Ontology-Based Query Answering over Datalog-Expressible Rule Sets is Undecidable

David Carral[1], Lucas Larroque[2], and Michaël Thomazo[2]

[1]LIRMM, Inria, University of Montpellier, CNRS
[2]Inria, DI ENS, ENS, CNRS, PSL University

October 3, 2024

Preliminaries

The OBQA Problem for Existential Rules

Definition: The OBQA Problem

Given a set $\mathcal{R}$ of existential rules, a set $\mathcal{F}$ of facts, and a fact φ; decide if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ under standard FO semantics.

The OBQA Problem for Existential Rules

Definition: The OBQA Problem

Given a set $\mathcal{R}$ of existential rules, a set $\mathcal{F}$ of facts, and a fact φ; decide if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ under standard FO semantics.

A pair such as $\langle \mathcal{R}, \mathcal{F} \rangle$ above is a *knowledge base*.

The OBQA Problem for Existential Rules

Definition: The OBQA Problem

Given a set $\mathcal{R}$ of existential rules, a set $\mathcal{F}$ of facts, and a fact φ; decide if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ under standard FO semantics.

A pair such as $\langle \mathcal{R}, \mathcal{F} \rangle$ above is a *knowledge base*.

Example: Existential Rules

$$\forall x, y, z. \big(\mathit{Connected}(x, y) \land \mathit{Connected}(y, z) \rightarrow \mathit{Connected}(x, z) \big)$$

$$\forall x. \big(\mathit{Human}(x) \rightarrow \exists y. \mathit{HasParent}(x, y) \land \mathit{Human}(y) \big)$$

$$\forall x, y, z. \big(P(x, y, z) \land Q(x, z) \rightarrow \exists w, v. R(x, y, v, w) \land S(w, y) \land P(z, z, w) \big)$$

The OBQA Problem for Existential Rules

Definition: The OBQA Problem

Given a set $\mathcal{R}$ of existential rules, a set $\mathcal{F}$ of facts, and a fact φ; decide if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ under standard FO semantics.

A pair such as $\langle \mathcal{R}, \mathcal{F} \rangle$ above is a *knowledge base*.

Example: Existential Rules

$$\forall x, y, z. \big(\mathit{Connected}(x, y) \land \mathit{Connected}(y, z) \to \mathit{Connected}(x, z) \big)$$

$$\forall x. \big(\mathit{Human}(x) \to \exists y. \mathit{HasParent}(x, y) \land \mathit{Human}(y) \big)$$

$$\forall x, y, z. \big(P(x, y, z) \land Q(x, z) \to \exists w, v. R(x, y, v, w) \land S(w, y) \land P(z, z, w) \big)$$

We ignore universal quantifiers when writing rules.

The OBQA Problem for Existential Rules

Definition: The OBQA Problem

Given a set $\mathcal{R}$ of existential rules, a set $\mathcal{F}$ of facts, and a fact φ; decide if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi$ under standard FO semantics.

A pair such as $\langle \mathcal{R}, \mathcal{F} \rangle$ above is a *knowledge base*.

Example: Existential Rules

$$\forall x, y, z. \big(\mathit{Connected}(x, y) \land \mathit{Connected}(y, z) \to \mathit{Connected}(x, z) \big)$$

$$\forall x. \big(\mathit{Human}(x) \to \exists y. \mathit{HasParent}(x, y) \land \mathit{Human}(y) \big)$$

$$\forall x, y, z. \big(P(x, y, z) \land Q(x, z) \to \exists w, v. R(x, y, v, w) \land S(w, y) \land P(z, z, w) \big)$$

We ignore universal quantifiers when writing rules.

Example: Facts

$$\mathit{Connected}(\mathit{paris}, \mathit{montpellier}) \qquad P(c, d, e)$$

Addressing OBQA in Practice

Theorem

We cannot decide if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ.

Addressing OBQA in Practice

Theorem

We cannot decide if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ.

Given some KB $\langle \mathcal{R}, \mathcal{F} \rangle$ and soome fact φ, there are two main types of approaches to address OBQA:

Addressing OBQA in Practice

Theorem

We cannot decide if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ.

Given some KB $\langle \mathcal{R}, \mathcal{F} \rangle$ and soome fact φ, there are two main types of approaches to address OBQA:

1. Bottom-up/Materialisation: compute a finite representation of a *universal model* $\mathcal{U}$ of $\langle \mathcal{R}, \mathcal{F} \rangle$ and check if $\mathcal{U} \models \varphi$.

Addressing OBQA in Practice

Theorem

We cannot decide if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ.

Given some KB $\langle \mathcal{R}, \mathcal{F} \rangle$ and soome fact φ, there are two main types of approaches to address OBQA:

1. Bottom-up/Materialisation: compute a finite representation of a *universal model* $\mathcal{U}$ of $\langle \mathcal{R}, \mathcal{F} \rangle$ and check if $\mathcal{U} \models \varphi$.

2. Top-down/Rewriting: compute a rewriting γ of the *rule query* $\langle \mathcal{R}, \varphi \rangle$ and check if $\mathcal{F} \models \gamma$.

Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase

Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

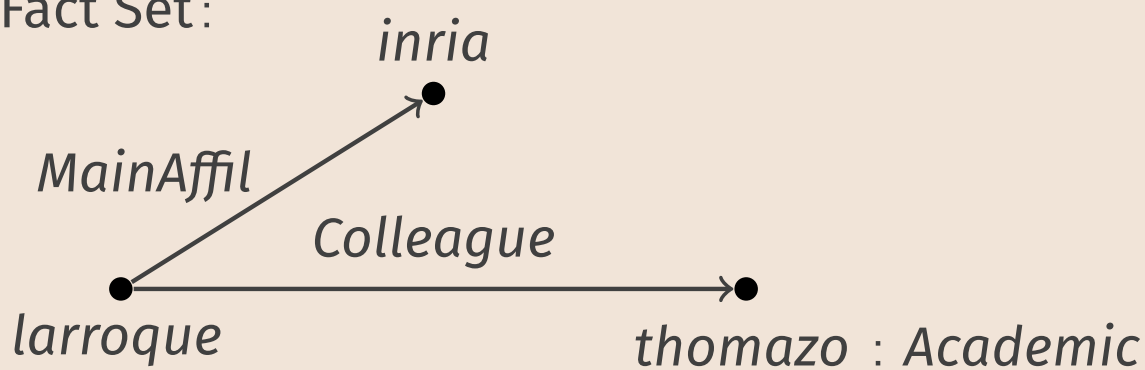
$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase

Input Fact Set :



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

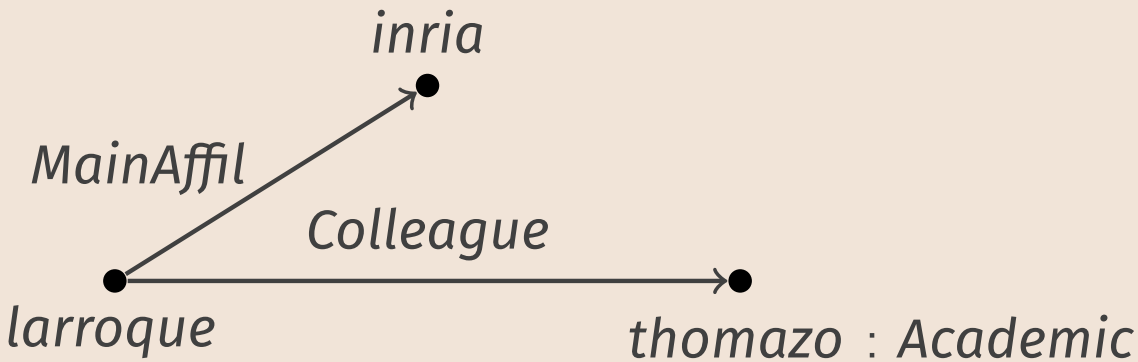
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

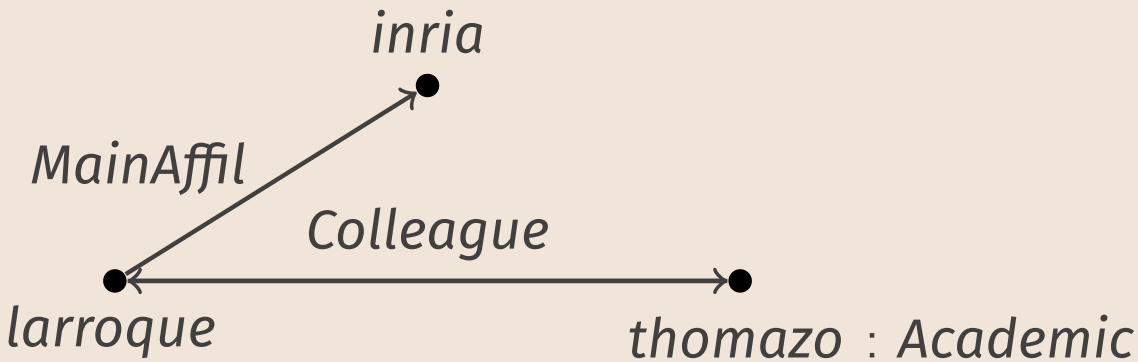
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

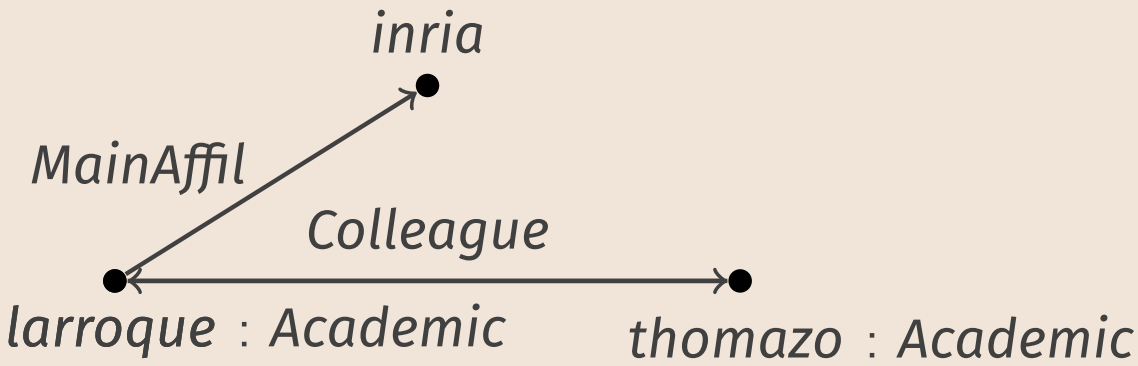
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

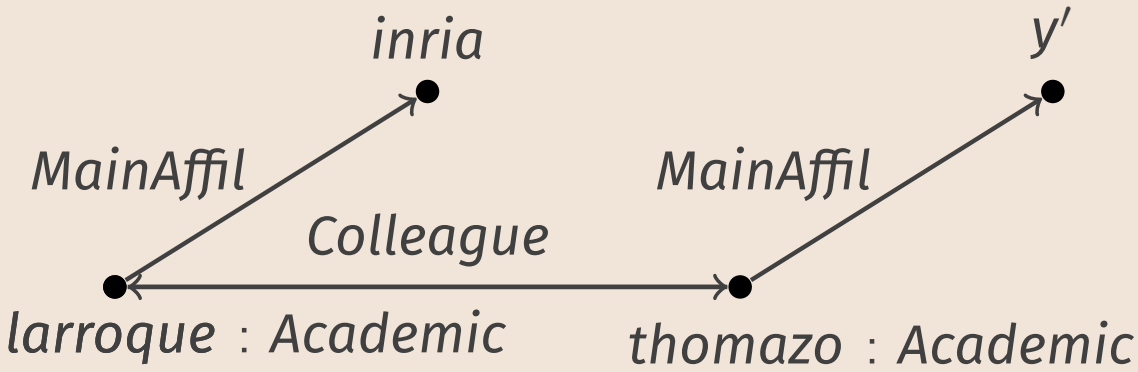
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

