots\sqcap A_{n}\sqsubseteq B &\mapsto& \bigwedge_{i=1}^{n}A_{i}(x)\rightarrow B(x)\\ A\sqsubseteq B_{1}\sqcup\ldots\sqcup B_{n} &\mapsto& A(x)\rightarrow\bigvee_{i=1}^{n}B_{i}(x)\\ A\sqsubseteq\forall R.B &\mapsto& A(y)\wedge R(x,y)\rightarrow B(x)\\ A\sqsubseteq\exists R.B &\mapsto& A(x)\rightarrow\exists y.R(x,y)\wedge B(y)\\ R\sqsubseteq S &\mapsto& R(x,y)\rightarrow S(x,y)\\ R\circ S\sqsubseteq V &\mapsto& R(x,y)\wedge S(y,z)\rightarrow S(x,z)\\ R_{1}\sqcap\ldots\sqcap R_{n}\sqsubseteq S &\mapsto& \bigwedge_{i=1}^{n}R_{i}(x,y)\rightarrow S(x,y)\\ A(a) &\mapsto& A(a)\\ R(a,b) &\mapsto& R(a,b) \end{array}$$

$$\begin{array}{rcl} A_{1}\sqcap \ldots \sqcap A_{n}\sqsubseteq B & \mapsto & \bigwedge_{i=1}^{n}A_{i}(x) \rightarrow B(x) \\ A\sqsubseteq B_{1}\sqcup \ldots \sqcup B_{n} & \mapsto & A(x) \rightarrow \bigvee_{i=1}^{n}B_{i}(x) \\ & A\sqsubseteq \forall R.B & \mapsto & A(y) \wedge R(x,y) \rightarrow B(x) \\ & A\sqsubseteq \exists R.B & \mapsto & A(x) \rightarrow \exists y.R(x,y) \wedge B(y) \\ & R\sqsubseteq S & \mapsto & R(x,y) \rightarrow S(x,y) \\ & R \circ S \sqsubseteq V & \mapsto & R(x,y) \wedge S(y,z) \rightarrow S(x,z) \\ & R_{1}\sqcap \ldots \sqcap R_{n}\sqsubseteq S & \mapsto & \bigwedge_{i=1}^{n}R_{i}(x,y) \rightarrow S(x,y) \\ & & A(a) & \mapsto & A(a) \\ & & R(a,b) & \mapsto & R(a,b) \end{array}$$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

 $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \mapsto \bigwedge^n A_i(x) \to B(x)$ $A \sqsubseteq B_1 \sqcup \ldots \sqcup B_n \mapsto A(x) \to \bigvee^n B_i(x)$ $A \sqsubseteq \forall R . B \mapsto A(y) \land R(x, y) \rightarrow B(x)$ $A \sqsubseteq \exists R . B \mapsto A(x) \rightarrow \exists y . R(x, y) \land B(y)$ $R \sqsubseteq S \mapsto R(x, y) \rightarrow S(x, y)$ $R \circ S \sqsubseteq V \mapsto R(x, y) \land S(y, z) \to S(x, z)$ $R_1 \sqcap \ldots \sqcap R_n \sqsubseteq S \mapsto \bigwedge^n R_i(x, y) \to S(x, y)$ i=1 $A(a) \mapsto A(a)$ $R(a,b) \mapsto R(a,b)$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

Remark 2.

1. SRI rules feature at most 3 variables.

$$\begin{array}{rcl} A_{1}\sqcap \ldots \sqcap A_{n} \sqsubseteq B & \mapsto & \bigwedge_{i=1}^{n} A_{i}(x) \rightarrow B(x) \\ A \sqsubseteq B_{1} \sqcup \ldots \sqcup B_{n} & \mapsto & A(x) \rightarrow \bigvee_{i=1}^{n} B_{i}(x) \\ A \sqsubseteq \forall R . B & \mapsto & A(y) \land R(x, y) \rightarrow B(x) \\ A \sqsubseteq \exists R . B & \mapsto & A(x) \rightarrow \exists y . R(x, y) \land B(y) \\ R \sqsubseteq S & \mapsto & R(x, y) \rightarrow S(x, y) \\ \hline R \circ S \sqsubseteq V & \mapsto & R(x, y) \land S(y, z) \rightarrow S(x, z) \\ R_{1}\sqcap \ldots \sqcap R_{n} \sqsubseteq S & \mapsto & \bigwedge_{i=1}^{n} R_{i}(x, y) \rightarrow S(x, y) \\ A(a) & \mapsto & A(a) \\ R(a, b) & \mapsto & R(a, b) \end{array}$$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

Remark 2.

