An Introduction to Reasoning with Existential Rules

David Carral
LIRMM, Inria, University of Montpellier, CNRS

November 17, 2021

About to the Seminar

Program

1. Morning Session from 9:30 to 12:00 (more general/introductory):
   - "Introduction to ∃-Rules" by D. Carral
   - "Reasoning with Guarded ∃-Rules" by M. Thomazo
   - Coffee Break (outside of the seminar room)
   - "Compressing Rule-based Reasoning" by J. Urbani

2. Lunch Break from 12:00 to 13:30 in Building 5 (did you register?)

3. Afternoon Session from 13:30 to 17:00 (more specific/technical):
   - "Capturing Homomorphism-Closed Decidable Queries with ∃-Rules" by S. Rudolph (winner of the Ray Reiter Best Paper Prize 2021)
   - "Answering Counting Queries over Lightweight Ontologies" by M. Thomazo
   - Coffee Break (outside of the seminar room)
   - "Derivation Graphs, Greediness, and Bounded-treewidth in the Context of ∃Rules" by T. Lyon
   - "∃-rules: Intersecting FO-Rewritability and Core Termination" by P. Ostropolski-Nalewaja

Program

1. Morning Session from 9:30 to 12:00
2. Lunch Break from 12:00 to 13:30
3. Afternoon Session from 13:30 to 17:00

Program

1. Morning Session from 9:30 to 12:00
2. Lunch Break from 12:00 to 13:30
3. Afternoon Session from 13:30 to 17:00

Remark

Complete schedule available at:
`https://www-sop.inria.fr/members/David.Carral/events/2021-montpellier-existential-rules-seminar.html`

I will also upload the slides by the end of the week!

Google “David Carral Sophia Inria” to find my personal webpage, then look under the “Events” tab.

Acknowledgements

Organisators: Virginie Fèche, Mégane Miquel, and Christian Retoré

Acknowledgements

Organisators: Virginie Fèche, Mégane Miquel, and Christian Retoré (also, Marie-Laure Mugnier and Federico Ulliana)

Acknowledgements

Organisators: Virginie Fèche, Mégane Miquel, and Christian Retoré (also, Marie-Laure Mugnier and Federico Ulliana)

Speakers: Timothy Stephen Lyon, Piotr Ostropolski-Nalewaja, Sebastian Rudolph, Michaël Thomazo x 2, and Jacopo Urbani

Acknowledgements

Organisators: Virginie Fèche, Mégane Miquel, and Christian Retoré (also, Marie-Laure Mugnier and Federico Ulliana)

Speakers: Timothy Stephen Lyon, Piotr Ostropolski-Nalewaja, Sebastian Rudolph, Michaël Thomazo x 2, and Jacopo Urbani

Funding provided by:

- The LIRMM through the “Action Transverse Logique”
- The GraphIK team
- The Chair of Computational Logic at TU Dresden (that is, Sebastian’s research group)

Intro to Existential Rules

Goals of this Presentation

What this talk is about:

- General intro to $\exists$-rules
- Provide context for the other talks
- Research goals in this field

Goals of this Presentation

What this talk is about:

- General intro to $\exists$-rules
- Provide context for the other talks
- Research goals in this field

What is not:

- Motivation for the research of $\exists$-rules
- Deep technical content

Goals of this Presentation

What this talk is about:

- General intro to $\exists$-rules
- Provide context for the other talks
- Research goals in this field

What is not:

- Motivation for the research of $\exists$-rules
- Deep technical content

Remark: Contact me for more info!

- Email: `david.carral@inria.fr`
- Office 3.129 at Building 5

Syntax: Existential Rules

Definition: Existential Rules

An *(existential) rule* is a first-order logic formula $\forall \vec{x}, \vec{z}.\beta[\vec{x}, \vec{z}] \rightarrow \exists \vec{y}.\eta[\vec{x}, \vec{y}]$ where β and η are conjunctions of atoms without function symbols.

In a nutshell: $\exists$-rules = Datalog rules + $\exists$-quantifiers in the head

Syntax: Existential Rules

Definition: Existential Rules

An *(existential) rule* is a first-order logic formula $\forall \vec{x}, \vec{z}.\beta[\vec{x}, \vec{z}] \rightarrow \exists \vec{y}.\eta[\vec{x}, \vec{y}]$ where β and η are conjunctions of atoms without function symbols.

In a nutshell: $\exists$-rules = Datalog rules + $\exists$-quantifiers in the head

Example: Existential Rules

$$\forall x.\mathit{PhDStudent}(x) \rightarrow \exists y.\mathit{MainSupervisor}(x, y) \wedge \mathit{Professor}(y)$$
$$\forall x, y, z.\mathit{MainSupervisor}(x, y) \wedge \mathit{EmployedAt}(y, z) \rightarrow \mathit{StudiesAt}(x, z)$$
$$\forall x, y, z.\mathit{MainSupervisor}(x, y) \wedge \mathit{MainSupervisor}(x, z) \rightarrow y \approx z$$
$$\forall x.\mathit{Professor}(x) \wedge \mathit{PhDStudent}(x) \rightarrow \bot$$

Syntax: Existential Rules

Definition: Existential Rules

An *(existential) rule* is a first-order logic formula $\forall \vec{x}, \vec{z}.\beta[\vec{x}, \vec{z}] \rightarrow \exists \vec{y}.\eta[\vec{x}, \vec{y}]$ where β and η are conjunctions of atoms without function symbols.

In a nutshell: $\exists$-rules = Datalog rules + $\exists$-quantifiers in the head

Example: Existential Rules

$$\begin{aligned} \mathit{PhDStudent}(x) &\rightarrow \exists y.\mathit{MainSupervisor}(x, y) \wedge \mathit{Professor}(y) \\ \mathit{MainSupervisor}(x, y) \wedge \mathit{EmployedAt}(y, z) &\rightarrow \mathit{StudiesAt}(x, z) \\ \mathit{MainSupervisor}(x, y) \wedge \mathit{MainSupervisor}(x, z) &\rightarrow y \approx z \\ \mathit{Professor}(x) \wedge \mathit{PhDStudent}(x) &\rightarrow \bot \end{aligned}$$

We often omit universal quantifiers

Syntax: Existential Rules

Definition: Existential Rules

An *(existential) rule* is a first-order logic formula $\forall \vec{x}, \vec{z}.\beta[\vec{x}, \vec{z}] \rightarrow \exists \vec{y}.\eta[\vec{x}, \vec{y}]$ where β and η are conjunctions of atoms without function symbols.

In a nutshell: $\exists$-rules = Datalog rules + $\exists$-quantifiers in the head

Example: Existential Rules

$$\begin{aligned} \mathit{PhDStudent}(x) &\rightarrow \exists y.\mathit{MainSupervisor}(x, y) \wedge \mathit{Professor}(y) \\ \mathit{MainSupervisor}(x, y) \wedge \mathit{EmployedAt}(y, z) &\rightarrow \mathit{StudiesAt}(x, z) \\ \mathit{MainSupervisor}(x, y) \wedge \mathit{MainSupervisor}(x, z) &\rightarrow y \approx z \\ \mathit{Professor}(x) \wedge \mathit{PhDStudent}(x) &\rightarrow \bot \end{aligned}$$

We often omit universal quantifiers

Remark: Database Theory

Diff. notation: tuple/equality generating dependencies and constraints.

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.
- A *Boolean conjunctive query (BCQ)* is a (closed) formula $\exists \vec{y}.\beta[\vec{y}]$ where β is a conjunction of atoms without function symbols.

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.
- A *Boolean conjunctive query (BCQ)* is a (closed) formula $\exists \vec{y}.\beta[\vec{y}]$ where β is a conjunction of atoms without function symbols.
- A *knowledge base (KB)* is a tuple $\langle \mathcal{R}, \mathcal{F} \rangle$ where $\mathcal{R}$ is a rule set and $\mathcal{F}$ is a fact set.

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.
- A *Boolean conjunctive query (BCQ)* is a (closed) formula $\exists \vec{y}.\beta[\vec{y}]$ where β is a conjunction of atoms without function symbols.
- A *knowledge base (KB)* is a tuple $\langle \mathcal{R}, \mathcal{F} \rangle$ where $\mathcal{R}$ is a rule set and $\mathcal{F}$ is a fact set.

All of the above: first-order logic formulas without function symbols!

Syntax and Semantics

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.
- A *Boolean conjunctive query (BCQ)* is a (closed) formula $\exists \vec{y}.\beta[\vec{y}]$ where β is a conjunction of atoms without function symbols.
- A *knowledge base (KB)* is a tuple $\langle \mathcal{R}, \mathcal{F} \rangle$ where $\mathcal{R}$ is a rule set and $\mathcal{F}$ is a fact set.

All of the above: first-order logic formulas without function symbols!

Definition: BCQ Entailment

A KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ *entails* a BCQ γ, written $\mathcal{K} \models \gamma$, if γ is a logical consequence of $\mathcal{R} \cup \mathcal{F}$ under standard first-order logic semantics.

Syntax and Semantics

Definition

- A *fact* is an atomic formula $P(\vec{c})$ where $\vec{c}$ are constants.
- A *Boolean conjunctive query (BCQ)* is a (closed) formula $\exists \vec{y}.\beta[\vec{y}]$ where β is a conjunction of atoms without function symbols.
- A *knowledge base (KB)* is a tuple $\langle \mathcal{R}, \mathcal{F} \rangle$ where $\mathcal{R}$ is a rule set and $\mathcal{F}$ is a fact set.

All of the above: first-order logic formulas without function symbols!

Definition: BCQ Entailment

A KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ *entails* a BCQ γ, written $\mathcal{K} \models \gamma$, if γ is a logical consequence of $\mathcal{R} \cup \mathcal{F}$ under standard first-order logic semantics.

Remark

Can we decide BCQ entailment?

Solving BCQ Entailment: Universal Models

Determine if $\mathcal{K}$ entails γ: determine if infinitely many models (which may be infinite!) satisfy γ

Solving BCQ Entailment: Universal Models

Determine if $\mathcal{K}$ entails γ: determine if infinitely many models (which may be infinite!) satisfy γ

$\rightsquigarrow$ 2 dimensions of infinity!

Solving BCQ Entailment: Universal Models

Determine if $\mathcal{K}$ entails γ: determine if infinitely many models (which may be infinite!) satisfy γ

$\rightsquigarrow$ 2 dimensions of infinity!

Definition

A *universal model* of a KB $\mathcal{K}$ is a model that can be homomorphically embedded into every other model of $\mathcal{K}$.

Solving BCQ Entailment: Universal Models

Determine if $\mathcal{K}$ entails γ: determine if infinitely many models (which may be infinite!) satisfy γ

$\rightsquigarrow$ 2 dimensions of infinity!

Definition

A *universal model* of a KB $\mathcal{K}$ is a model that can be homomorphically embedded into every other model of $\mathcal{K}$.

Theorem

A KB $\mathcal{K}$ entails a BCQ iff it is satisfied by some universal model of $\mathcal{K}$.

Solving BCQ Entailment: Universal Models

Determine if $\mathcal{K}$ entails γ: determine if infinitely many models (which may be infinite!) satisfy γ

$\rightsquigarrow$ 2 dimensions of infinity!

Definition

A *universal model* of a KB $\mathcal{K}$ is a model that can be homomorphically embedded into every other model of $\mathcal{K}$.

Theorem

A KB $\mathcal{K}$ entails a BCQ iff it is satisfied by some universal model of $\mathcal{K}$.

Proof Intuition: universal models are the *logically weakest* models

Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x) \text{ and } q' = \exists x,y,z.R(x,y) \wedge R(y,z);$$

and the following models of $\mathcal{K}$:

$$\mathcal{M} = \{R(a,b), R(b,u), R(u,v), R(v,b)\}$$
$$\mathcal{U} = \{R(a,b), R(b,u_1), R(u_1,u_2), R(u_2,u_3), \ldots\}$$

Example

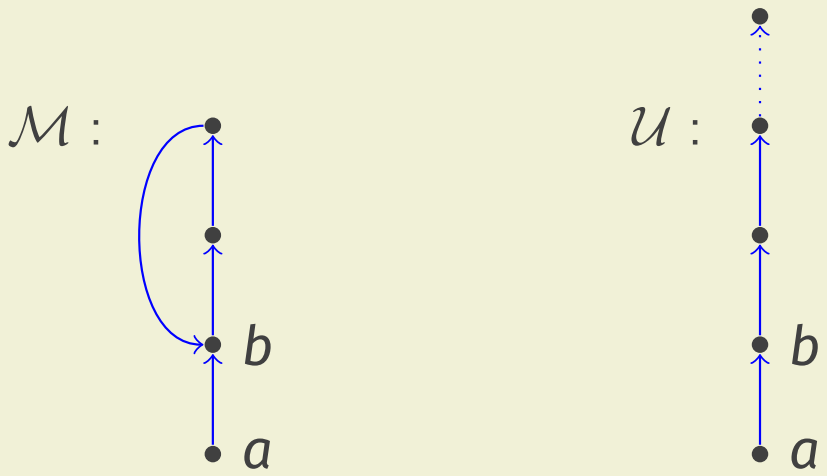
Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x) \text{ and } q' = \exists x,y,z.R(x,y) \wedge R(y,z);$$

and the following models of $\mathcal{K}$:

$$\mathcal{M} = \{R(a,b), R(b,u), R(u,v), R(v,b)\}$$
$$\mathcal{U} = \{R(a,b), R(b,u_1), R(u_1,u_2), R(u_2,u_3), \ldots\}$$

Graphical representation:



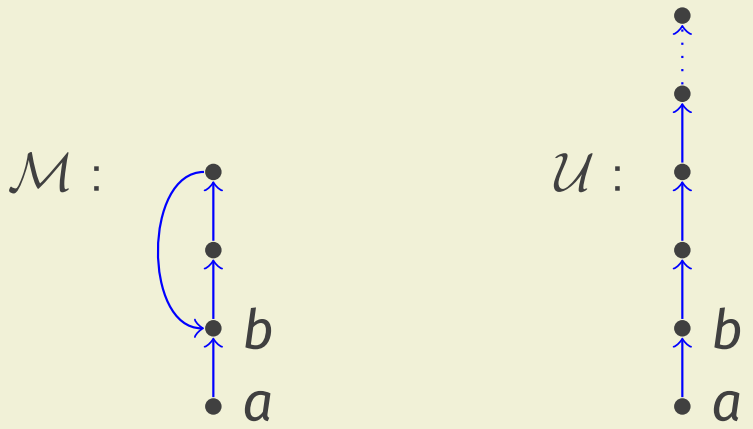
Solving BCQ Entailment with Universal Models

Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$q = \exists x, y, z.R(x,y) \wedge R(y,z) \wedge R(z,x)$ and $q' = \exists x, y, z.R(x,y) \wedge R(y,z)$;

and the following models of $\mathcal{K}$:



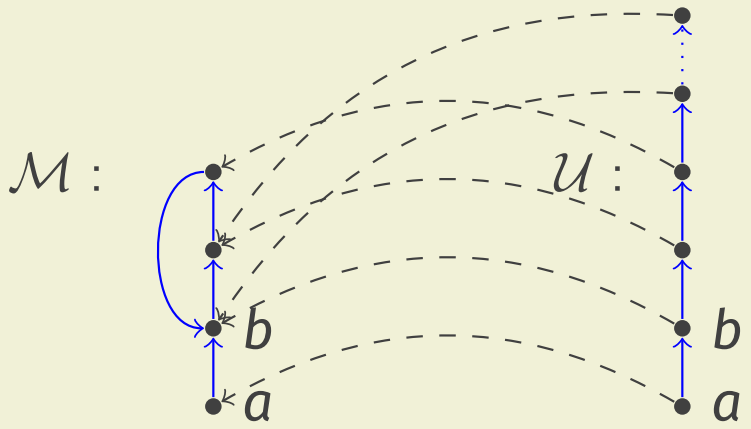
Solving BCQ Entailment with Universal Models

Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x)$ and $q' = \exists x,y,z.R(x,y) \wedge R(y,z)$;

and the following models of $\mathcal{K}$:



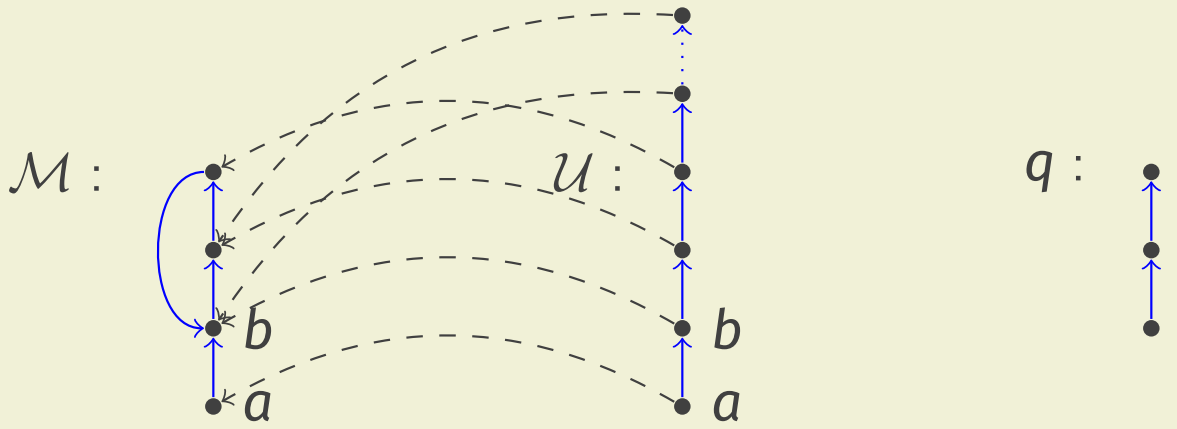
Solving BCQ Entailment with Universal Models

Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x)$ and $q' = \exists x,y,z.R(x,y) \wedge R(y,z)$;

and the following models of $\mathcal{K}$:

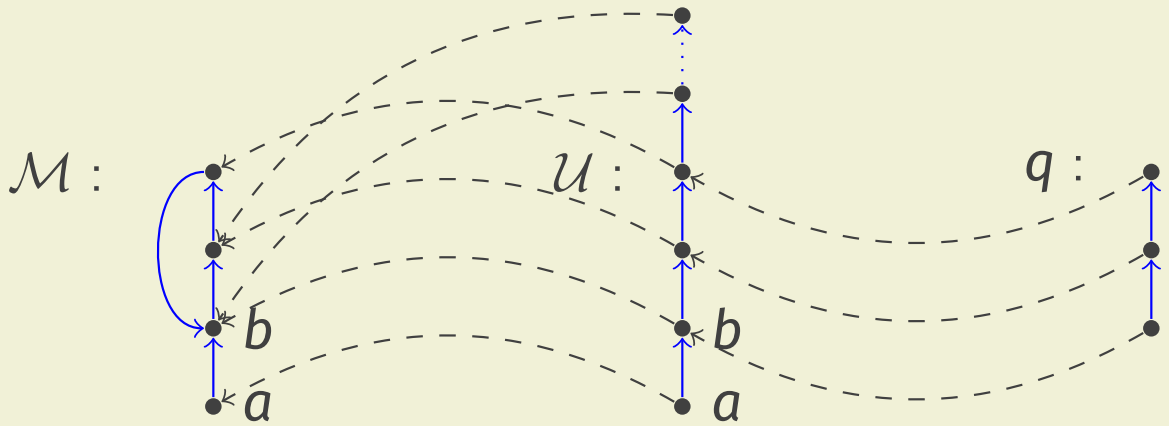


Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x) \text{ and } q' = \exists x,y,z.R(x,y) \wedge R(y,z);$$

and the following models of $\mathcal{K}$:

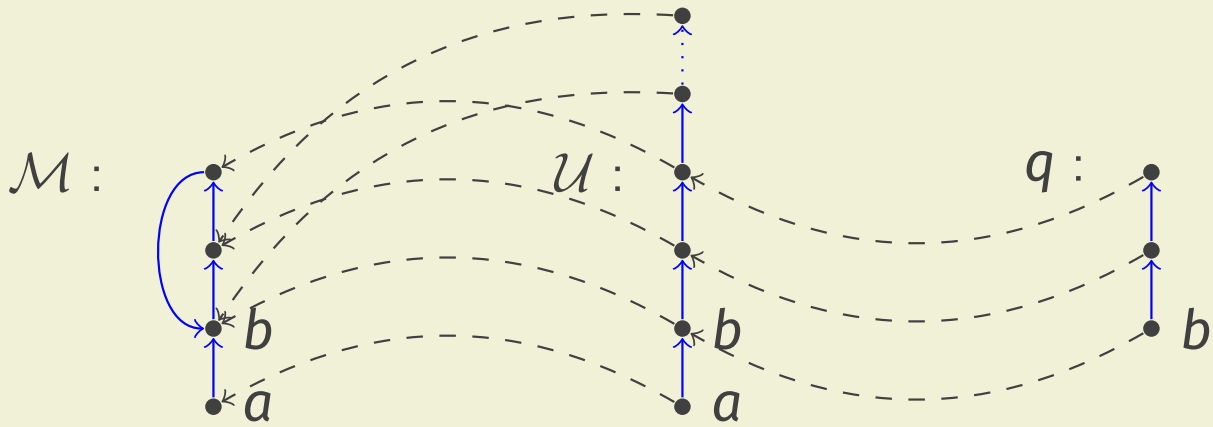


Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x) \text{ and } q' = \exists x,y,z.R(x,y) \wedge R(y,z);$$

and the following models of $\mathcal{K}$:



Theorem

A KB $\mathcal{K}$ entails a BCQ iff it is satisfied by some universal model of $\mathcal{K}$.

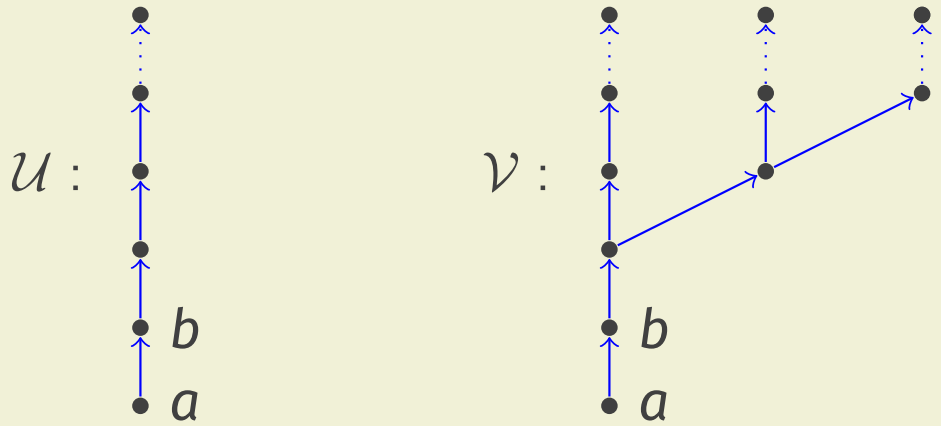
Solving BCQ Entailment with Universal Models

Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$q = \exists x, y, z.R(x,y) \wedge R(y,z) \wedge R(z,x)$ and $q' = \exists x, y, z.R(x,y) \wedge R(y,z)$;

and the following models of $\mathcal{K}$:

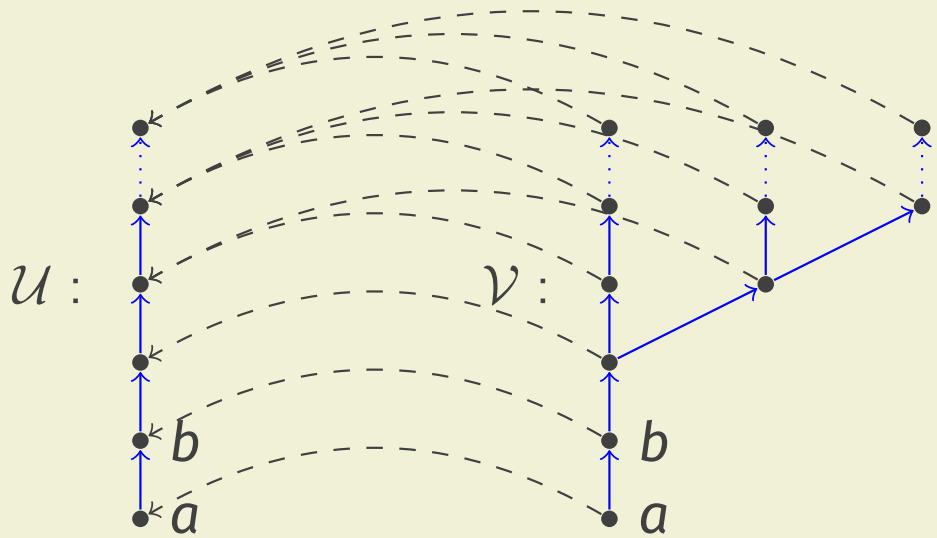


Example

Consider the KB $\mathcal{K} = \langle \{R(x,y) \rightarrow \exists z.R(y,z)\}, \{R(a,b)\} \rangle$; the BCQs

$$q = \exists x,y,z.R(x,y) \wedge R(y,z) \wedge R(z,x) \text{ and } q' = \exists x,y,z.R(x,y) \wedge R(y,z);$$

and the following models of $\mathcal{K}$:



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$

$\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

$\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$

Director(spielberg)

ActsIn(judeLaw, ai)

DirectedBy(ai, spielberg)

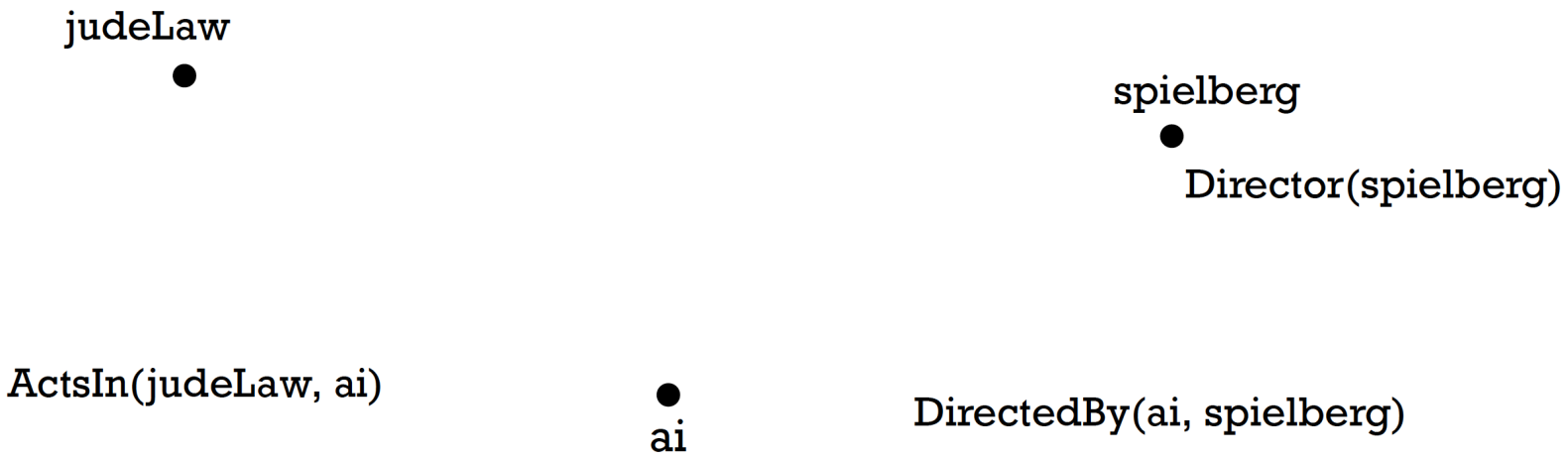
Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$

$\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

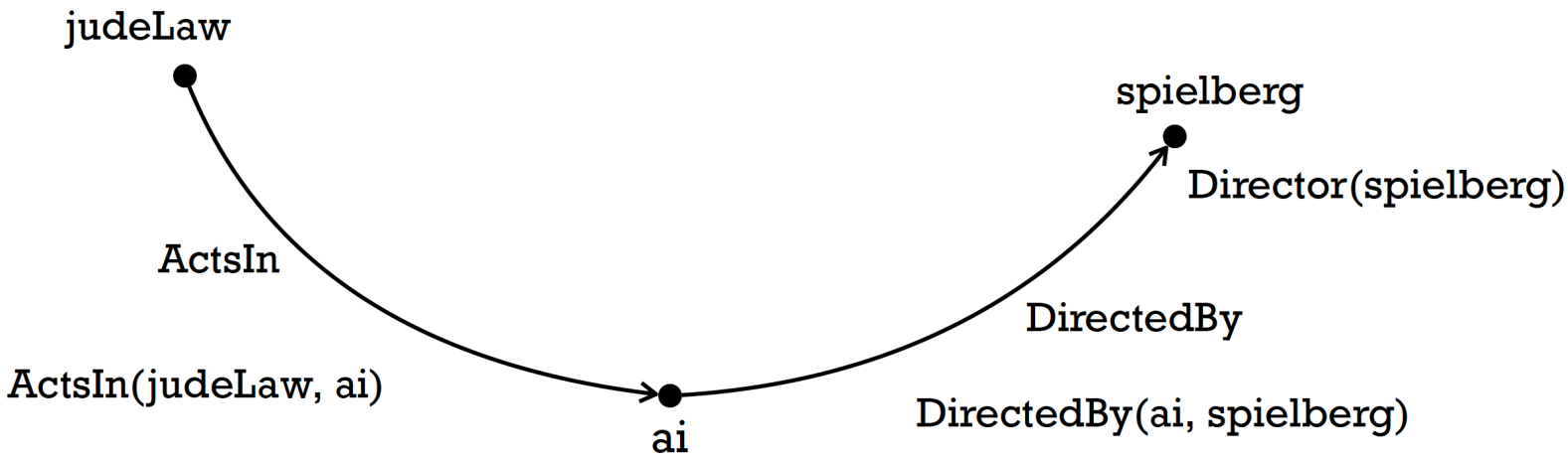
$\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$ $\quad$ $\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

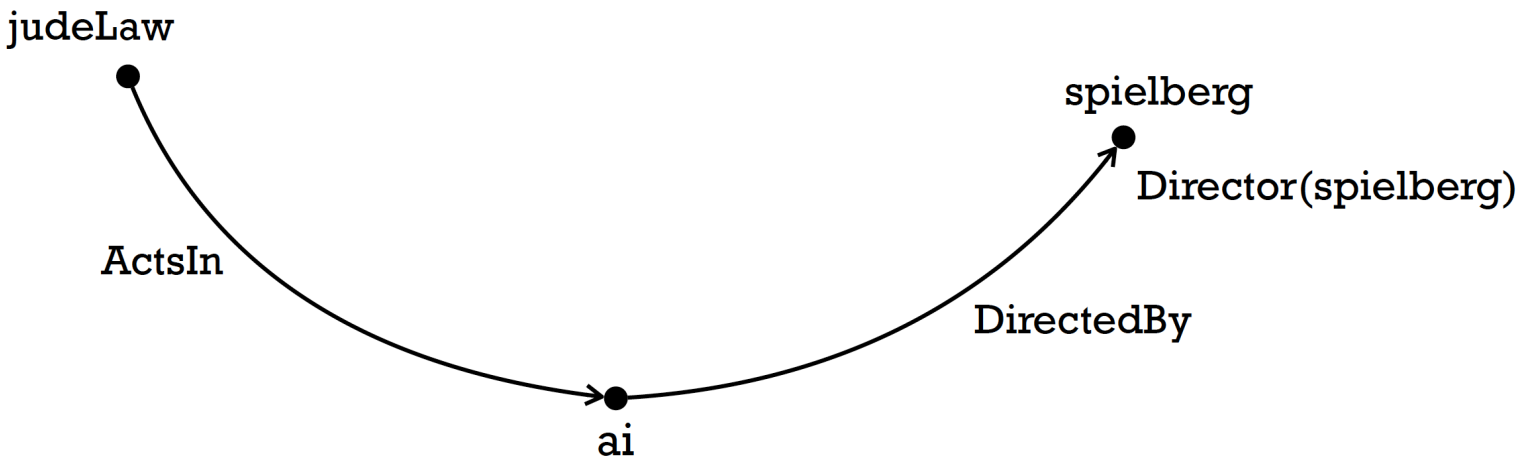
$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$ $\quad$ $\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$ $\quad$ $\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