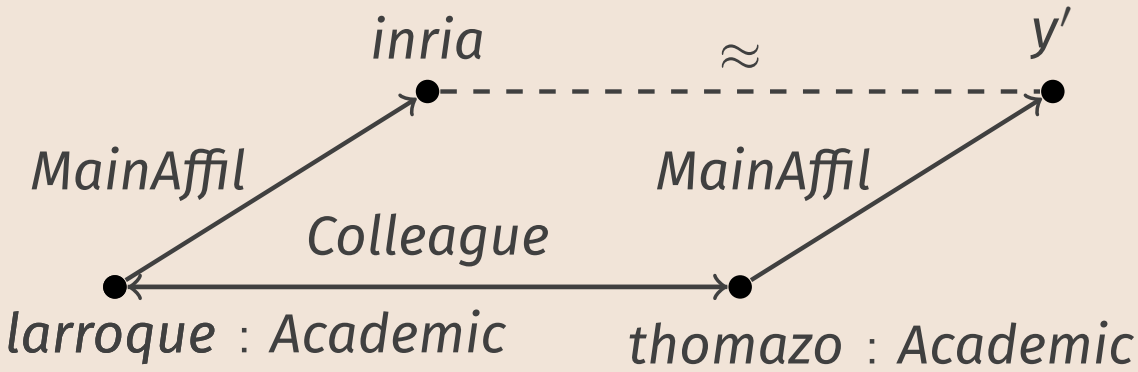
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

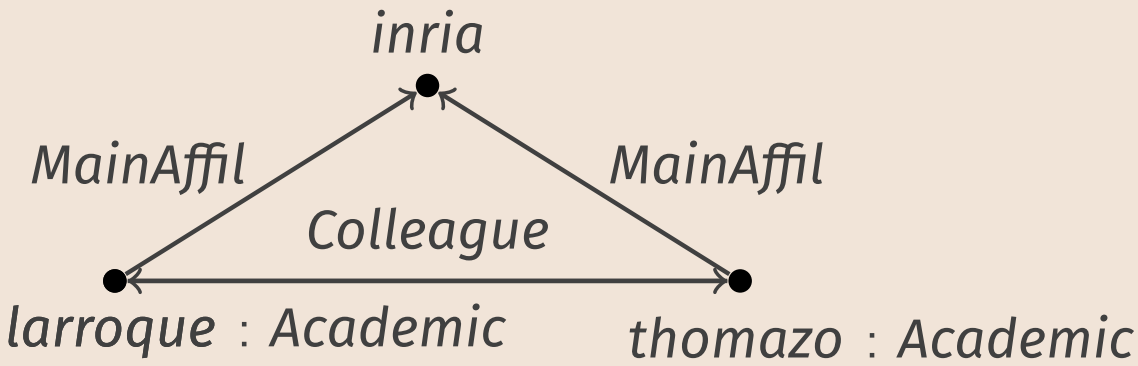
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

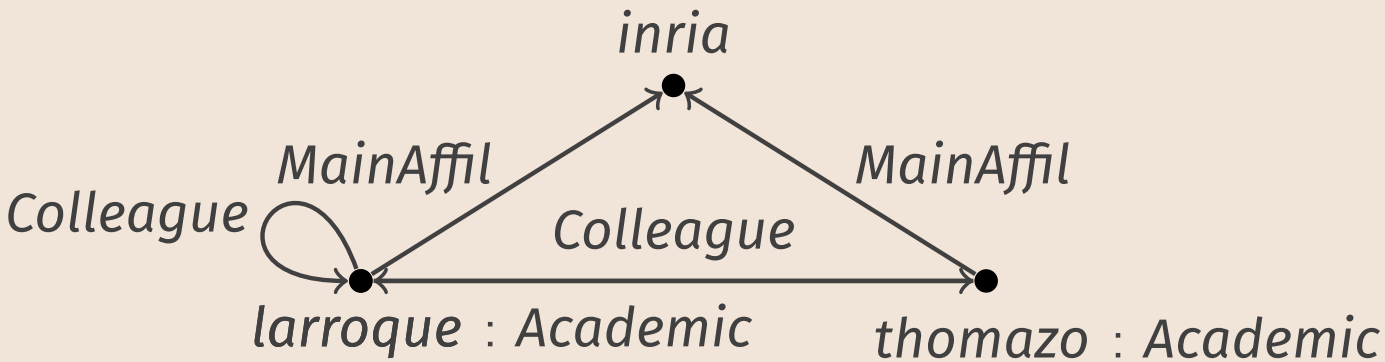
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A Chase Terminating Rule Set