1. SRI rules feature at most 3 variables.

 $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \mapsto \bigwedge^n A_i(x) \to B(x)$ $A \sqsubseteq B_1 \sqcup \ldots \sqcup B_n \mapsto A(x) \to \bigvee^n B_i(x)$ $A \sqsubseteq \forall R . B \mapsto A(y) \land R(x, y) \rightarrow B(x)$ $A \sqsubseteq \exists R . B \mapsto A(x) \rightarrow \exists y . R(x, y) \land B(y)$ $R \sqsubseteq S \mapsto R(x, y) \rightarrow S(x, y)$ $R \circ S \sqsubseteq V \mapsto R(x, y) \land S(y, z) \to S(x, z)$ $R_1 \sqcap \ldots \sqcap R_n \sqsubseteq S \mapsto \bigwedge^n R_i(x, y) \to S(x, y)$ i=1 $A(a) \mapsto A(a)$ $R(a,b) \mapsto R(a,b)$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

Remark 2.

1. SRI rules feature at most 3 variables.

 $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \mapsto \bigwedge^n A_i(x) \to B(x)$ $A \sqsubseteq B_1 \sqcup \ldots \sqcup B_n \mapsto A(x) \to \bigvee^n B_i(x)$ $A \sqsubseteq \forall R . B \mapsto A(y) \land R(x, y) \rightarrow B(x)$ $A \sqsubseteq \exists R . B \mapsto A(x) \rightarrow \exists y . R(x, y) \land B(y)$ $R \sqsubseteq S \mapsto R(x, y) \rightarrow S(x, y)$ $R \circ S \sqsubseteq V \mapsto R(x, y) \land S(y, z) \to S(x, z)$ $R_1 \sqcap \ldots \sqcap R_n \sqsubseteq S \mapsto \bigwedge^n R_i(x, y) \to S(x, y)$ i=1 $A(a) \mapsto A(a)$ $R(a,b) \mapsto R(a,b)$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

Remark 2.

1. SRI rules feature at most 3 variables.

 $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \mapsto \bigwedge^n A_i(x) \to B(x)$ $A \sqsubseteq B_1 \sqcup \ldots \sqcup B_n \mapsto A(x) \to \bigvee^n B_i(x)$ $A \sqsubseteq \forall R . B \mapsto A(y) \land R(x, y) \rightarrow B(x)$ $A \sqsubseteq \exists R . B \mapsto A(x) \rightarrow \exists y . R(x, y) \land B(y)$ $R \sqsubseteq S \mapsto R(x, y) \rightarrow S(x, y)$ $R \circ S \sqsubseteq V \mapsto R(x, y) \land S(y, z) \to S(x, z)$ $R_1 \sqcap \ldots \sqcap R_n \sqsubseteq S \mapsto \bigwedge^n R_i(x, y) \to S(x, y)$ i=1 $A(a) \mapsto A(a)$ $R(a,b) \mapsto R(a,b)$

Remark 1. Deciding CQ entailment for SRI ontologies is 2ExpTime-Hard and in 3ExpTime.

Remark 2.

1. SRI rules feature at most 3 variables.

SRI Axioms

Remark 2.

1. Every rule in an SRI ontology has at most 3 variables. 2. Every function symbol in the skolemisation of a SRI ontology has arity one

SRI Axioms

Remark 2.

1. Every rule in an SRI ontology has at most 3 variables. 2. Every function symbol in the skolemisation of a SRI ontology has arity one

Corollary. To guarantee that tractable CQ entailment over a SRI ontology is possible we only need to verify the following: *1. Acyclicity.*

2. Braid length in the dependency graph is bounded.

Evaluation Results

Evaluation Results

Acyclicity

| | MOWL Corpus | Oxford Ontology Repo |
|------------|-------------|----------------------|
| Ontologies | 1576 | 225 |
| Acyclic | 974 (61.8%) | 170 (75.6%) |

Evaluation Results

Acyclicity

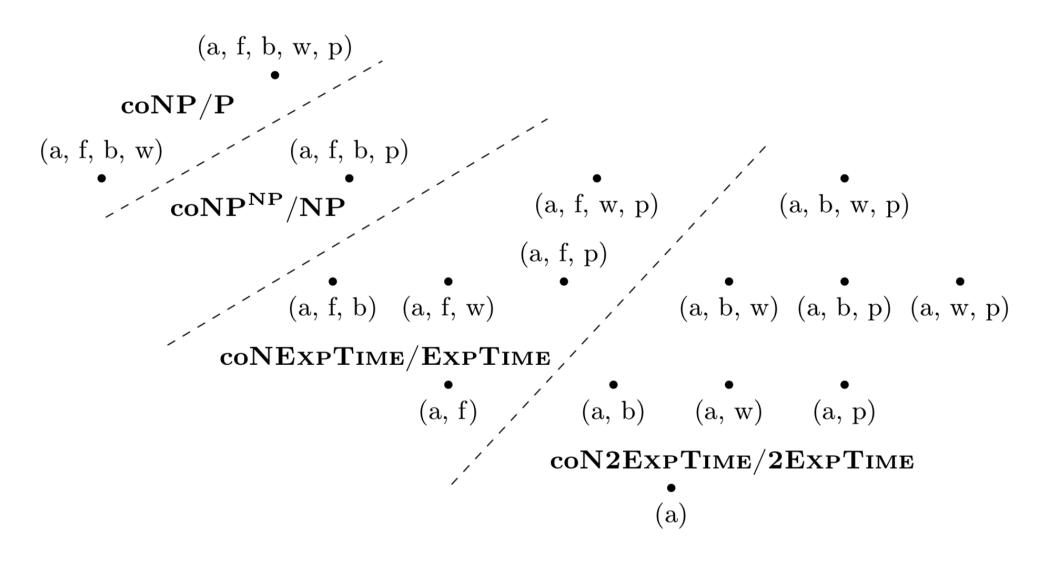
| | MOWL Corpus | Oxford Ontology Repo |
|------------|-------------|----------------------|
| Ontologies | 1576 | 225 |
| Acyclic | 974 (61.8%) | 170 (75.6%) |