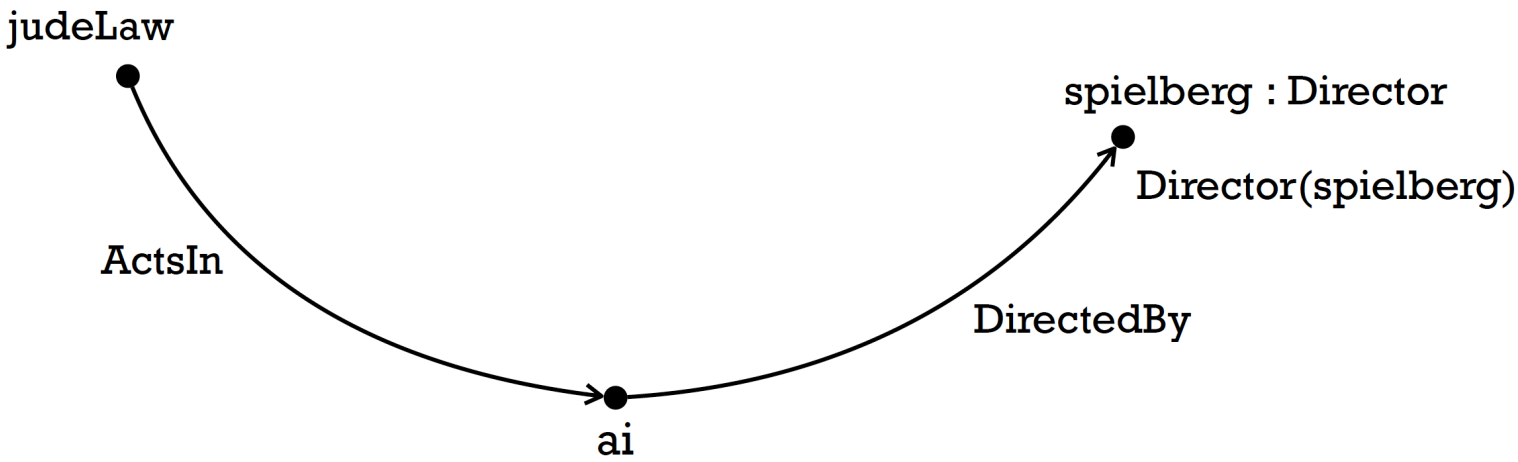
$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$ $\quad$ $\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$ $\quad$ $\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$ $\quad$ $\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



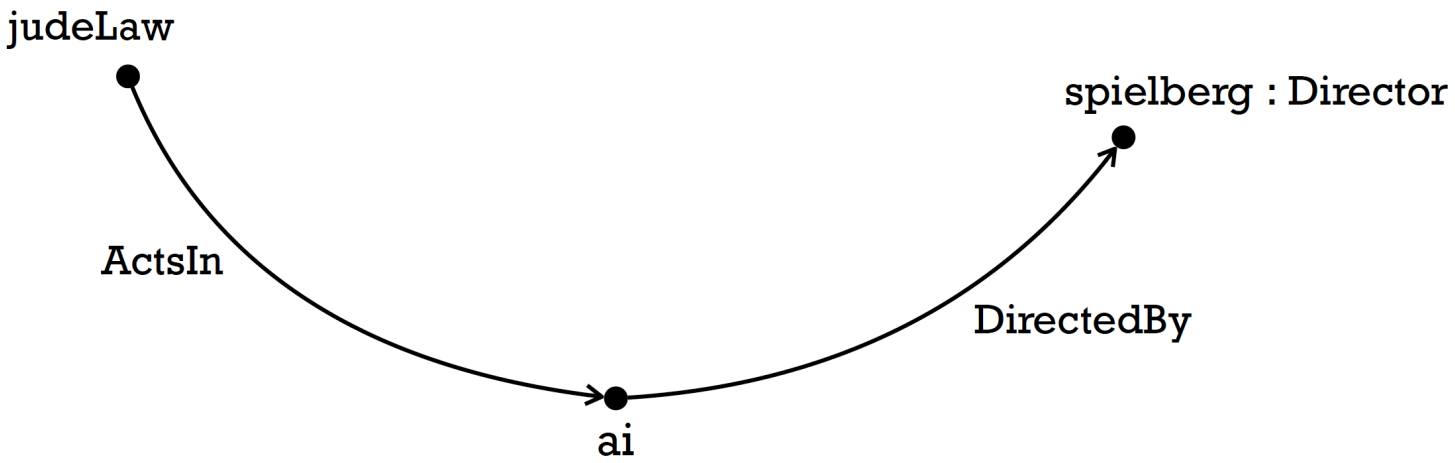
Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$

$\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

$\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



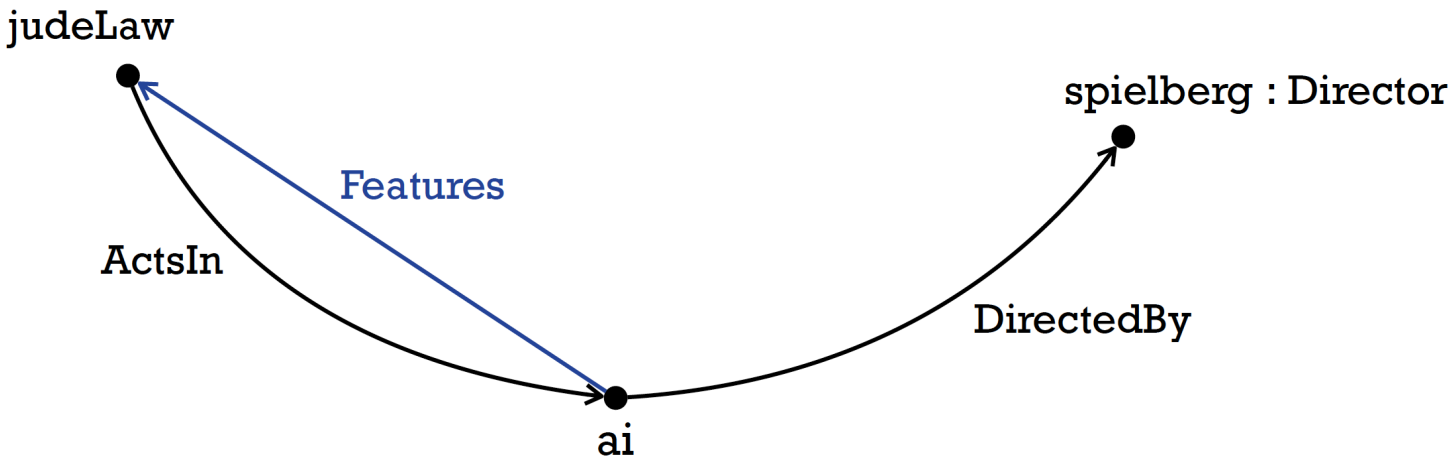
Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$

$\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

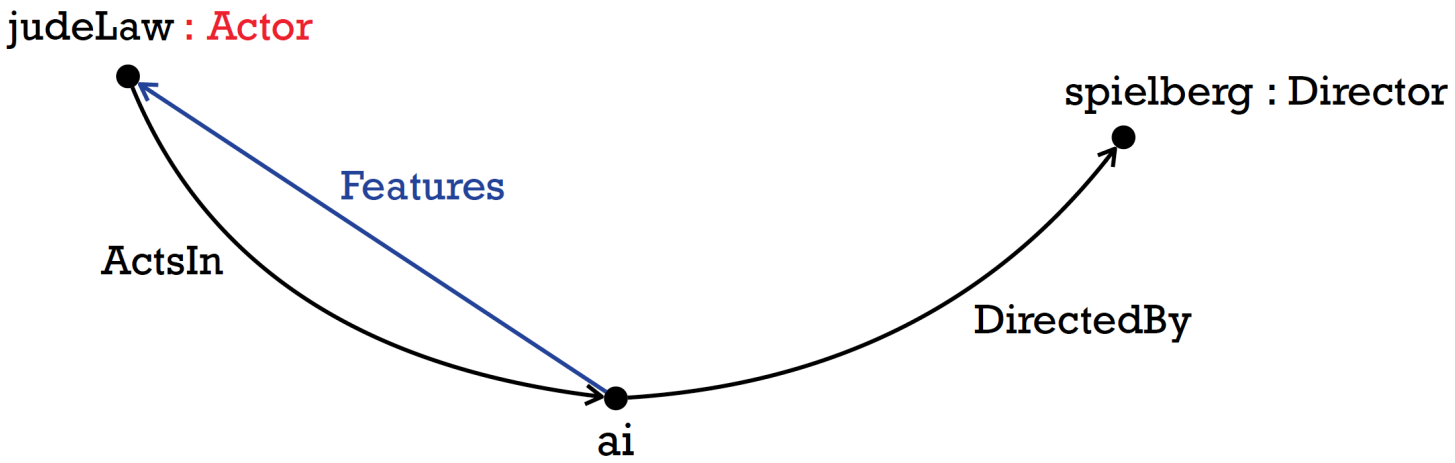
$\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$ $\quad$ $\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$ $\quad$ $\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



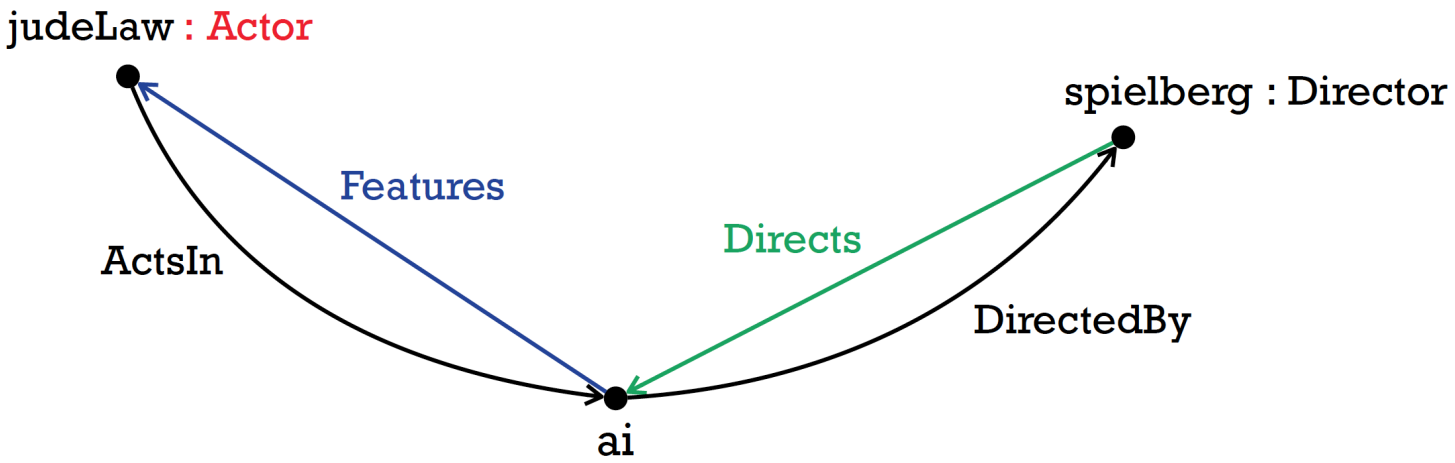
Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$

$\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

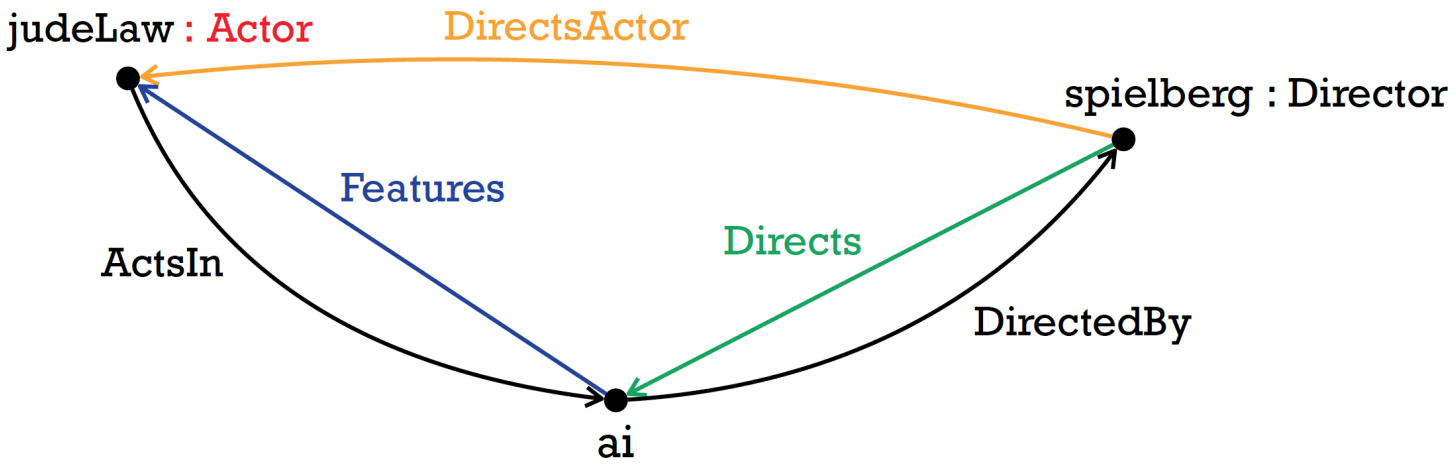
$\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Features}(x, y) \longrightarrow \text{Actor}(y)$ $\quad$ $\text{DirectedBy}(x, y) \longrightarrow \text{Directs}(y, x)$

$\text{ActsIn}(x, y) \longrightarrow \text{Features}(y, x)$ $\quad$ $\text{Directs}(x, y) \wedge \text{Features}(y, z) \longrightarrow \text{DirectsActor}(x, z)$



Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \exists v \,.\, \text{HasPart}(x, v) \wedge \text{Wheel}(v)$

$\text{Wheel}(x) \longrightarrow \exists w \,.\, \text{IsPartOf}(x, w) \wedge \text{Bicycle}(w)$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$

b : Bicycle

Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, f_v(x)) \wedge \text{Wheel}(f_v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, f_w(x)) \wedge \text{Bicycle}(f_w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$

Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$

b : Bicycle

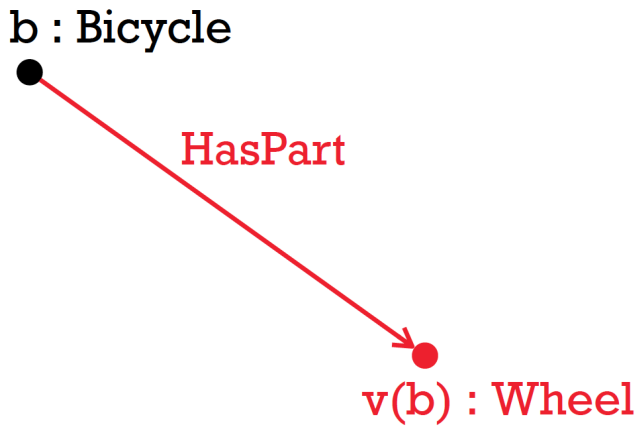
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



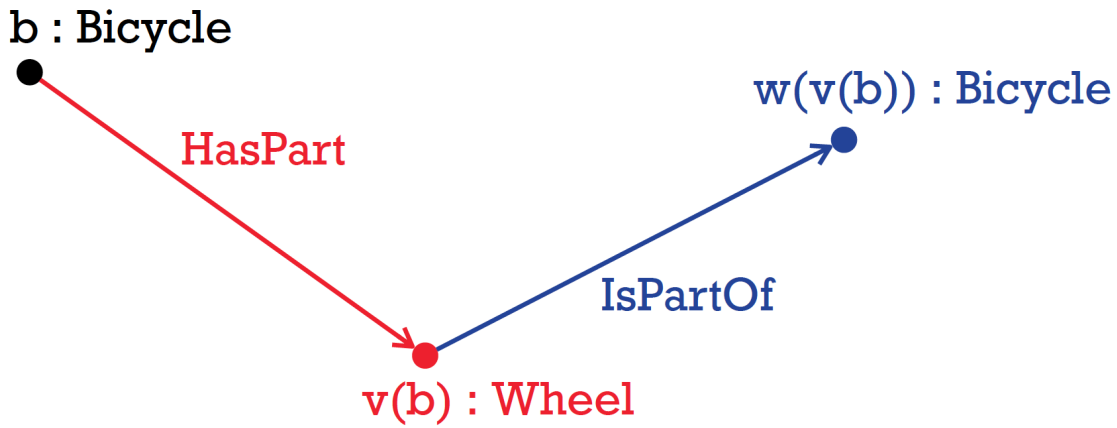
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



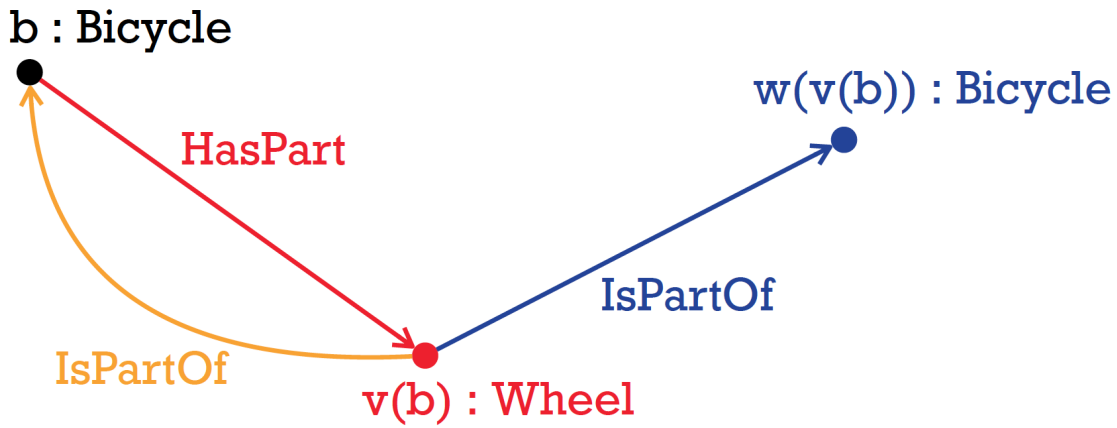
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