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (1)$$

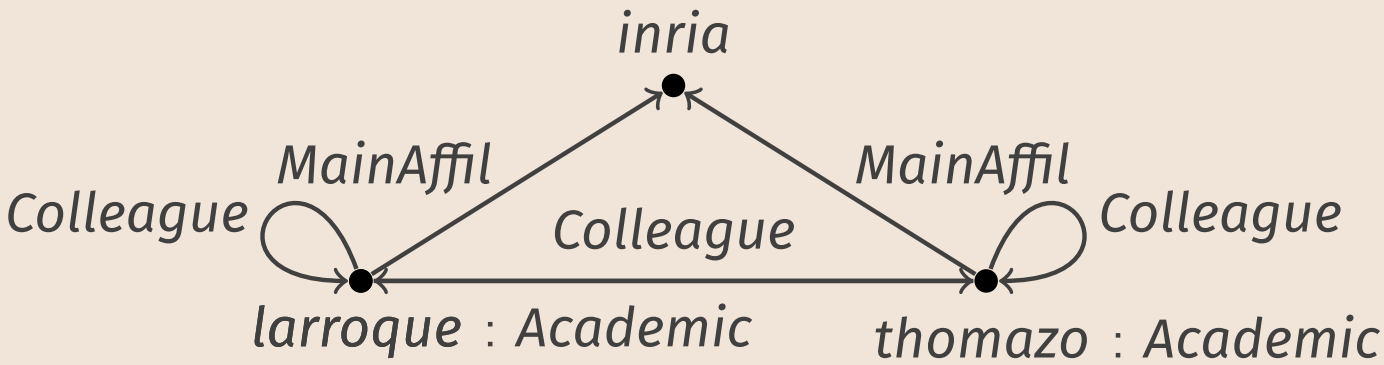
$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (2)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (3)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (4)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (5)$$

Definition: The Chase



Example: A UBCQ-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$MathProfessor(x) \rightarrow Professor(x) \tag{6}$$

$$MathClass(x) \rightarrow Class(x) \tag{7}$$

$$Teaches(x, y) \wedge Class(y) \rightarrow Professor(x) \tag{8}$$

$$Class(x) \rightarrow \exists y.IsTaughtBy(x, y) \wedge Professor(y) \tag{9}$$

$$Professor(x) \rightarrow \exists y.Teaches(x, y) \wedge Class(y) \tag{10}$$

Example: A UBCQ-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$\mathit{MathProfessor}(x) \rightarrow \mathit{Professor}(x) \tag{6}$$

$$\mathit{MathClass}(x) \rightarrow \mathit{Class}(x) \tag{7}$$

$$\mathit{Teaches}(x, y) \wedge \mathit{Class}(y) \rightarrow \mathit{Professor}(x) \tag{8}$$

$$\mathit{Class}(x) \rightarrow \exists y.\mathit{IsTaughtBy}(x, y) \wedge \mathit{Professor}(y) \tag{9}$$

$$\mathit{Professor}(x) \rightarrow \exists y.\mathit{Teaches}(x, y) \wedge \mathit{Class}(y) \tag{10}$$

The rule query $\langle \mathcal{R}, \mathit{Professor}(al) \rangle$ admits the UBCQ-rewriting:

$$\gamma = \mathit{Professor}(al) \vee \mathit{MathProfessor}(al) \vee \big(\exists y.\mathit{Teaches}(al, y) \wedge \mathit{Class}(y)\big) \vee \big(\exists y.\mathit{Teaches}(al, y) \wedge \mathit{MathClass}(y)\big)$$

Example: A UBCQ-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$\mathit{MathProfessor}(x) \to \mathit{Professor}(x) \tag{6}$$
$$\mathit{MathClass}(x) \to \mathit{Class}(x) \tag{7}$$
$$\mathit{Teaches}(x, y) \wedge \mathit{Class}(y) \to \mathit{Professor}(x) \tag{8}$$
$$\mathit{Class}(x) \to \exists y.\mathit{IsTaughtBy}(x, y) \wedge \mathit{Professor}(y) \tag{9}$$
$$\mathit{Professor}(x) \to \exists y.\mathit{Teaches}(x, y) \wedge \mathit{Class}(y) \tag{10}$$

The rule query $\langle \mathcal{R}, \mathit{Professor}(al) \rangle$ admits the UBCQ-rewriting:

$$\gamma = \mathit{Professor}(al) \vee \mathit{MathProfessor}(al) \vee \big(\exists y.\mathit{Teaches}(al, y) \wedge \mathit{Class}(y)\big) \vee \big(\exists y.\mathit{Teaches}(al, y) \wedge \mathit{MathClass}(y)\big)$$

That is, for every fact set $\mathcal{F}$, we have that:

$$\langle \mathcal{R}, \mathcal{F} \rangle \models \mathit{Professor}(al) \iff \mathcal{F} \models \gamma$$

Motivation

One Procedure to Rewrite Them All

Definition: UBCQ-Expressibility

A rule query $\langle \mathcal{R}, \varphi \rangle$ is *UBCQ-expressible* if it admits a *UBCQ-rewriting*. That is, if there is a UBCQ γ such that, for every fact set $\mathcal{F}$:

$$\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi \iff \mathcal{F} \models \gamma$$

One Procedure to Rewrite Them All

Definition: UBCQ-Expressibility

A rule query $\langle \mathcal{R}, \varphi \rangle$ is *UBCQ-expressible* if it admits a *UBCQ-rewriting*. That is, if there is a UBCQ γ such that, for every fact set $\mathcal{F}$:

$$\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi \iff \mathcal{F} \models \gamma$$

Theorem

The class of all UBCQ-expressible rule queries is UBCQ-rewritable.

One Procedure to Rewrite Them All

Definition: UBCQ-Expressibility

A rule query $\langle \mathcal{R}, \varphi \rangle$ is *UBCQ-expressible* if it admits a *UBCQ-rewriting*. That is, if there is a UBCQ γ such that, for every fact set $\mathcal{F}$:

$$\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi \iff \mathcal{F} \models \gamma$$

Theorem

The class of all UBCQ-expressible rule queries is UBCQ-rewritable.

Theorem (see [KLMT15] for more info)

There is a procedure that, on input $\langle \mathcal{R}, \varphi \rangle$ a rule query,

- outputs a UBCQ-rewriting of $\langle \mathcal{R}, \varphi \rangle$ and
- terminates if $\langle \mathcal{R}, \varphi \rangle$ is UBCQ-expressible.