Braid Length

MOWL Corpus + Oxford

| (max. length of a braid) | 1 | 2 | 3 | 4 | 5 | 6 | 11 | 22 | 23 | 25 | Total |
|-----------------------------|-----|-----|----|----|----|----|----|------|------|------|-------|
| (count) | 851 | 153 | 56 | 61 | 11 | 1 | 1 | 2 | 7 | 1 | 1144 |
| | 74 | 88 | 93 | 98 | 99 | 99 | 99 | 99.1 | 99.3 | 99.9 | 100 |

More Results!



David Carral





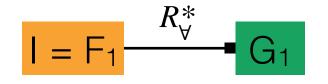
Efficient Model Construction for Horn Logic with VLog — System Description Jacopo Urbani, Markus Krötzsch, Ceriel J. H. Jacobs, Irina Dragoste, and <u>David Carral</u> [IJCAR 2018]

Consider a rule set *R* and an instance *I*.

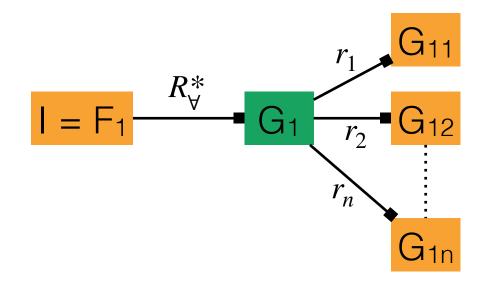
Consider a rule set *R* and an instance *I*.



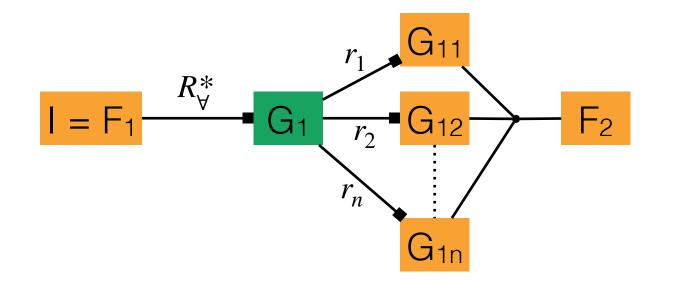
Consider a rule set *R* and an instance *I*.



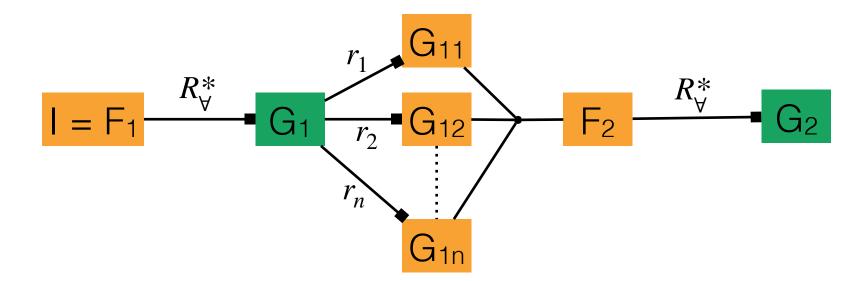
Consider a rule set *R* and an instance *I*.



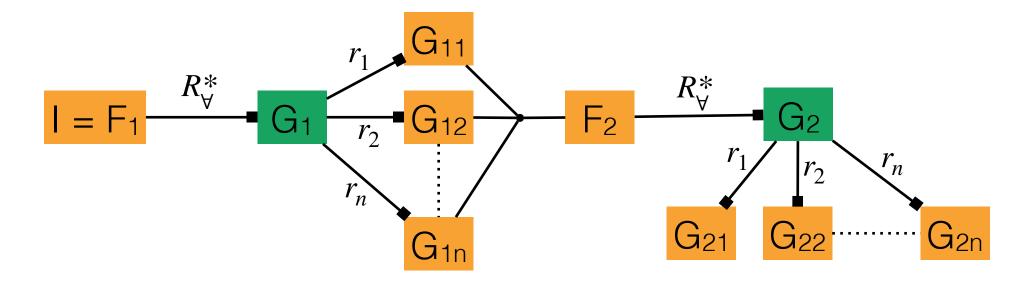
Consider a rule set *R* and an instance *I*.