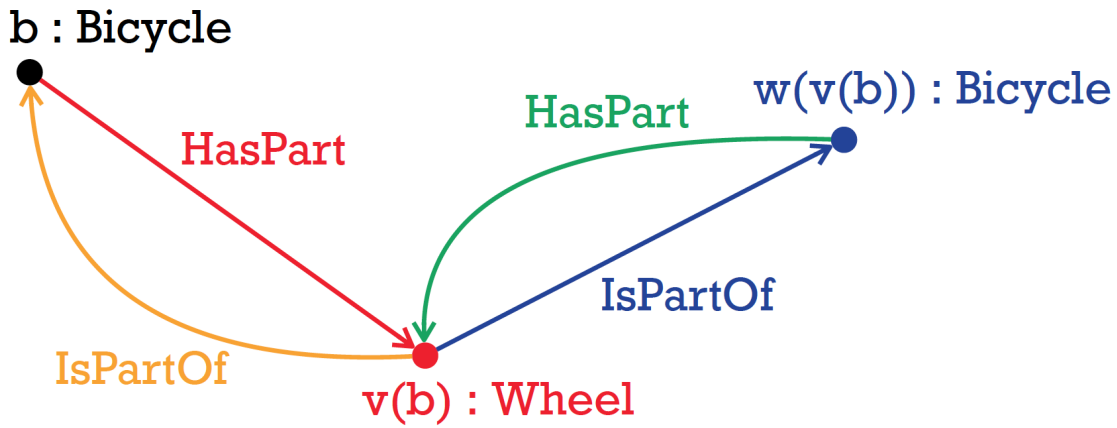
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



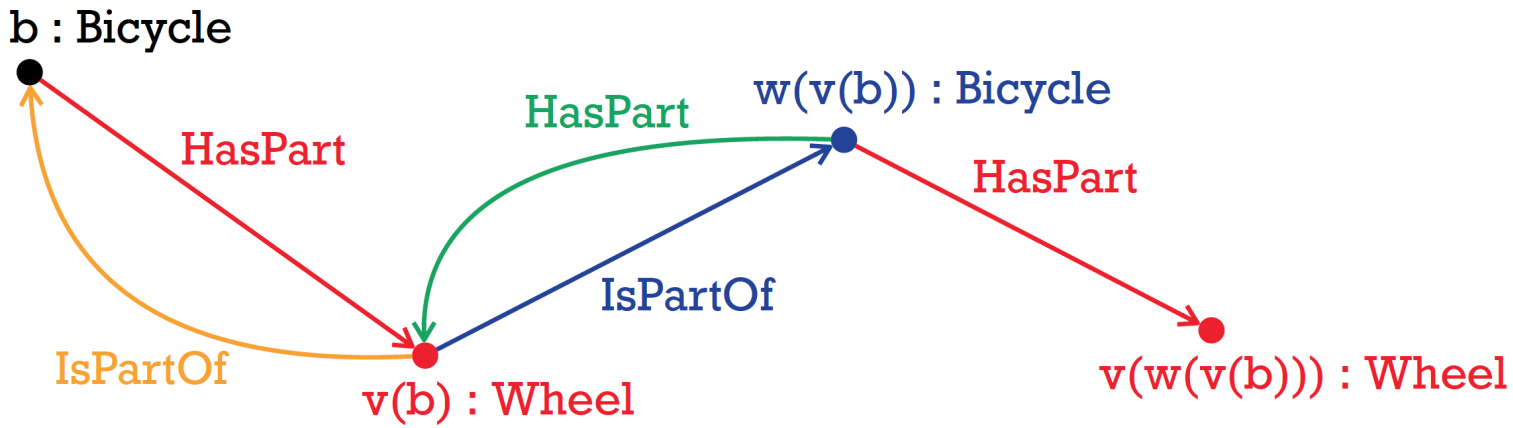
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



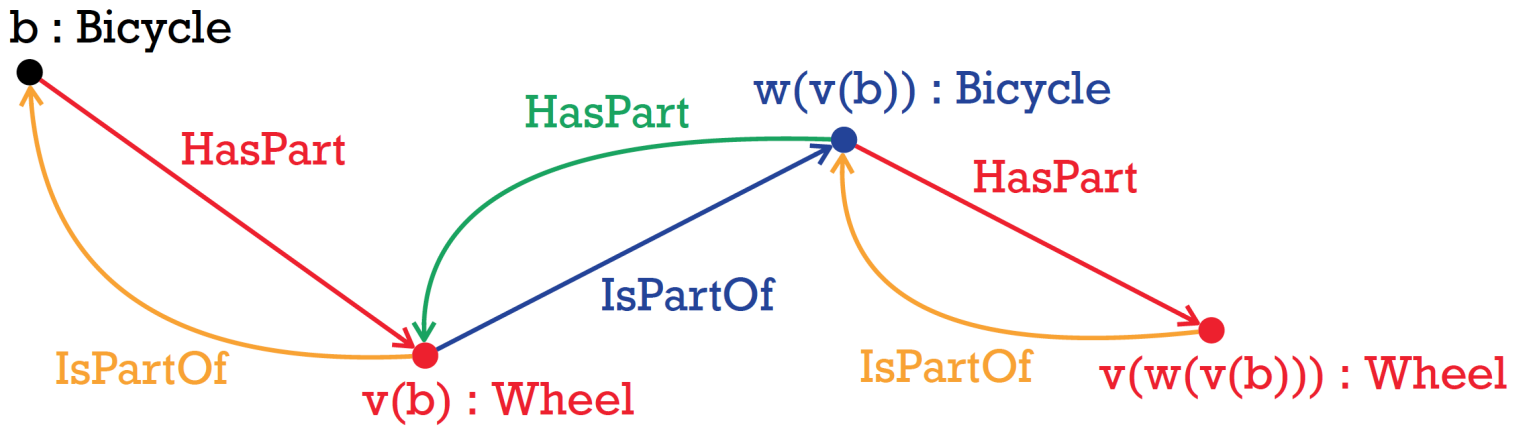
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



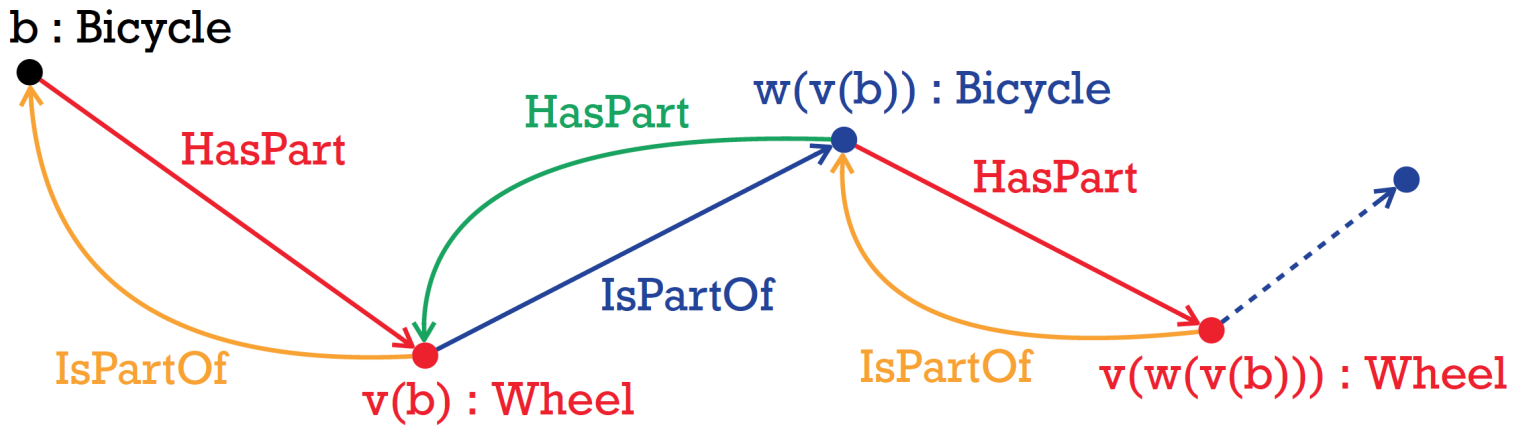
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



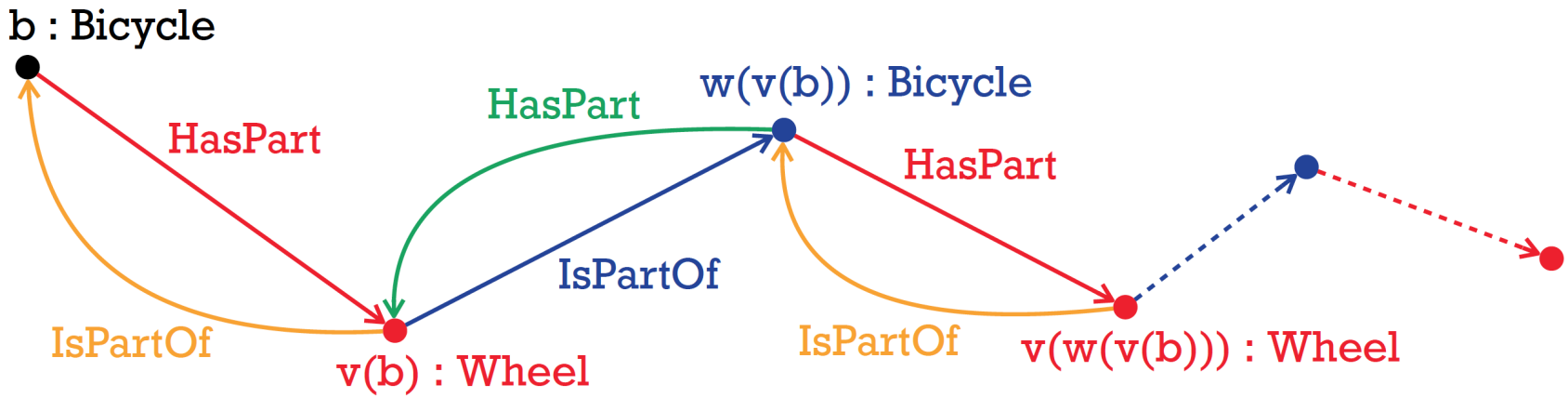
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



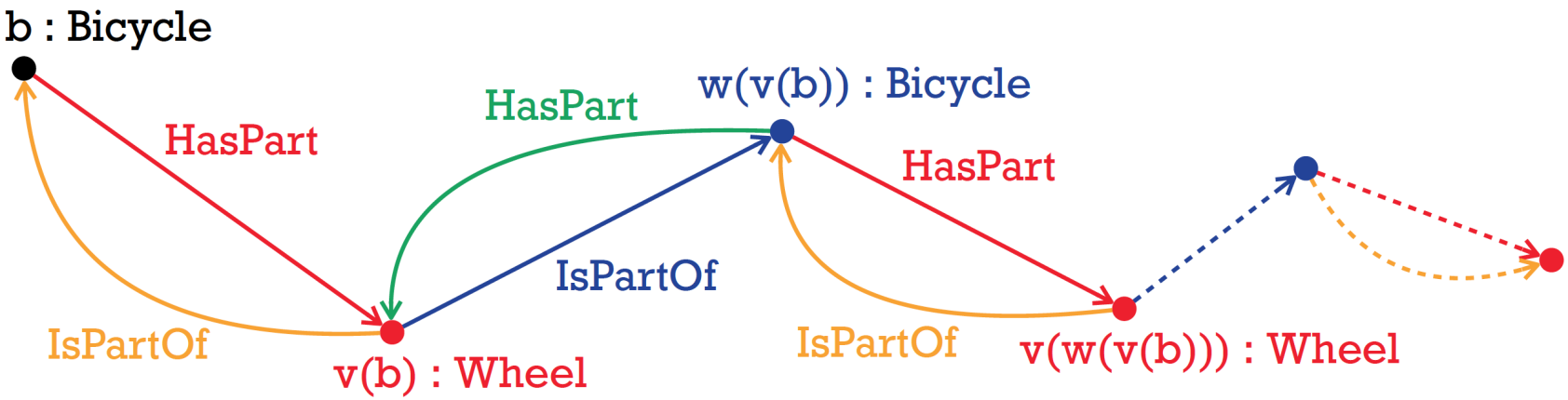
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



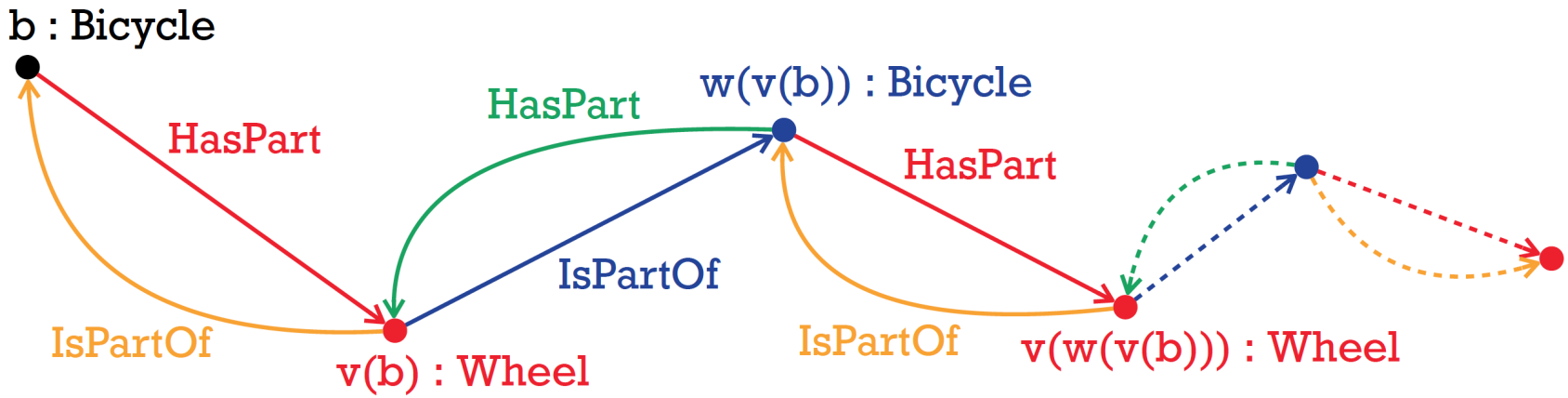
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



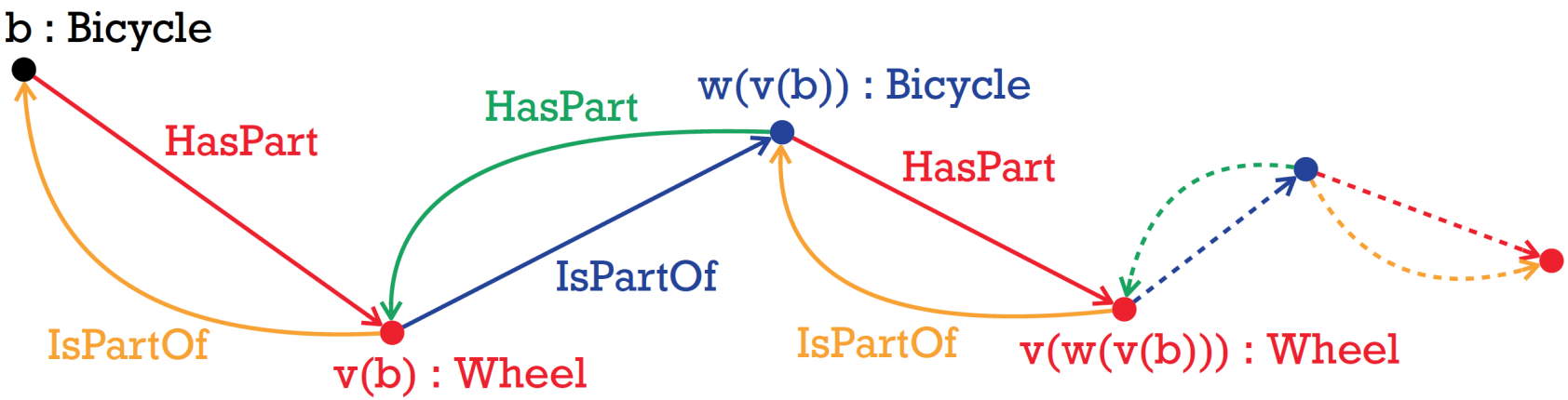
Universal Models: Skolem Chase

$\text{Bicycle}(x) \longrightarrow \text{HasPart}(x, v(x)) \wedge \text{Wheel}(v(x))$

$\text{Wheel}(x) \longrightarrow \text{IsPartOf}(x, w(x)) \wedge \text{Bicycle}(w(x))$

$\text{HasPart}(x, y) \longrightarrow \text{IsPartOf}(y, x)$

$\text{IsPartOf}(x, y) \longrightarrow \text{HasPart}(y, x)$



Seminar Talk

Jacopo will discuss an implementation of the chase algorithm that uses compression to improve performance.