One Procedure to Rewrite Them All

Definition: UBCQ-Expressibility

A rule query $\langle \mathcal{R}, \varphi \rangle$ is *UBCQ-expressible* if it admits a *UBCQ-rewriting*. That is, if there is a UBCQ γ such that, for every fact set $\mathcal{F}$:

$$\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi \iff \mathcal{F} \models \gamma$$

Theorem

The class of all UBCQ-expressible rule queries is UBCQ-rewritable.

Theorem (see [KLMT15] for more info)

There is a procedure that, on input $\langle \mathcal{R}, \varphi \rangle$ a rule query,

- outputs a UBCQ-rewriting of $\langle \mathcal{R}, \varphi \rangle$ and
- terminates if $\langle \mathcal{R}, \varphi \rangle$ is UBCQ-expressible.

Remark

We can reuse the very same procedure to rewrite every UBCQ-expressible fragment: OWL QL, linear, sticky...

Example: A Datalog-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$Colleague(x, y) \wedge MainAff(x, z) \wedge MainAff(y, w) \rightarrow z \approx w \quad (11)$$

$$Colleague(x, y) \rightarrow Colleague(y, x) \quad (12)$$

$$Colleague(x, y) \wedge Colleague(y, z) \rightarrow Colleague(x, z) \quad (13)$$

$$Academic(x) \wedge Colleague(x, y) \rightarrow Academic(y) \quad (14)$$

$$Academic(x) \rightarrow \exists y.MainAff(x, y) \quad (15)$$

Example: A Datalog-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$\mathit{Colleague}(x, y) \wedge \mathit{MainAff}(x, z) \wedge \mathit{MainAff}(y, w) \rightarrow z \approx w \quad (11)$$

$$\mathit{Colleague}(x, y) \rightarrow \mathit{Colleague}(y, x) \quad (12)$$

$$\mathit{Colleague}(x, y) \wedge \mathit{Colleague}(y, z) \rightarrow \mathit{Colleague}(x, z) \quad (13)$$

$$\mathit{Academic}(x) \wedge \mathit{Colleague}(x, y) \rightarrow \mathit{Academic}(y) \quad (14)$$

$$\mathit{Academic}(x) \rightarrow \exists y.\mathit{MainAff}(x, y) \quad (15)$$

The rule query $\langle \mathcal{R}, \mathit{MainAff}(\mathit{alice}, \mathit{inria}) \rangle$ is not UBCQ-expressible.

Example: A Datalog-Expressible Rule Query

Consider the rule set $\mathcal{R}$:

$$\mathit{Colleague}(x,y) \wedge \mathit{MainAff}(x,z) \wedge \mathit{MainAff}(y,w) \rightarrow z \approx w \quad (11)$$

$$\mathit{Colleague}(x,y) \rightarrow \mathit{Colleague}(y,x) \quad (12)$$

$$\mathit{Colleague}(x,y) \wedge \mathit{Colleague}(y,z) \rightarrow \mathit{Colleague}(x,z) \quad (13)$$

$$\mathit{Academic}(x) \wedge \mathit{Colleague}(x,y) \rightarrow \mathit{Academic}(y) \quad (14)$$

$$\mathit{Academic}(x) \rightarrow \exists y.\mathit{MainAff}(x,y) \quad (15)$$

The rule query $\langle \mathcal{R}, \mathit{MainAff}(\mathit{alice}, \mathit{inria}) \rangle$ is not UBCQ-expressible. However, we obtain a datalog-rewriting by replacing (15) in $\mathcal{R}$ with

$$\mathit{Colleague}(x,y) \wedge \mathit{MainAff}(x,z) \rightarrow \mathit{MainAff}(y,z)$$

Many Procedures to Rewrite into Datalog

Source Language	Target Language	Implemented	Reference
$\mathcal{SHIQ}$	Disj. Datalog	Yes	[HMS07]
$\mathcal{SHIQ}b_S$	Disj. Datalog	No	[RKH12]
Horn-$\mathcal{ALCHOIQ}$	Datalog	Yes	[CDK18]
Horn-$\mathcal{SRIQ}$	Datalog	Yes	[CGK19]

Many Procedures to Rewrite into Datalog

Source Language	Target Language	Implemented	Reference
$\mathcal{SHIQ}$	Disj. Datalog	Yes	[HMS07]
$\mathcal{SHIQ}b_S$	Disj. Datalog	No	[RKH12]
Horn-$\mathcal{ALCHOIQ}$	Datalog	Yes	[CDK18]
Horn-$\mathcal{SRIQ}$	Datalog	Yes	[CGK19]
Bounded Depth Rules	Datalog	No	[Mar12]

Many Procedures to Rewrite into Datalog

Source Language	Target Language	Implemented	Reference
$\mathcal{SHIQ}$	Disj. Datalog	Yes	[HMS07]
$\mathcal{SHIQ}b_S$	Disj. Datalog	No	[RKH12]
Horn-$\mathcal{ALCHOIQ}$	Datalog	Yes	[CDK18]
Horn-$\mathcal{SRIQ}$	Datalog	Yes	[CGK19]
Bounded Depth Rules	Datalog	No	[Mar12]
Frontier Guarded Rules	Datalog	No	[BBtC13]
Nearly Guarded Rules	Datalog	No	[GRS14]
Guarded Disj. Rules	Disj. Datalog	No	[AOS18]
Guarded Rules	Datalog	Yes	[BBG$^+$22]

Many Procedures to Rewrite into Datalog

Source Language	Target Language	Implemented	Reference
$\mathcal{SHIQ}$	Disj. Datalog	Yes	[HMS07]
$\mathcal{SHIQ}b_S$	Disj. Datalog	No	[RKH12]
Horn-$\mathcal{ALCHOIQ}$	Datalog	Yes	[CDK18]
Horn-$\mathcal{SRIQ}$	Datalog	Yes	[CGK19]
Bounded Depth Rules	Datalog	No	[Mar12]
Frontier Guarded Rules	Datalog	No	[BBtC13]
Nearly Guarded Rules	Datalog	No	[GRS14]
Guarded Disj. Rules	Disj. Datalog	No	[AOS18]
Guarded Rules	Datalog	Yes	[BBG$^+$22]
Warded rules	Datalog	Yes	[BGPS22]