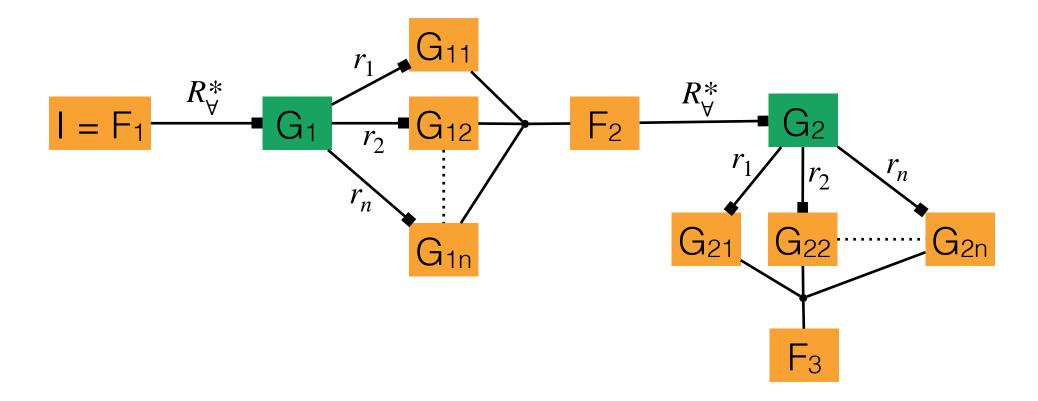
Consider a rule set *R* and an instance *I*.



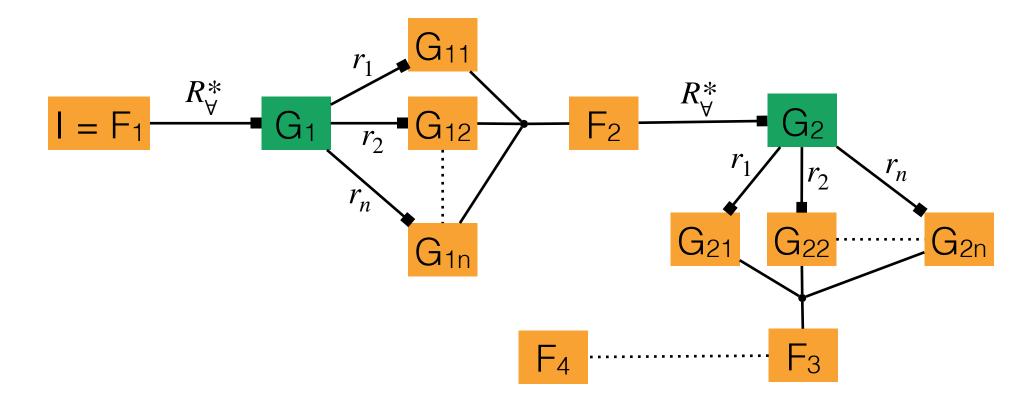
Consider a rule set *R* and an instance *I*.



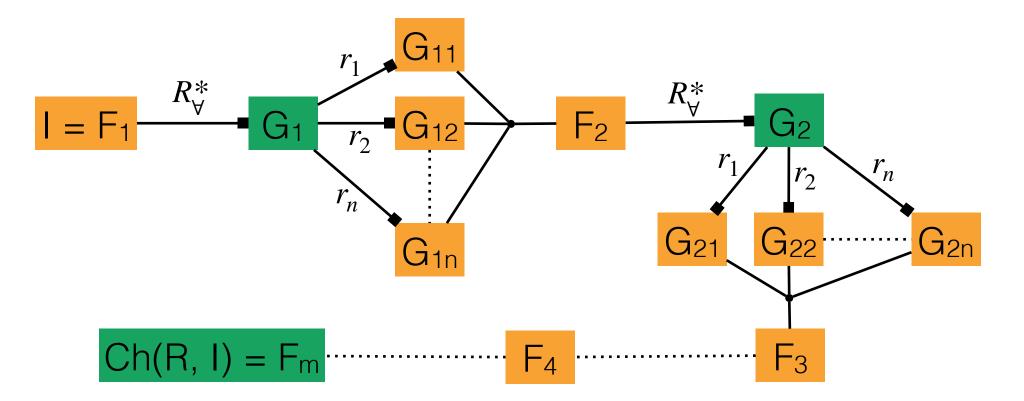
Consider a rule set *R* and an instance *I*.



Consider a rule set *R* and an instance *I*.



Consider a rule set *R* and an instance *I*.