Also, see [Mar09] for a formal definition!

Is BCQ Entailment Decidable?

Universal models and the chase: we only need one model!

Is BCQ Entailment Decidable?

Universal models and the chase: we only need one model!

$\rightsquigarrow$ However, the problem is still undecidable...

Is BCQ Entailment Decidable?

Universal models and the chase: we only need one model!

$\rightsquigarrow$ However, the problem is still undecidable...

Reduction from:

Theorem

Consider two context-free grammars G_1 and G_2. The problem of checking if $\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset$ is undecidable.

Is BCQ Entailment Decidable?

Universal models and the chase: we only need one model!
$\rightsquigarrow$ However, the problem is still undecidable...

Reduction from:

Theorem

Consider two context-free grammars G_1 and G_2. The problem of checking if $\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset$ is undecidable.

Definition

A *context-free grammar (CFG)* G consists of a start non-terminal S and a set of production rules of the form

$$A \to B_1 \cdot \ldots \cdot B_n \qquad A \to \epsilon$$

where A is a non-terminal and each B_i is a terminal or a non-terminal. Moreover, there is some production rule of the form $S \to B_1 \cdot \ldots \cdot B_n$ in G.

Wlog assumptions: all CFGs are defined over the binary alphabet $\{0, 1\}$; non-terminals do not reoccur across different grammars

BCQ Entailment is Undecidable (1)

How to encode a CFG with an existential rule KB:

Definition

For a CFG G, let $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$ wher $\mathcal{R}_G$ is the rule set that contains:

$$V(x) \to \exists y.R_0(x,y) \wedge V(y)$$
$$V(x) \to \exists y.R_1(x,y) \wedge V(y)$$
$$R_{B_1}(x_0,x_1) \wedge \ldots \wedge R_{B_n}(x_{n-1},x_n) \to R_A(x_0,x_n) \quad \text{for all } A \to B_1 \cdot \ldots \cdot B_n \in G$$
$$\to R_A(x,x) \quad \text{for all } A \to \epsilon \in G$$

In the above, each R_s is a fresh binary predicate unique for the non-terminal/terminal symbol s.

BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

$$S \to \mathrm{o}S\mathrm{1} \qquad V(x) \to \exists y.R_{\mathrm{o}}(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_{\mathrm{1}}(x,z) \wedge V(z)$$

$$S \to \epsilon \qquad R_{\mathrm{o}}(x,y) \wedge R_S(y,z) \wedge R_{\mathrm{1}}(z,w) \to R_S(x,w)$$

$$\to R_S(x,x)$$

Note that $\mathcal{L}(G) = \mathrm{o}^n\mathrm{1}^n$.

BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

$$S \to \mathrm{o}S\mathrm{1} \qquad V(x) \to \exists y.R_{\mathrm{o}}(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_{\mathrm{1}}(x,z) \wedge V(z)$$

$$S \to \epsilon \qquad R_{\mathrm{o}}(x,y) \wedge R_S(y,z) \wedge R_{\mathrm{1}}(z,w) \to R_S(x,w)$$

$$\to R_S(x,x)$$

Note that $\mathcal{L}(G) = \mathrm{o}^n\mathrm{1}^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:

BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

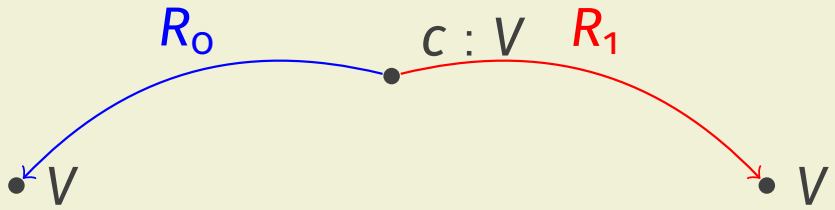
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

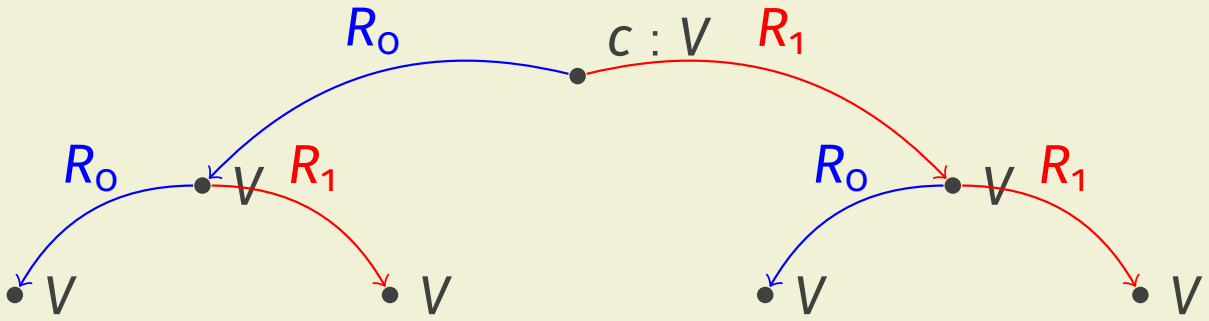
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

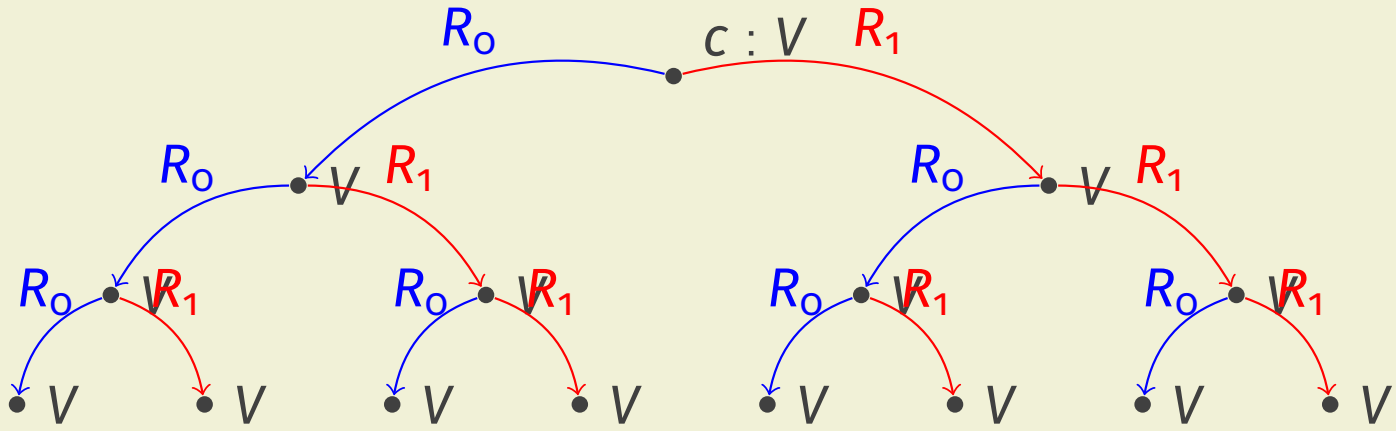
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

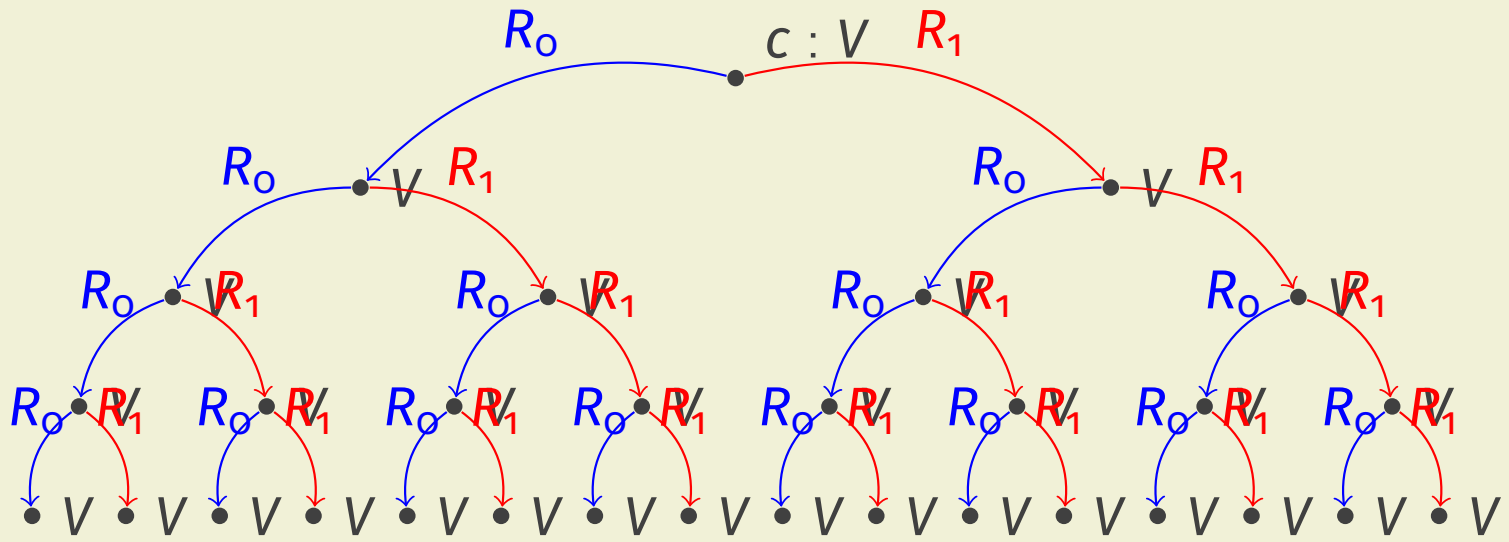
$$S \to \mathrm{o}S\mathrm{1} \qquad V(x) \to \exists y.R_{\mathrm{o}}(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_{\mathrm{1}}(x,z) \wedge V(z)$$

$$S \to \epsilon \qquad R_{\mathrm{o}}(x,y) \wedge R_S(y,z) \wedge R_{\mathrm{1}}(z,w) \to R_S(x,w)$$

$$\to R_S(x,x)$$

Note that $\mathcal{L}(G) = \mathrm{o}^n\mathrm{1}^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

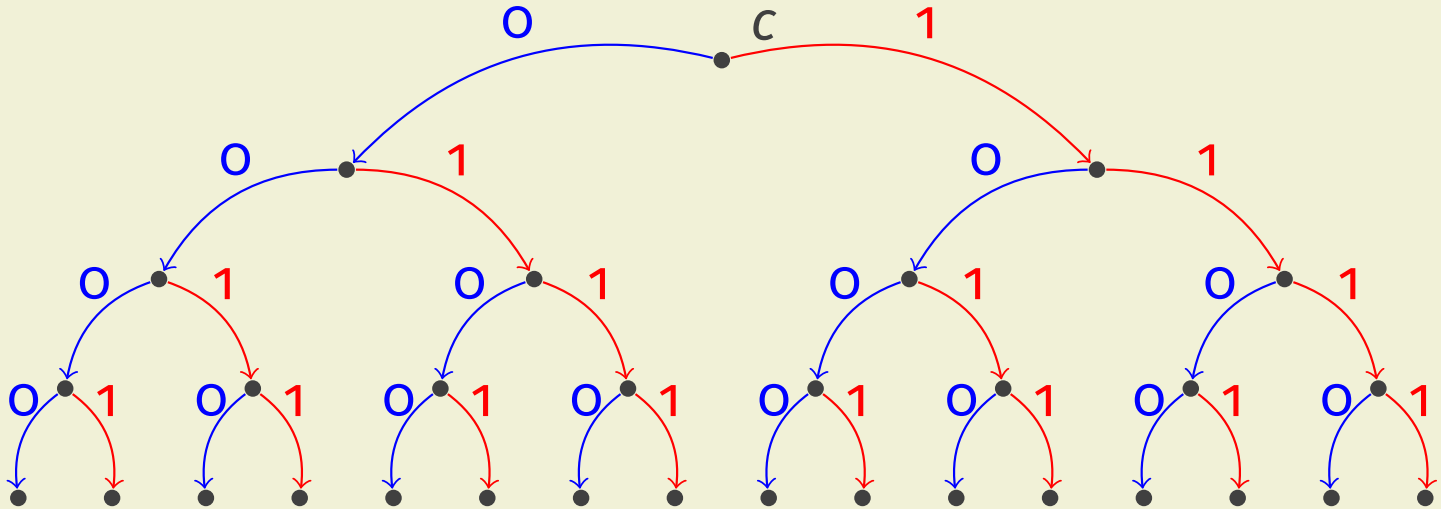
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

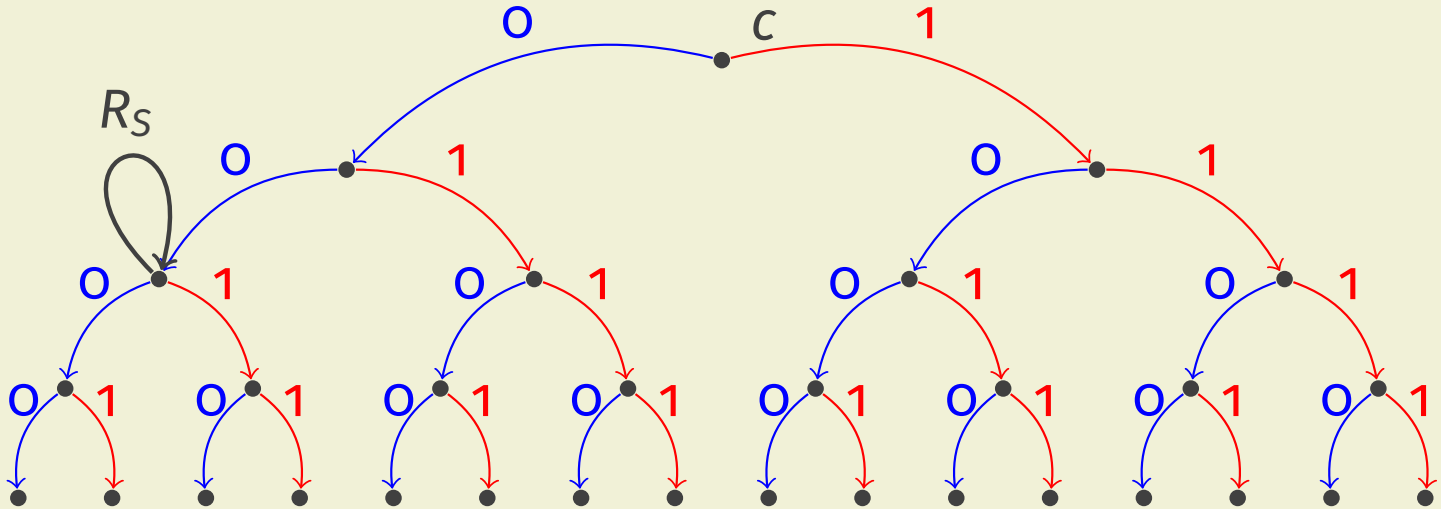
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