Many Procedures to Rewrite into Datalog

Source Language	Target Language	Implemented	Reference
$\mathcal{SHIQ}$	Disj. Datalog	Yes	[HMS07]
$\mathcal{SHIQ}b_S$	Disj. Datalog	No	[RKH12]
Horn-$\mathcal{ALCHOIQ}$	Datalog	Yes	[CDK18]
Horn-$\mathcal{SRIQ}$	Datalog	Yes	[CGK19]
Bounded Depth Rules	Datalog	No	[Mar12]
Frontier Guarded Rules	Datalog	No	[BBtC13]
Nearly Guarded Rules	Datalog	No	[GRS14]
Guarded Disj. Rules	Disj. Datalog	No	[AOS18]
Guarded Rules	Datalog	Yes	[BBG$^+$22]
Warded rules	Datalog	Yes	[BGPS22]
Linear	Non-Rec. Datalog	No	[GS12]
Sticky(-Join)	Non-Rec. Datalog	No	[GS12]

A Single Procedure to Rewrite into Datalog?

Research Question

Is the datalog-expressible fragment datalog-rewritable?

A Single Procedure to Rewrite into Datalog?

Research Question

Is the datalog-expressible fragment datalog-rewritable?

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

A Single Procedure to Rewrite into Datalog?

Research Question

Is the datalog-expressible fragment datalog-rewritable?

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Corollary

The class of all datalog-expressible rule queries is not datalog-rewritable.

Reasoning with Datalog-Expressible Rule Queries: An Undecidable Problem

Reasoning with Dat-Expressible Rule Queries

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Proof. We reduce machine M to a rule set $\mathcal{R}_M$ such that:

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Proof. We reduce machine M to a rule set $\mathcal{R}_M$ such that:

- Lemma 1. The machine M halts on ϵ if and only if the KB $\langle \mathcal{R}_M, \emptyset \rangle$ entails the (nullary) fact *Halt*.

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Proof. We reduce machine M to a rule set $\mathcal{R}_M$ such that:

- Lemma 1. The machine M halts on ϵ if and only if the KB $\langle \mathcal{R}_M, \emptyset \rangle$ entails the (nullary) fact *Halt*.
- Lemma 2. The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

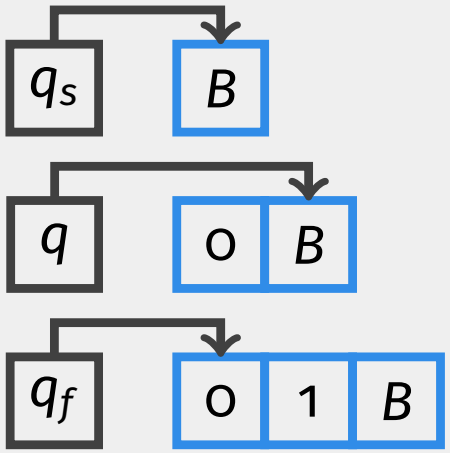
Proof. We reduce machine M to a rule set $\mathcal{R}_M$ such that:

- Lemma 1. The machine M halts on ϵ if and only if the KB $\langle \mathcal{R}_M, \emptyset \rangle$ entails the (nullary) fact *Halt*.
- Lemma 2. The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

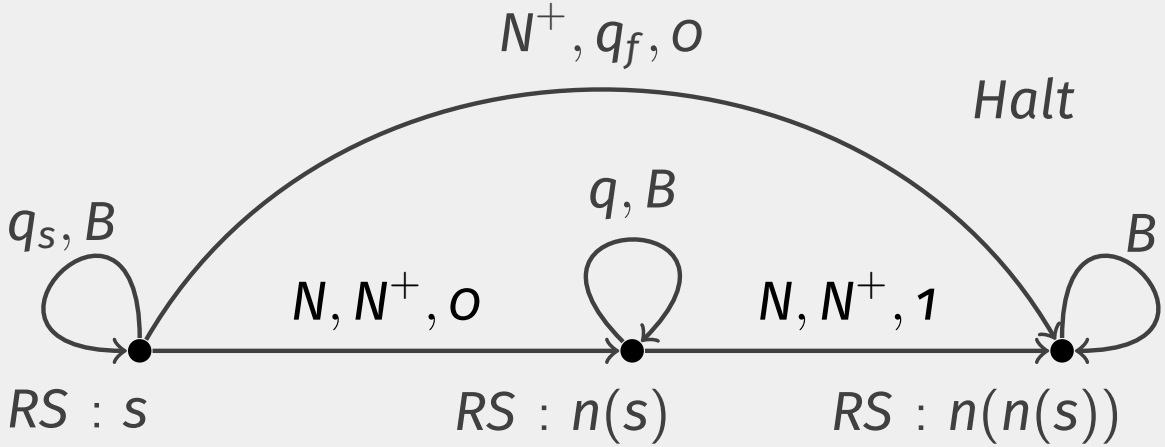
After proving the above, the theorem follows by contradiction.

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



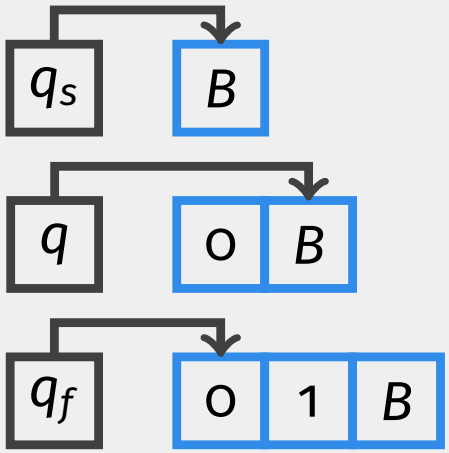
A universal model of $\langle \mathcal{R}_M, \emptyset \rangle$:



Lemma 1. The machine M halts on the empty word if and only if the KB $\langle \mathcal{R}_M, \emptyset \rangle$ entails the fact *Halt*.