Performance: VLog vs RDFox

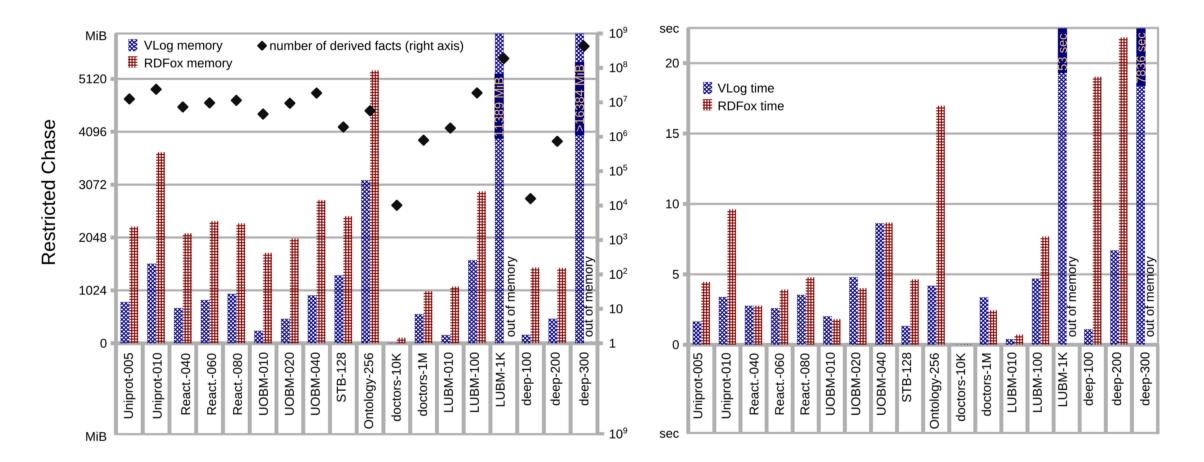


Fig. 1. Memory usage (left) and materialisation time (right) for VLog and RDFox

Conclusions

This is it, everybody should use existential rules + acyclicity notions!

This is it, everybody should use existential rules + acyclicity notions! Surely not, a lot of work still needs to be done:

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

* Experiments with non-DL existential rule sets

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets

| ile Edit Source Refactor Navigate S | /home/markus/Documents/demo-rules.dpg - Rule IDE earch Project Run Window Help | | |
|--|---|---|--------------|
| 🗅 v 🖻 v 🗟 🖄 🔺 🍫 | · O · Q · B G · B G · A · P B B I B I · S · + + + + | ~ 🖻 🛛 🔍 | Quick Access |
| Package Explorer X P Constraints. Project Proj | <pre>@ drone-control.dpg @ demo-rules.dpg M</pre> | System 12 Resource Monitor Coccepted RAME 1419/14 8% Derivation steps | |
| | Writable Smart Insert 25:12 | | |

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- Develop tools that can assist knowledge * engineers in "repairing" non-acyclic rule sets

| | /home/markus/Docume | ents/demo-rules. | dpg - Rule IDE | | | | | | | |
|---|---|---|---|---------------------------|----------------|---|--|-----------|----------|----|
| File Edit Source Refactor Navigate | Search Project Run Window Help | | | | | | | | | |
| 📫 × 🖻 × 🗟 🙆 🔌 🍫 | × 0 × % × ₩ G × ⊅ ⊖ A | × 🖗 🗾 🕯 | 0 M 🕴 | v § v ¢⊃ ¢ | 5 × ¢ | × ₫ | Q Quick Acc | ess | | |
| I* Package Explorer IX I* I* I* | <pre>@ drone-control.dpg</pre> | <pre>rg/transitivi al chain as p min0(c0). ma evel 0:) . > succl(2z 2z </pre> | ty-example/> per [Cali, Go ux0(c1). | | Т | System 23 Resource Mon Occupied RAM: 1: 8% Derivation sta | 419/16384 MB | | | |
| | <pre>% Constructing level 2 from 1 r1(r2, r1(r2) ~ > 1(r2, ry, r2) a1(r2, ry, r2) ~> 1(r2, ry, r2) a1(r2, ry, r2), a1(r2, ry, r2, r2) min1(r2), a1(r2, r2, r2), r2) max1(r2), a1(r2, r2, r2), r2) max1(r2), a1(r2, r2, r2) ~ max </pre> | <pre>.) . succ1(?y,?yr, , max1(?y), m (2(?y)2(?y)3(?y) . Search Conso ces connected</pre> | nin1(?yn), su | x,?m) . ccl(?x,?xn) -> | 91 | x → 1 35 (→ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | real and the second sec | | | |
| | | Writable | Smart Insert | 25:12 | | | | | | |
| | | Search I | tem | c | Q Start | Properties | Classes | Rules | About | Lo |
| Rules I spouse is symm | | | | | | | materialisa | ble conse | equences | |
| (?x.spous → (?y.sp | e ^{P26} = ?y)@?S ouse ^{P26} = ?x)@?S | | | | | | | | | |
| (?father.s | afather child ^{P40} = ?child)@?X, sex or gender ^{P21} = male ⁰⁶⁵⁸¹⁰⁹⁷)@? d.father ^{P22} = ?father) | Y | | | | | materialisa | ble conse | equences | |
| A female parent | is a mother | | | | | | materialisa | ble cons | equences | |

a male child of a child is a grandson

(?mother.child^{P40} = ?child)@?X, (?mother.sex or gender^{P21} = female^{Q6581072})@?Y \rightarrow (?child.mother^{P25} = ?mother) a male parent of a parent is a grandfather