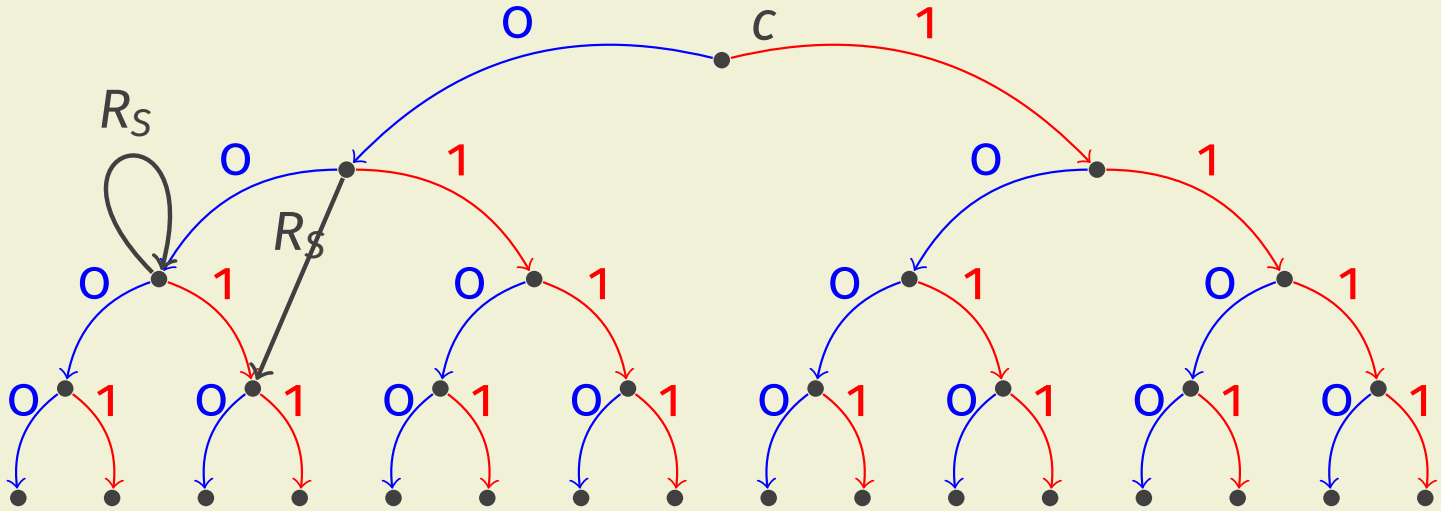
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

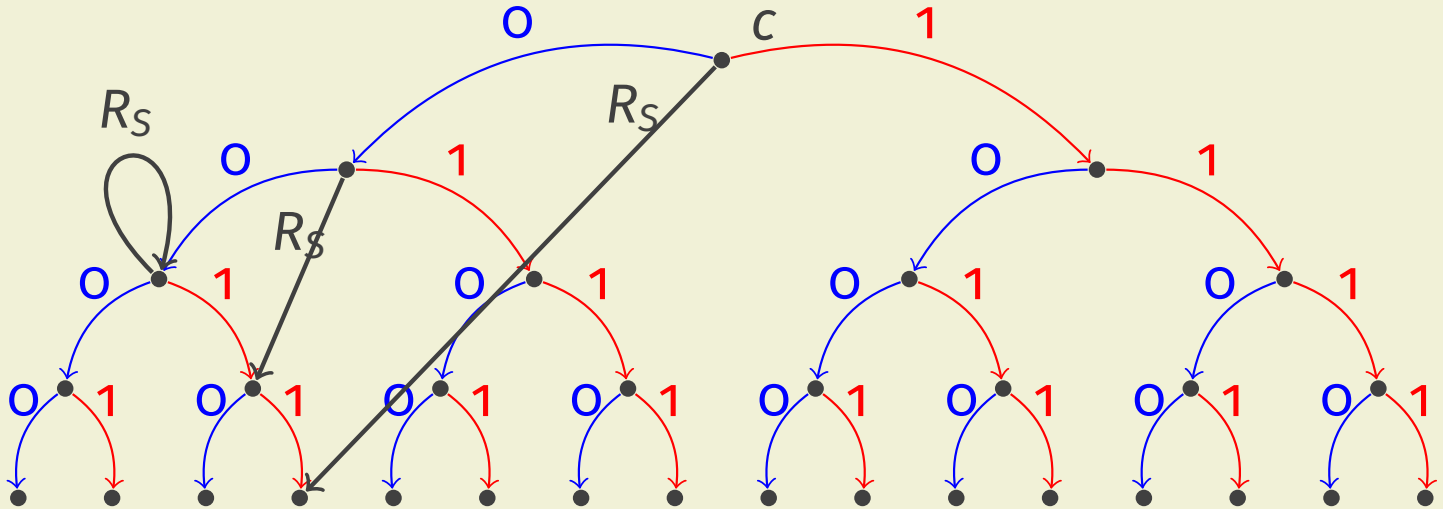
$$V(x) \to \exists y.R_0(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_1(x,z) \wedge V(z)$$

$$S \to 0S1 \qquad R_0(x,y) \wedge R_S(y,z) \wedge R_1(z,w) \to R_S(x,w)$$

$$S \to \epsilon \qquad \to R_S(x,x)$$

Note that $\mathcal{L}(G) = 0^n 1^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

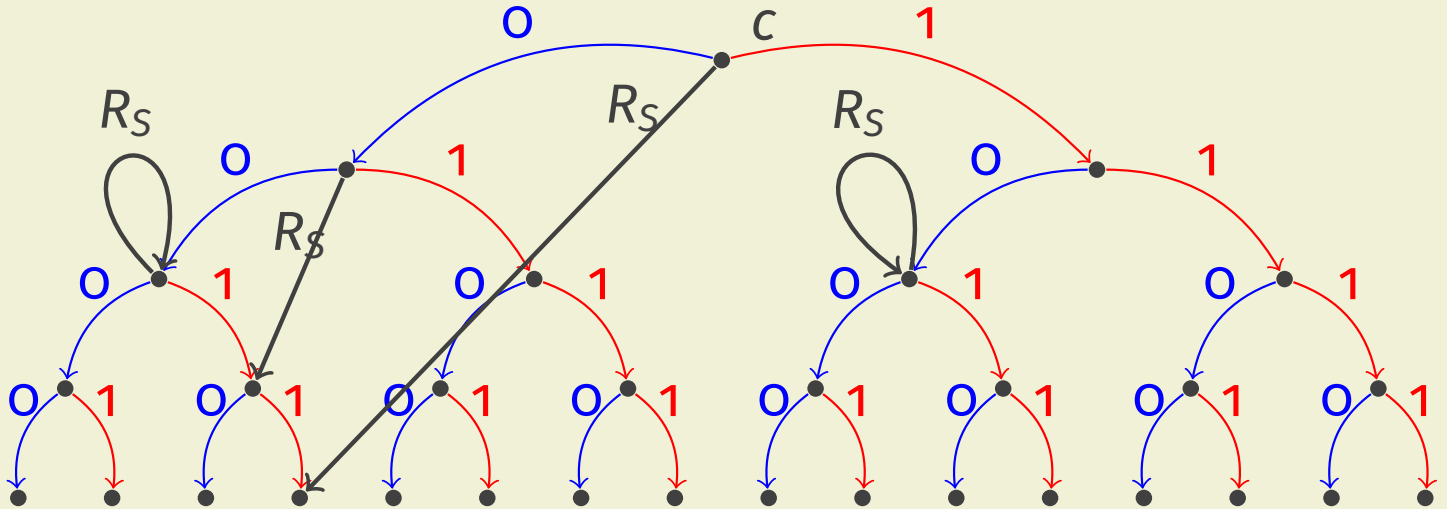
$$S \to \mathrm{o}S\mathrm{1} \qquad V(x) \to \exists y.R_{\mathrm{o}}(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_{\mathrm{1}}(x,z) \wedge V(z)$$

$$S \to \epsilon \qquad R_{\mathrm{o}}(x,y) \wedge R_S(y,z) \wedge R_{\mathrm{1}}(z,w) \to R_S(x,w)$$

$$\to R_S(x,x)$$

Note that $\mathcal{L}(G) = \mathrm{o}^n\mathrm{1}^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (2)

Example

Consider the CFG G with start symbol S (left) and the rule set $\mathcal{R}_G$ (right):

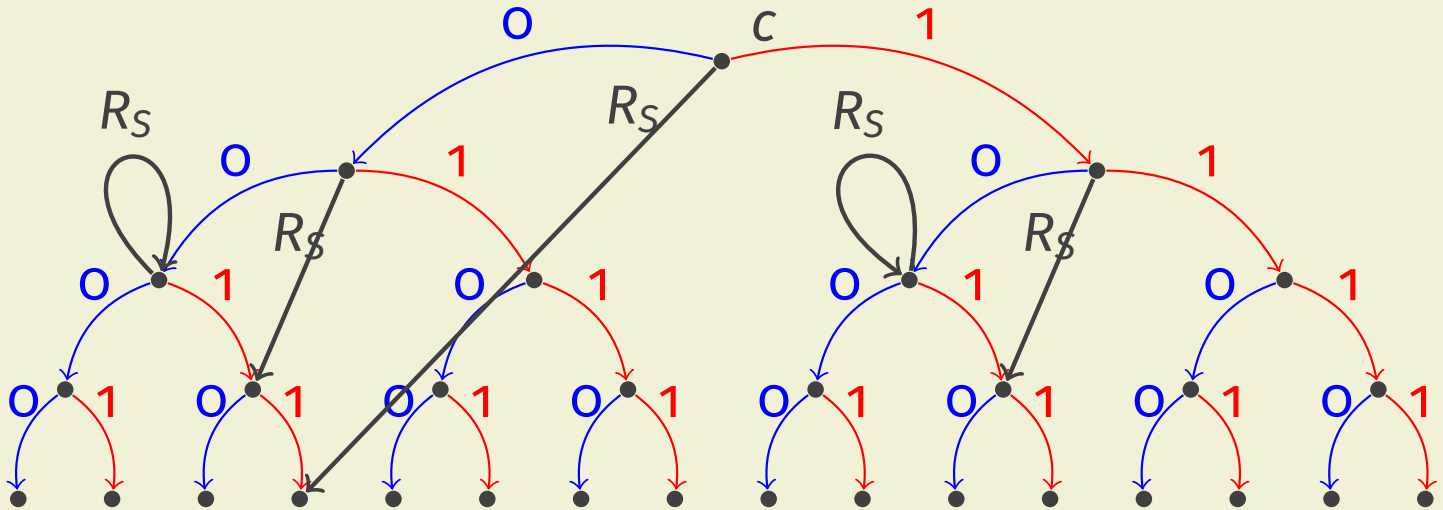
$$S \to \mathrm{o}S\mathrm{1} \qquad V(x) \to \exists y.R_{\mathrm{o}}(x,y) \wedge V(y) \qquad V(x) \to \exists y.R_{\mathrm{1}}(x,z) \wedge V(z)$$

$$S \to \epsilon \qquad R_{\mathrm{o}}(x,y) \wedge R_S(y,z) \wedge R_{\mathrm{1}}(z,w) \to R_S(x,w)$$

$$\to R_S(x,x)$$

Note that $\mathcal{L}(G) = \mathrm{o}^n\mathrm{1}^n$.

Output of the Skolem chase on input $\mathcal{K}_G = \langle \mathcal{R}_G, \{V(c)\} \rangle$:



BCQ Entailment is Undecidable (3)

Definition: Reduction

Consider some CFGs G_1 and G_2 with start symbols S_1 and S_2, respectively. Then, let $\mathcal{K}_{G_1}^{G_2}$ be the KB:

$$\langle \mathcal{R}_{G_1} \cup \mathcal{R}_{G_2} \cup \{R_{S_1}(x,y) \wedge R_{S_2}(x,y) \rightarrow Int\}, \{V(c)\} \rangle$$

BCQ Entailment is Undecidable (3)

Definition: Reduction

Consider some CFGs G_1 and G_2 with start symbols S_1 and S_2, respectively. Then, let $\mathcal{K}_{G_1}^{G_2}$ be the KB:

$$\langle \mathcal{R}_{G_1} \cup \mathcal{R}_{G_2} \cup \{R_{S_1}(x,y) \wedge R_{S_2}(x,y) \rightarrow Int\}, \{V(c)\} \rangle$$

Lemma

Given some CFGs G_1 and G_2,

$$\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset \iff \mathcal{K}_{G_1}^{G_2} \not\models Int$$

BCQ Entailment is Undecidable (3)

Definition: Reduction

Consider some CFGs G_1 and G_2 with start symbols S_1 and S_2, respectively. Then, let $\mathcal{K}_{G_1}^{G_2}$ be the KB:

$$\langle \mathcal{R}_{G_1} \cup \mathcal{R}_{G_2} \cup \{R_{S_1}(x,y) \wedge R_{S_2}(x,y) \rightarrow Int\}, \{V(c)\} \rangle$$

Lemma

Given some CFGs G_1 and G_2,

$$\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset \iff \mathcal{K}_{G_1}^{G_2} \not\models Int$$

Theorem

The problem of checking if a KB entails a BCQ is undecidable.

BCQ Entailment is Undecidable (3)

Definition: Reduction

Consider some CFGs G_1 and G_2 with start symbols S_1 and S_2, respectively. Then, let $\mathcal{K}_{G_1}^{G_2}$ be the KB:

$$\langle \mathcal{R}_{G_1} \cup \mathcal{R}_{G_2} \cup \{R_{S_1}(x,y) \wedge R_{S_2}(x,y) \rightarrow \mathit{Int}\}, \{V(c)\} \rangle$$

Lemma

Given some CFGs G_1 and G_2,

$$\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset \iff \mathcal{K}_{G_1}^{G_2} \not\models \mathit{Int}$$

Theorem

The problem of checking if a KB entails a BCQ is undecidable.

See [BV81] for an alternative proof of the above result.

Retrieving Decidability (in Some Cases)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

Retrieving Decidability (in Some Cases)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if the Skolem chase terminates on $\mathcal{R}$

Retrieving Decidability (in Some Cases)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if the Skolem chase terminates on $\mathcal{R}$
2. Apply this algorithm to $\mathcal{K}$ and check if its result entails γ

Retrieving Decidability (in Some Cases)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if the Skolem chase terminates on $\mathcal{R}$
2. Apply this algorithm to $\mathcal{K}$ and check if its result entails γ

Theorem from [GM14]

The problem of checking if the Skolem chase terminates for a rule set or knowledge base is undecidable.

Retrieving Decidability (in Some Cases)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if the Skolem chase terminates on $\mathcal{R}$
2. Apply this algorithm to $\mathcal{K}$ and check if its result entails γ

Theorem from [GM14]

The problem of checking if the Skolem chase terminates for a rule set or knowledge base is undecidable.

Remark: Universal Termination

Note that we consider a check that verifies if the chase of $\mathcal{R}$ terminates with respect to any fact set.

Retrieving Decidability (in Some Cases) (2)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

Retrieving Decidability (in Some Cases) (2)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if $\mathcal{R}$ is *weakly acyclic* (~~the Skolem chase terminates on $\mathcal{R}$~~)

Retrieving Decidability (in Some Cases) (2)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if $\mathcal{R}$ is *weakly acyclic* (~~the Skolem chase terminates on $\mathcal{R}$~~)
2. Compute the Skolem chase of $\mathcal{K}$ and check if its result entails γ

Retrieving Decidability (in Some Cases) (2)

In order to decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a BCQ γ:

1. Check if $\mathcal{R}$ is *weakly acyclic* (~~the Skolem chase terminates on $\mathcal{R}$~~)
2. Compute the Skolem chase of $\mathcal{K}$ and check if its result entails γ

Remark

Weak acyclicity [FKMP05] is a sufficient condition that guarantees universal Skolem chase termination.