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

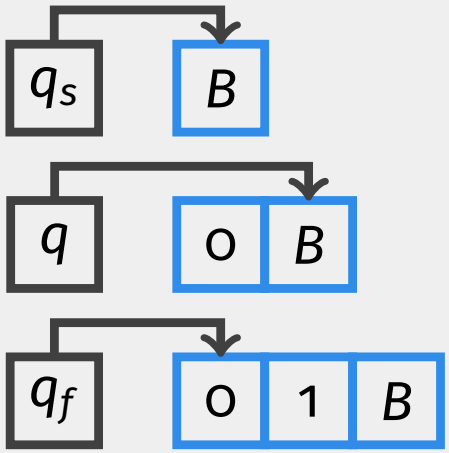
The computation of M on ϵ:



The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:

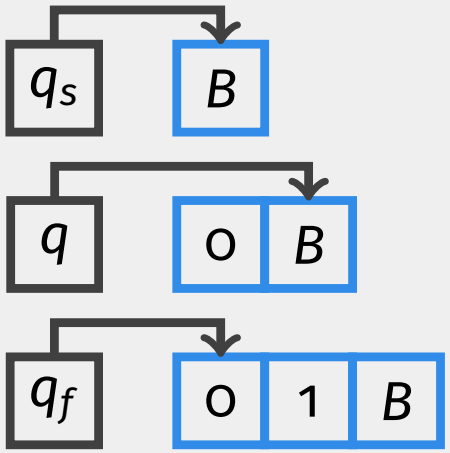


The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

$$\text{Apply} \rightarrow \exists s.q_s(s, s) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:

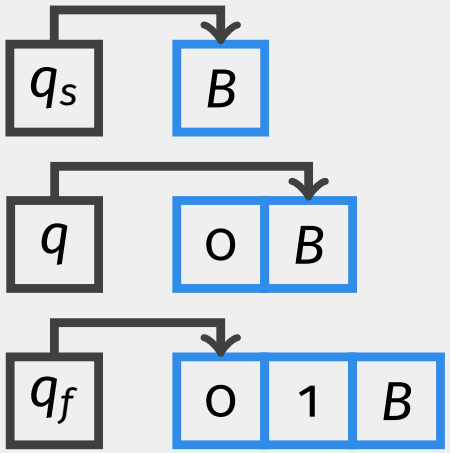


The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

$$\text{Apply} \rightarrow \exists s.q_s(s,s) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:

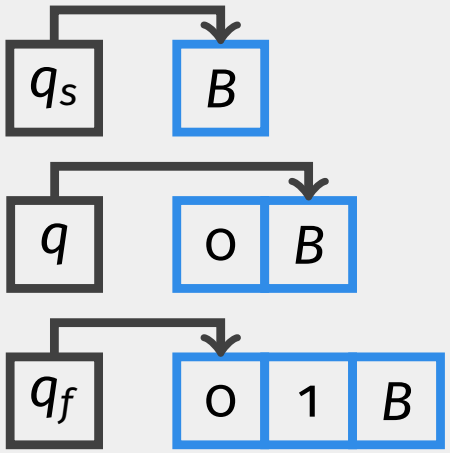


The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

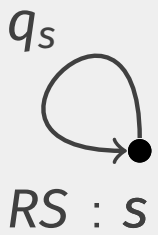
Apply $q_s(x, x) \rightarrow RS(x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



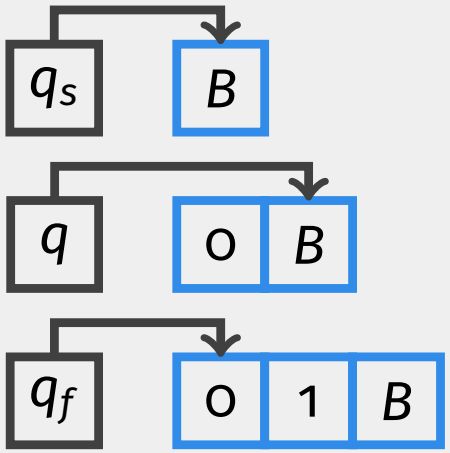
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



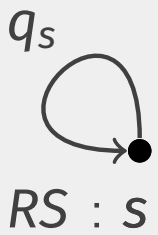
Apply $q_s(x, x) \rightarrow RS(x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



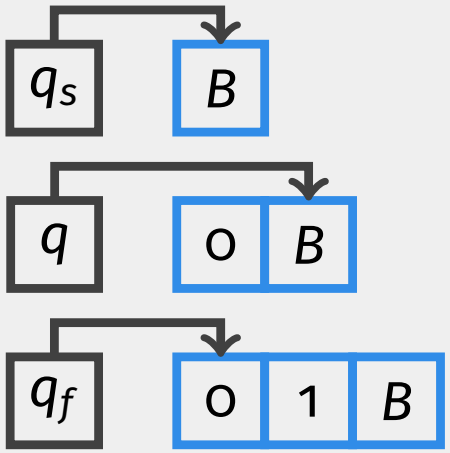
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



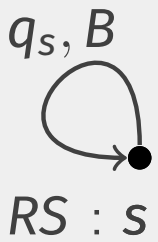
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