(?grandfather.sex or gender^{P21} = male^{Q6581097})@?X, (?grandfather.child^{P40} = ?parent)@?Y, (?parent.child^{P40} = ?child)@?Z

(?son.sex or gender^{P21} = male^{Q6581097})@?X

→ (?child.relative^{P1038} = ?grandfather)@{type of kinship^{P1039} = grandfather^{Q9238344}}

informational consequences

informational consequences

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets

| ile Edit Source Refactor Navigate S | /home/markus/Documents/demo- earch Project Run Window Help | rules.dpg - Rule IDE | |
|--|---|---|--|
| | × 0 × % × ⊕ G × Ø G ≁ × Ø . | ∛ ≈ 0 1 <u>9</u> × 0 × 0 ↔ < | 🛇 🖂 🖂 Quick Access |
| Package Explorer R Package Explorer R Rule IDE example Project Frace [1,3M facts] Frace [1,3M facts] Frace [0,3M facts] | <pre>D drone-control.dgg D demo-rules.dgg H demo.rules.dgg H e SBASS <file: (::="" documents="" erestrix="" home="" http:="" http:<="" markus="" th="" trans="" xxmple.org=""><th><pre>tivity-example/></pre></th><th>System 23 Concerned with All Single Mass By Concerned with All Single Mass By Concerned with All Single Mass Concerned with</th></file:></pre> | <pre>tivity-example/></pre> | System 23 Concerned with All Single Mass By Concerned with All Single Mass By Concerned with All Single Mass Concerned with |
| SQID | | earch Item Q Str | art Properties Classes Rules Abou |
| | | | materialisable consequence |
| A male parent is (?father.c (?father.s | | | materialisable consequence |
| A female parent i (?mother.c (?mother.s | | https://tools.wmfl | abs.org/sqid/ |
| | a parent is a grandfather | Y70 | |

Lo

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets
- * Efficient implementation for acyclicity checks

| ile Edit Source Refactor Navigate | | cuments/demo-rules. | dpg - Rule IDE | | | | | | |
|--|--|--|---|----------|---------|--|----------------------|----------|---------|
| | | 🔗 v 🛛 😵 🌛 😜 | u n 🕴 | v ∦ v ¢⊃ | ф × ф | × 13 | Quick Acces | 5 S | |
| Package Explorer 32 Image: Constraints.dpg Image | ☐ drone-controldpp ☐ demo-rules.dpg ● demo-science.dpg ● demo-rules.dpg ● dem | <pre>a/Documents/demo- le.org/transitiv; ential chain as p on lavel 0: , iz) - min0(c0), ms on lavel 0: , iz) - succl(7z,7z) ; on lavel 1: y, iz) - min2(2y) - min2(2y) - min2(2y) - min2(2y) - second constant & Search Conso slources connected Se</pre> | <pre>ity-example/> per [Cali, Gof ax0(cl). an) . an) . an) . ani .</pre> | | - | Statem 23 | itor 119/16384 MB | | |
| | | Writable | Smart Insert | 25:12 | | | | | |
| Rules I | list | Search | ltem | | Q Start | Properties | Classes | Rules | About |
| spouse is symm (?x. spouse → (?y. sp | metric se ^{P26} = ?y)@?S pouse ^{P26} = ?x)@?S | | | | | | materialisab | le conse | quences |
| A male parent is (?father. (?father.s | | ')@?Y | | | | | materialisabl | le conse | quences |
| | | | | | | and the second sec | | | |
| (?mother.s → (?chil | <pre>tis a mother .child^{P40} = ?child)@?X, sex or gender^{P21} = female⁰⁶⁵⁸¹¹ ld.mother^{P25} = ?mother) of a parent is a grandfather</pre> | | https:// | tools.v | wmfla | lbs.org | /sqid/ | | |
| (?mother. (?mother. → (?chil a male parent of (?grandfat (?grandfat (?grandrat, (?grandr. (?grandr.))) | .child ^{P40} = ?child)@?X, sex or gender ^{P21} = female ^{Q6581} ld.mother ^{P25} = ?mother) | ⁰⁷²)@?Y ²⁶⁵⁸¹⁰⁹⁷)@?X, | https:// | tools. | wmfla | ibs.org | /sqid/ | | |

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets
- * Efficient implementation for acyclicity checks
- * Optimise chase reasoners