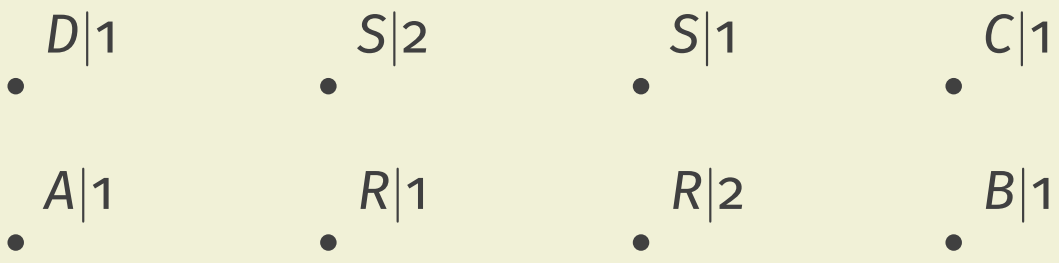
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



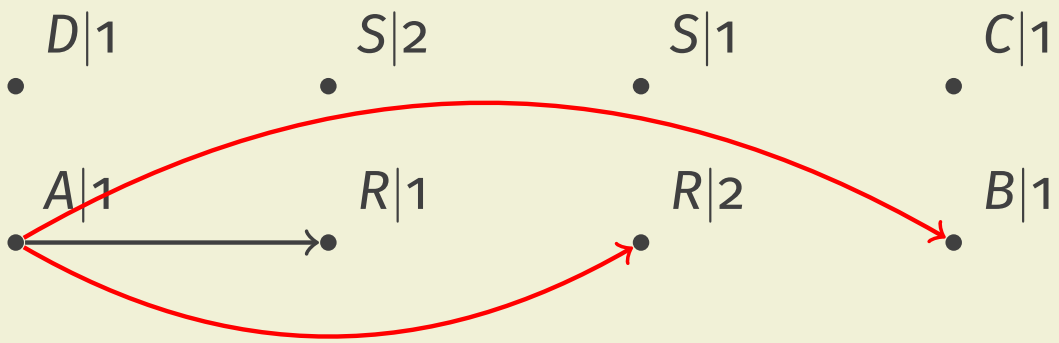
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



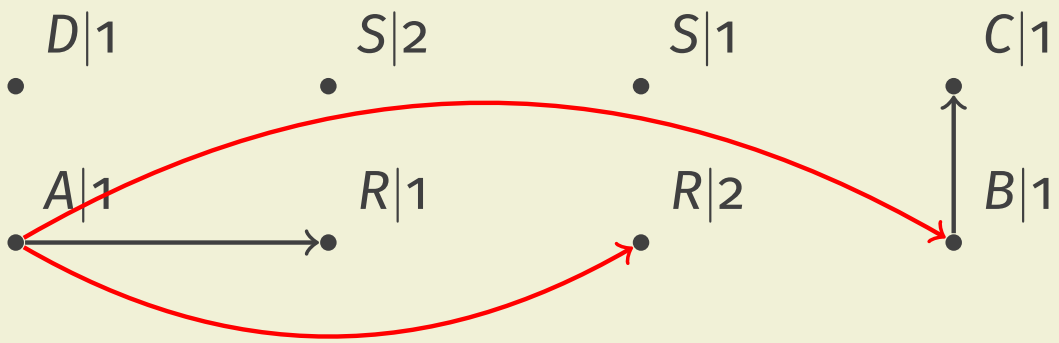
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



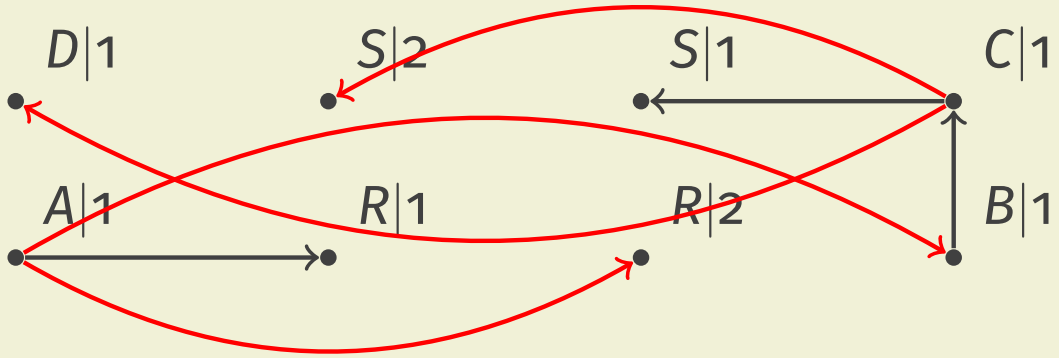
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



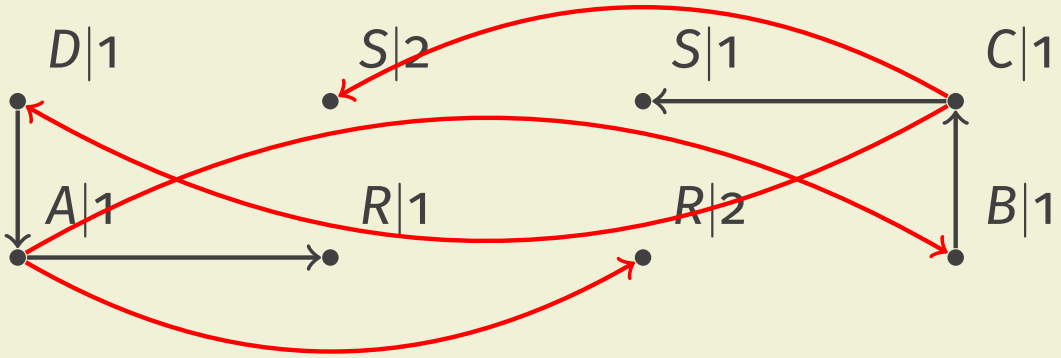
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



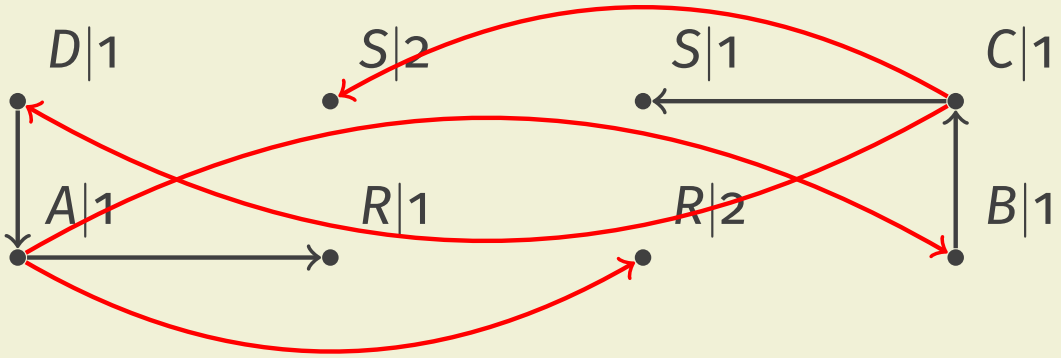
Weak Acyclicity (1)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



Fails the test: true negative!

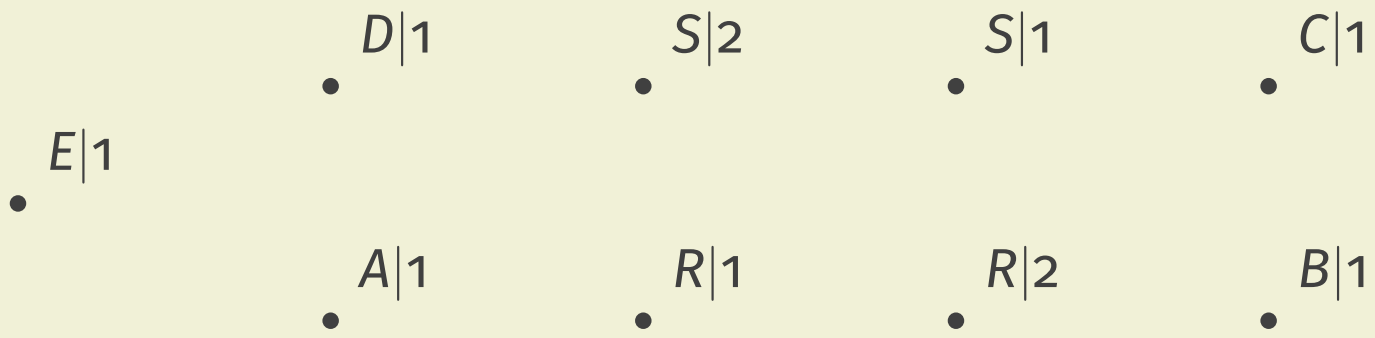
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



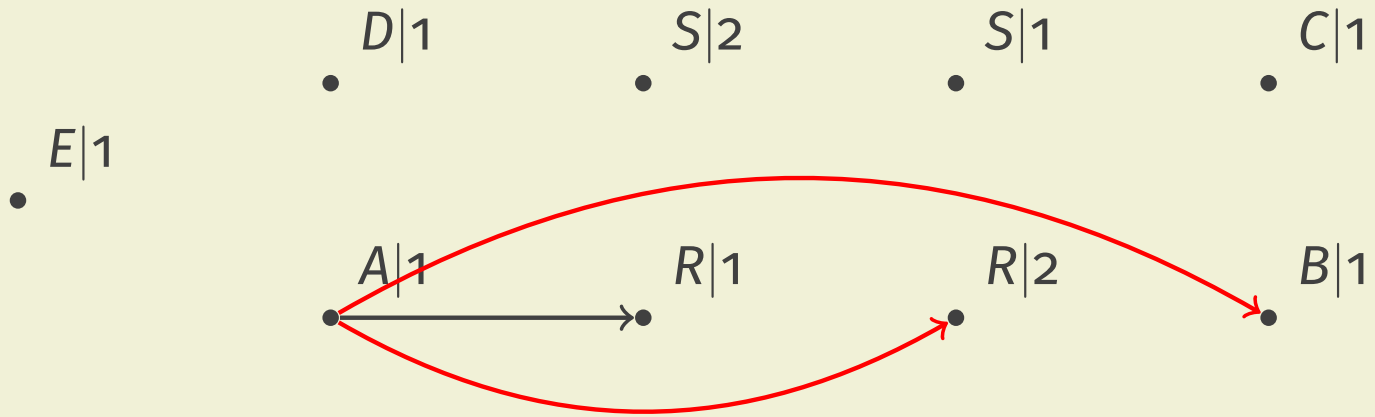
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



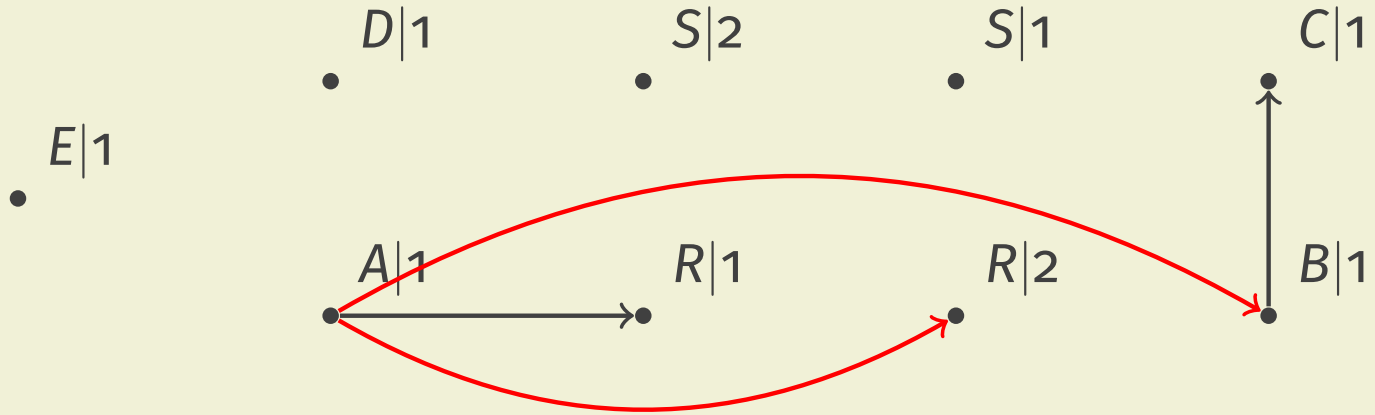
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



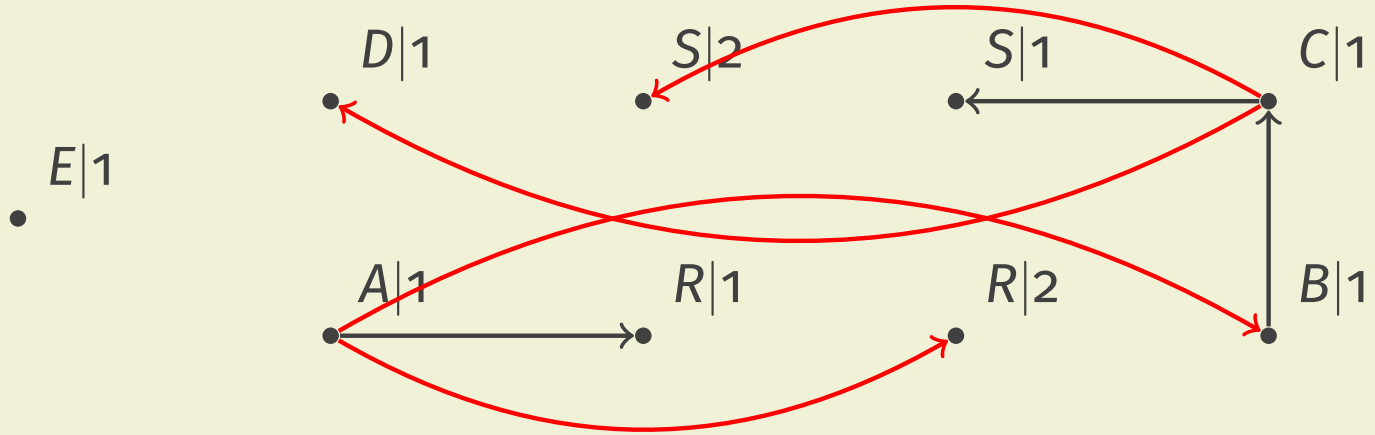
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



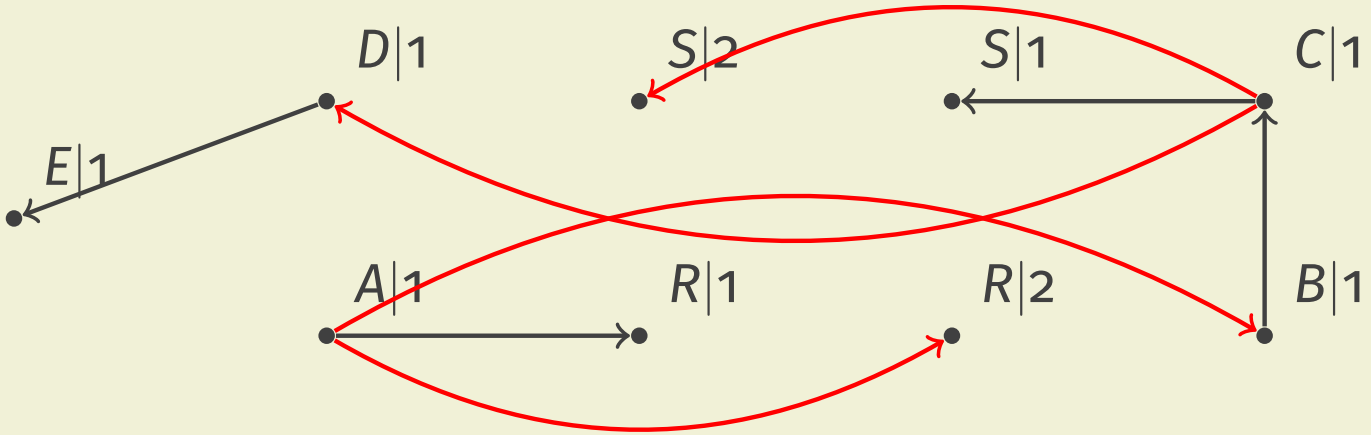
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



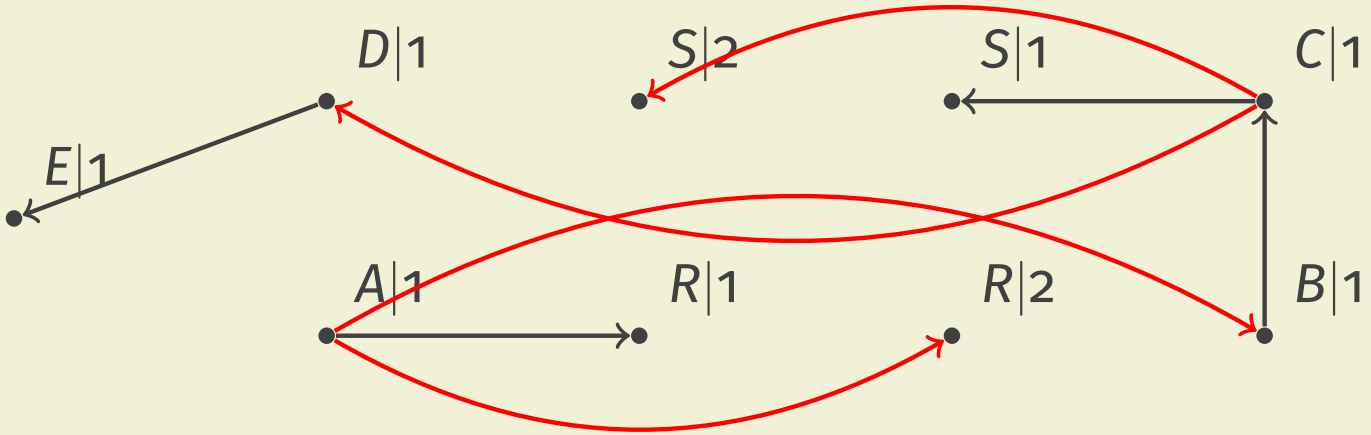
Weak Acyclicity (2)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to E(x)$$

and its weakly acyclic graph



Passes the test: (true) positive!