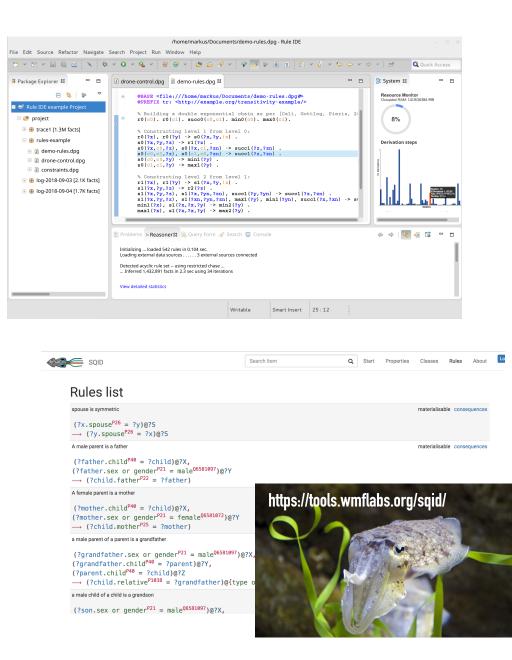
| | /home/markus/Doc | uments/demo-rules | .dpg - Rule IDE | | | | | | |
|--|---|---|---|--------------|--------------|----------------------|---|-------|---------|
| | Search Project Run Window Help ⊧ v Q v Q v ⊕ G v 🥭 ⊝ 🦯 | 4 × 19 🗾 1 | | v ∰ v %⊃ (| þ v ¢ | × B | Q Quick Acc | ess | |
| Package Explorer 20 Image: Complex Compl | donne-control.dgg demo-rules.dgg B demo-rules.dgg B Bland demo-rules.dgg B | <pre>//ocuments/demo e.org/transitiv ntial chain as mi lavel 0: /il) -> succl(72,7) -> succl(72,7) -> succl(72,7 mlavel 1: /il) - n), macl(72,7 min2(72) - macl(72,7 min2(72) - sources connected ></pre> | <pre>ity-example/> per [Cali, Got ax0(cl). zn) . zn) . n) -> succl(?? minl(?yn), suc</pre> | tlog, Pieris | | 8% Derivation ste | am II II IIIIIIIIIIIIIIIIIIIIIIIIIIIIII | | |
| SQID | | Writable | Smart Insert Item | 25:12 | Q Star | t Properties | Classes | Rules | About |
| → (?y.sp | netric se ^{P26} = ?y)@?S pouse ^{P26} = ?x)@?S | | | | | | | | |
| (?father.: → (?chil A female parent (?mother. → (?chil a male parent of (?grandfa (?grandfa (?grandfa (?parent. → (?chil a male parent of | .child ^{P40} = ?child)@?X, sex or gender ^{P21} = male ⁰⁶⁵⁸¹⁰⁹⁷) ld.father ^{P22} = ?father) | ⁷²)@?Y ⁵⁵⁸¹⁰⁹⁷)@?X, r)@{type o | https:// | tools.w | vmfla | abs.org | | | quences |

Lo

This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

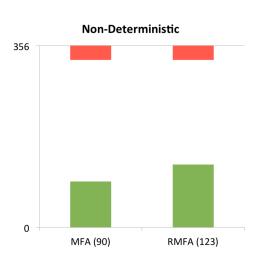
- * Experiments with non-DL existential rule sets
- Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets
- * Efficient implementation for acyclicity checks
- * Optimise chase reasoners
- * The disjunctive case



This is it, everybody should use existential rules + acyclicity notions!

Surely not, a lot of work still needs to be done:

- * Experiments with non-DL existential rule sets
- * Develop tools that can assist knowledge engineers in "repairing" non-acyclic rule sets
- * Efficient implementation for acyclicity checks
- * Optimise chase reasoners
- * The disjunctive case