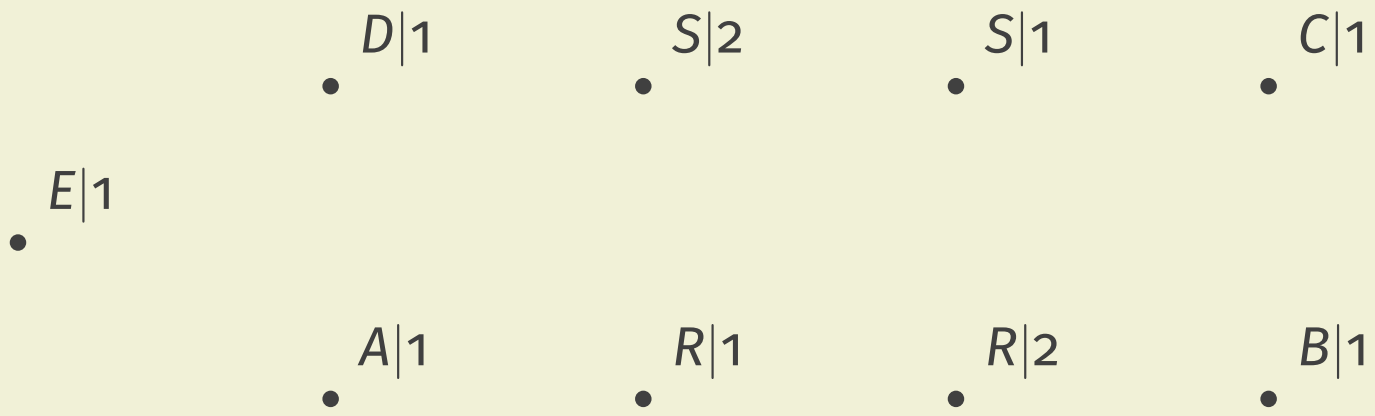
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



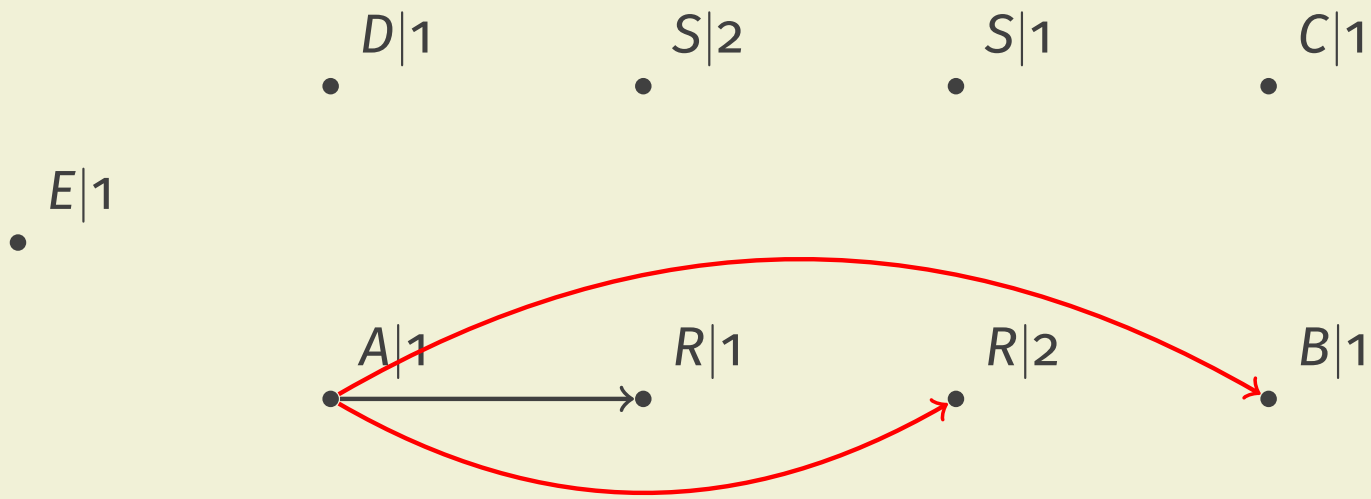
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



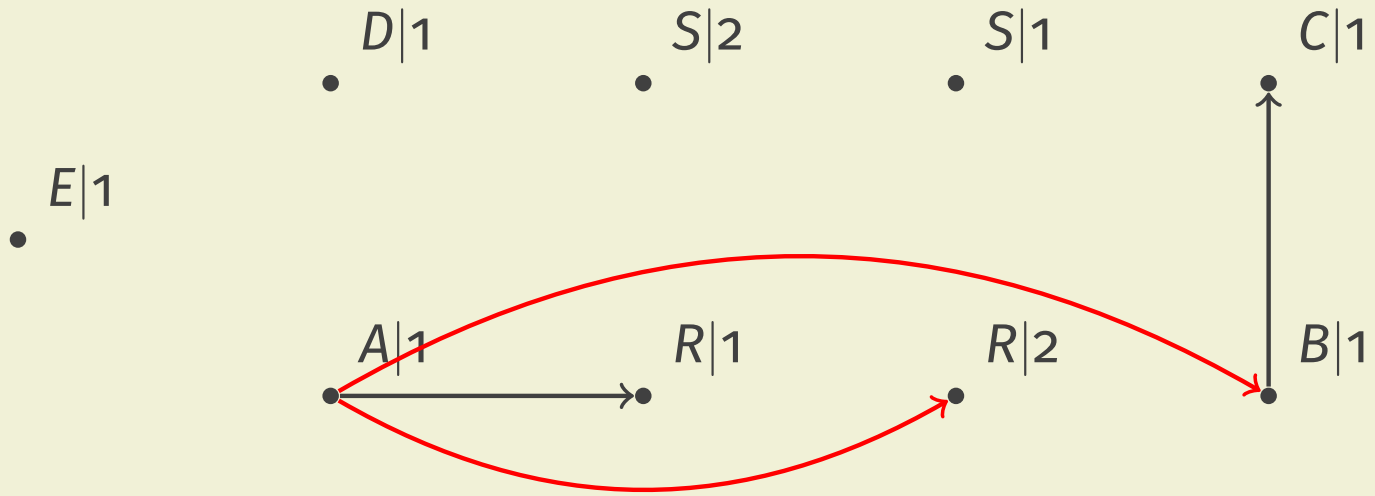
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



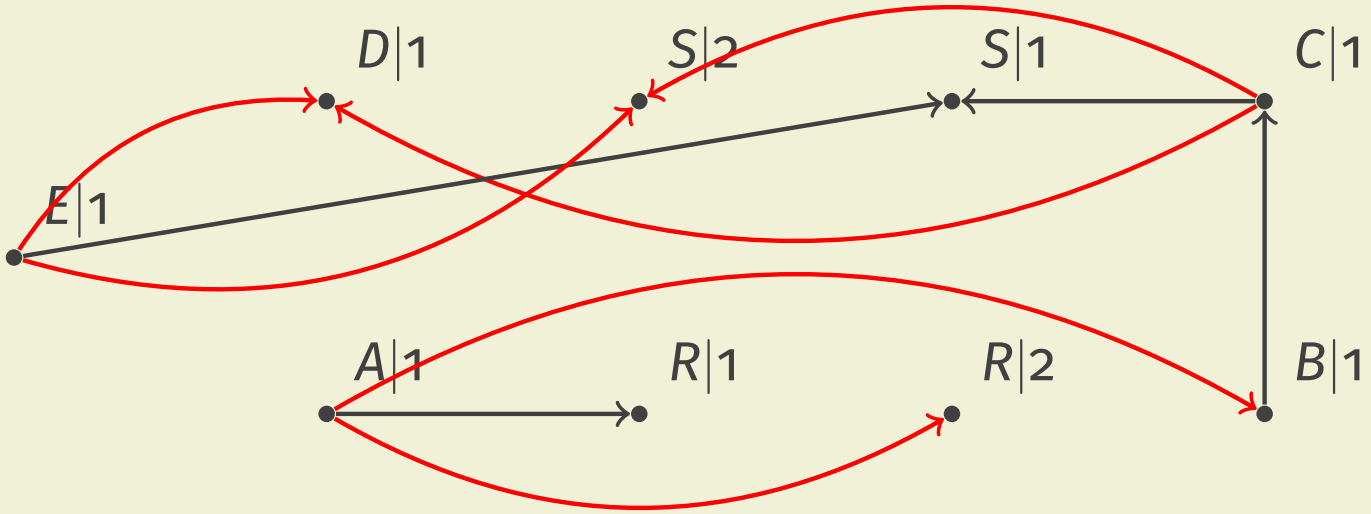
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



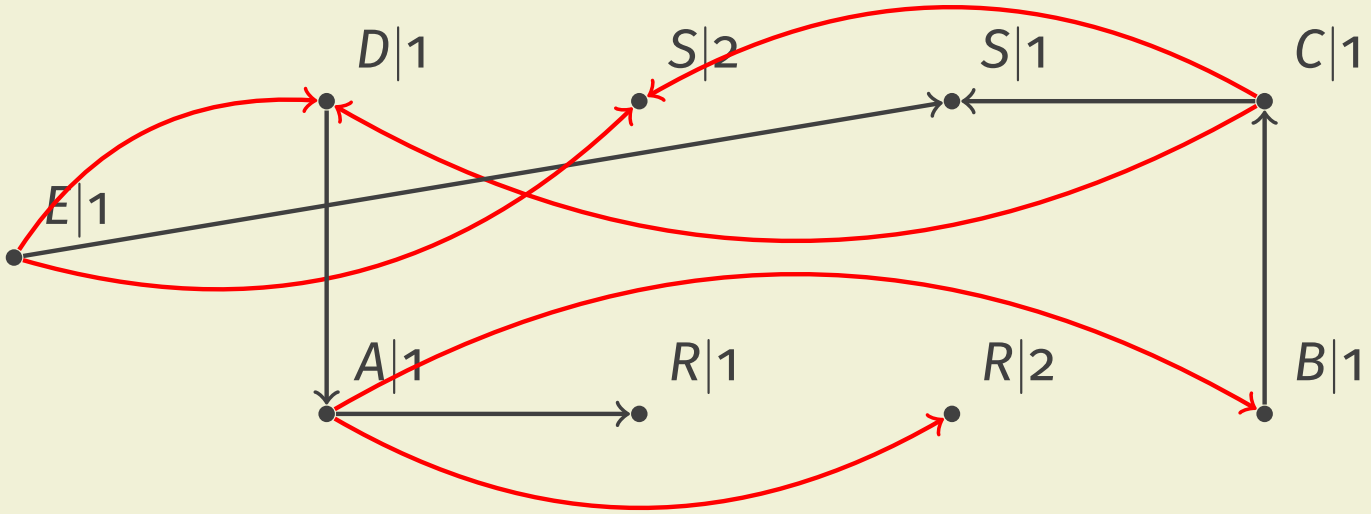
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



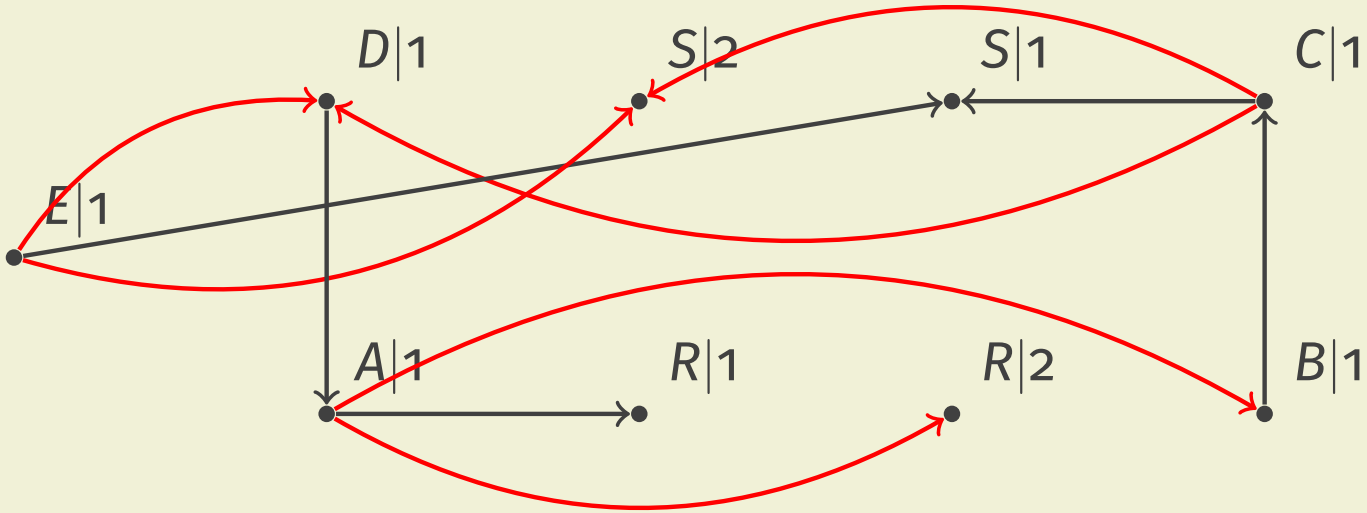
Weak Acyclicity (3)

Example

Consider the rule set $\mathcal{R}$

$$A(x) \to \exists y.R(x,y) \wedge B(y) \qquad B(x) \to C(x)$$
$$C(x) \wedge E(x) \to \exists y.S(x,y) \wedge D(y) \qquad D(x) \to A(x)$$

and its weakly acyclic graph



Fails the test: false negative!

A Recipe for Decidability

To decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a query γ:

1. *Decidability property:* the Skolem chase terminates on $\mathcal{R}$
2. *Sufficient condition:* check if $\mathcal{R}$ is weakly acyclicity
3. *Algorithm:* compute the chase of $\mathcal{K}$ and check if it entails γ

A Recipe for Decidability

To decide if a KB $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ entails a query γ:

1. *Decidability property:* the Skolem chase terminates on $\mathcal{R}$
2. *Sufficient condition:* check if $\mathcal{R}$ is weakly acyclicity
3. *Algorithm:* compute the chase of $\mathcal{K}$ and check if it entails γ

Seminar Talk: Michaël Thomazo (morning)

1. *Decidability property:* bounded tree-width model property
2. *Decidable language:* guarded existential rules
3. *Algorithm:* compute a finite rep. of a (possibly) infinite model

Research Goals

The recipe:

1. Choose a property that guarantees decidability
2. Choose a (decidable) sufficient condition for or a language that has this property
3. Apply the algorithm if the input satisfies the sufficient condition

Research Goals

The recipe:

1. Choose a property that guarantees decidability
2. Choose a (decidable) sufficient condition for or a language that has this property
3. Apply the algorithm if the input satisfies the sufficient condition

Research goals:

- Develop general sufficient conditions and languages that guarantee decidability (Tim's talk).

Research Goals

The recipe:

1. Choose a property that guarantees decidability
2. Choose a (decidable) sufficient condition for or a language that has this property
3. Apply the algorithm if the input satisfies the sufficient condition

Research goals:

- Develop general sufficient conditions and languages that guarantee decidability (Tim's talk).
- Study the complexity of reasoning for theories that satisfy some decidability property or some sufficient condition (Piotr's talk; Michaël's afternoon talk).

Research Goals

The recipe:

1. Choose a property that guarantees decidability
2. Choose a (decidable) sufficient condition for or a language that has this property
3. Apply the algorithm if the input satisfies the sufficient condition

Research goals:

- Develop general sufficient conditions and languages that guarantee decidability (Tim's talk).
- Study the complexity of reasoning for theories that satisfy some decidability property or some sufficient condition (Piotr's talk; Michaël's afternoon talk).
- Efficient algorithm implementation (Jacopo's talk).

Research Goals

The recipe:

1. Choose a property that guarantees decidability
2. Choose a (decidable) sufficient condition for or a language that has this property
3. Apply the algorithm if the input satisfies the sufficient condition

Research goals:

- Develop general sufficient conditions and languages that guarantee decidability (Tim's talk).
- Study the complexity of reasoning for theories that satisfy some decidability property or some sufficient condition (Piotr's talk; Michaël's afternoon talk).
- Efficient algorithm implementation (Jacopo's talk).
- Study the expressivity of decidable classes of rule sets (Sebastian's talk).

Thanks for your attention!

References I

- Catriel Beeri and Moshe Y. Vardi, *The implication problem for data dependencies*, Automata, Languages and Programming, 8th Colloquium, Proceedings (Shimon Even and Oded Kariv, eds.), Lecture Notes in Computer Science, vol. 115, Springer, 1981, pp. 73–85.
- Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa, *Data exchange: semantics and query answering*, Theor. Comput. Sci. **336** (2005), no. 1, 89–124.
- Tomasz Gogacz and Jerzy Marcinkowski, *All-instances termination of chase is undecidable*, Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Proceedings, Part II (Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, eds.), Lecture Notes in Computer Science, vol. 8573, Springer, 2014, pp. 293–304.

References II

Bruno Marnette, *Generalized schema-mappings: from termination to tractability*, Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009 (Jan Paredaens and Jianwen Su, eds.), ACM, 2009, pp. 13–22.