| Edit Course Defector Maria | | kus/Documents/demo-rules.o | dpg - Rule IDE | | | | | |
|--|--|---|--|-------------------------|---------|--|-------------------|------------|
| | ate Search Project Run Window Helj | | | 5 × 45 6 - | v 🔿 v I | | Quick Access | |
| | I drone-control.dpg I demo-rul | | | | | ystem 🛙 | | |
| Aule IDE example Project Trace1 [1.3M facts] Trace1 [1.3M facts] Constraints.dpg Constraints.dpg log-2018-09-03 [2.1K facts] dog-2018-09-04 [1.7K facts] | <pre>ePREFIX tr: <htp: %="" <htp:="" a="" building="" constructing="" double="" eprefix="" level="" r0(1x,.o(1y)="" r0(s0).r0(s1).succ="" tr:=""> a 00(1x,.o(1y) > a 0(1x,.o(1y) ></htp:></pre> | (12x, 1y; 13) . 2) . (1, 72m) > succl(12x, 77 (12y) . (12y) . (12y) . 2) from level 1: (12x, 1y, 12) . 3) . 1) . (12y) . 2) from level 1: (12x, 1y, 12) . 3) . 1) . 1) . 2) from level 1: (12x, 1y, 12) . 3) . 1) . 2) . 2 | <pre>ty-example/> er [Cali, Gott: n) . n) .) -> succl(?s, inl(?yn), succ: e</pre> | /mm). (7x,7xm) -> mm | De | source Monitor B% rivation steps | | |
| | | Writable | Smart Insert 2 | 5:12 | | | | |
| sq | ID | Search I | tem | ٩ | Start | Properties | Classes Rule | es Abou |
| Rule | s list | | | | | | | |
| spouse is s | ymmetric | | | | | n | naterialisable co | onsequence |
| | ouse ^{P26} = ?y)@?S .spouse ^{P26} = ?x)@?S | | | | | | | |
| A male pare | ent is a father | | | | | n | naterialisable co | onsequence |
| | er.child ^{P40} = ?child)@?X, er.sex or gender ^{P21} = male ⁰⁶ | ⁵⁸¹⁰⁹⁷)@?Y | | | | | | |
| | hild.father ^{P22} = ?father) | | | | | | | |
| → (?c | hild.father ^{P22} = ?father) arent is a mother | | https://t | | flah | oral | aid | |
| → (?c A female pa (?moth) (?mothe | | | https://t | ools.wn | nflabs | s.org/s | sqid/ | |
| → (?c A female pr (?moth (?moth → (?c | arentis a mother er.child ^{P40} = ?child)@?X, er.sex or gender ^{P21} = female | | https://t | ools.wn | nflab | s.org/s | sqid/ | |
| → (?c A female par (?mothe → (?c a male par (?granu (?granu (?pare) | arent is a mother er.child ^{P40} = ?child)@?X, er.sex or gender ^{P21} = female hild.mother ^{P25} = ?mother) | 2 ⁰⁶⁵⁸¹⁰⁷²)@?Y male ^{Q6581097})@?X, g?Y, | https://t | pols.wn | nflab | s.org/s | sqid/ | |
| → (?c A female par (?moth (?moth → (?c a male par (?granu (?granu (?pare (?pare (?pare) (?c))))))))))))))))))))))))))))))))))) | arent is a mother er.child ^{P40} = ?child)@7X, er.sex or gender ^{P21} = female hild.mother ^{P25} = ?mother) ent of a parent is a grandfather dfather.sex or gender ^{P21} = 1 ffather.child ^{P40} = ?child)@72 | 2 ⁰⁶⁵⁸¹⁰⁷²)@?Y male ^{Q6581097})@?X, g?Y, | https://t | ools.wm | nflab | s.org/s | sqid/ | |

Lo

Bibliography

Reasoning over Existential Rules with Acyclicity Notions and the Datalog-first Restricted Chase

David Carral



Slides available at https://iccl.inf.tu-dresden.de/web/Existential-rules-acyclicity

Bibliography: Sections

* First section

Restricted Chase (Non)Termination for Existential Rules with Disjunctions [IJCAI 2017] <u>https://iccl.inf.tu-dresden.de/web/Inproceedings3140/en</u>

* Second section

Tractable Query Answering for Expressive Ontologies and Rules [ISWC 2017] <u>https://iccl.inf.tu-dresden.de/web/Inproceedings3163/en</u>

* Third section

Efficient Model Construction for Horn Logic with VLog [IJCAR 2018] <u>https://iccl.inf.tu-dresden.de/web/Article3046/en</u>

Bibliography: Rule Engines

* VLog

Efficient Model Construction for Horn Logic with VLog [IJCAR 2018]

https://iccl.inf.tu-dresden.de/web/Article3046/en

Column-Oriented Datalog Materialization for Large Knowledge Graphs [AAAI 2016]

http://korrekt.org/papers/Urbani-Jacobs-Kroetzsch_Vlog-datalog-materialization-AAAI2016.pdf

* **RDFox**

Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems [AAAI 2014] http://www.cs.ox.ac.uk/ian.horrocks/Publications/download/2014/MNPHO14a.pdf

Bibliography: Acyclicity Notions

* Restricted Model-Faithful Acyclicity (RMFA) Restricted Chase (Non)Termination for Existential Rules with Disjunctions. [IJCAI 2017] <u>https://iccl.inf.tu-dresden.de/web/Inproceedings3140/en</u>

* Model-Faithful Acyclicity (MFA)

Acyclicity Notions for Existential Rules and Their Application to QA in Ontologies [J. Artif. Intell. Res. 47] <u>https://iccl.inf.tu-dresden.de/web/Article4005/en</u>

* Joint Acyclicity (JA)

Extending Decidable Existential Rules by Joining Acyclicity and Guardedness [IJACI 2011] <u>https://iccl.inf.tu-dresden.de/web/Inproceedings3149/en</u>

* Weak Acyclicity (WA)

Data Exchange: Semantics and Query Answering [Theor. Comput. Sci. 336] https://www.sciencedirect.com/science/article/pii/S030439750400725X