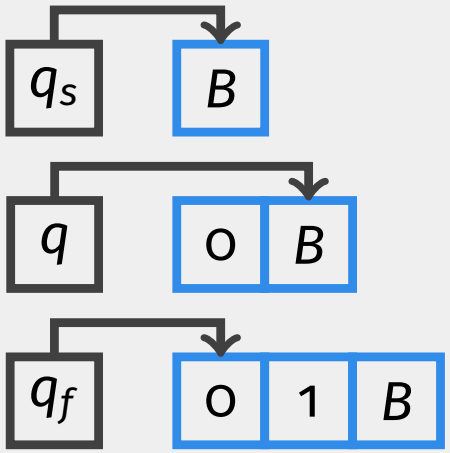
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



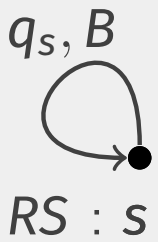
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



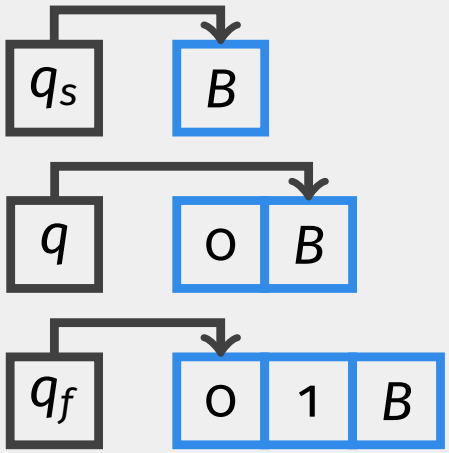
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



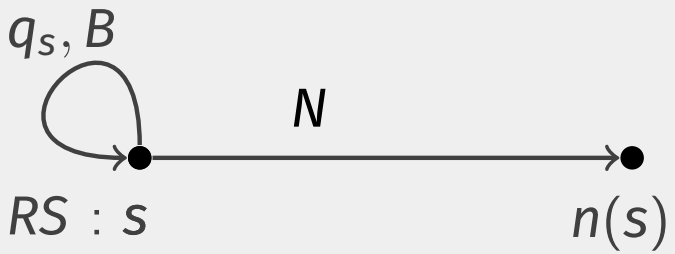
Since q_s is a non-final state in M, we can apply $q_s(p, c) \wedge RS(c) \rightarrow \exists n.N(c, n) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



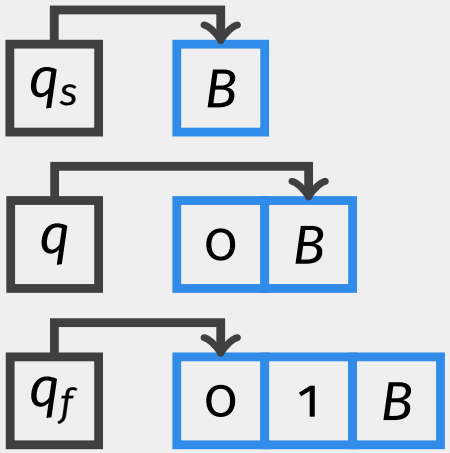
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



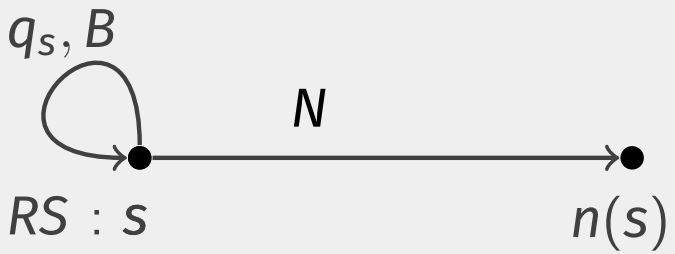
Since q_s is a non-final state in M, we can apply $q_s(p, c) \wedge RS(c) \rightarrow \exists n.N(c, n) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



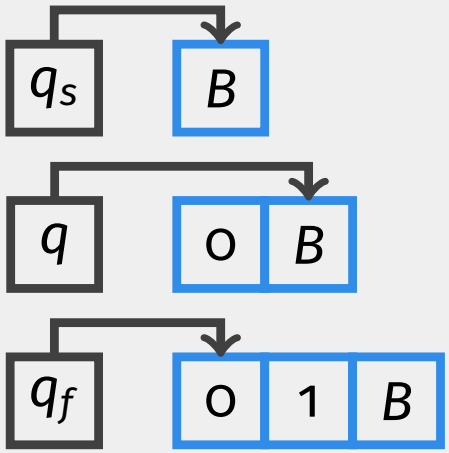
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



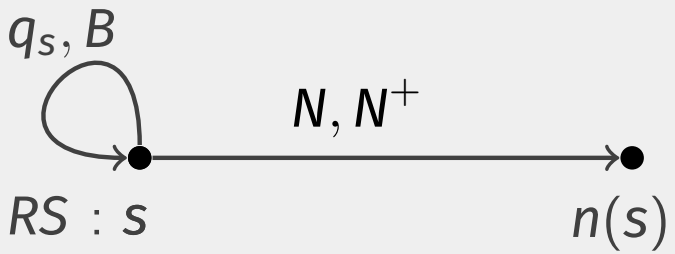
Apply $N(\ell_i, \ell_{i+1}) \rightarrow N^+(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



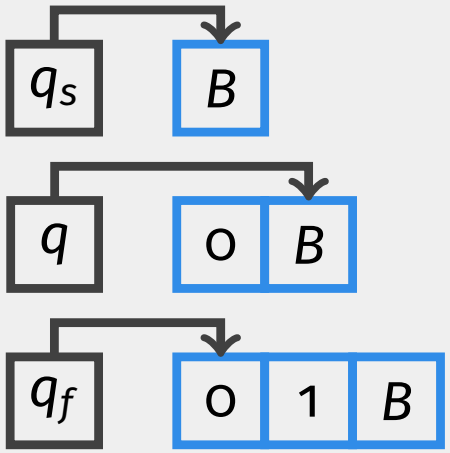
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



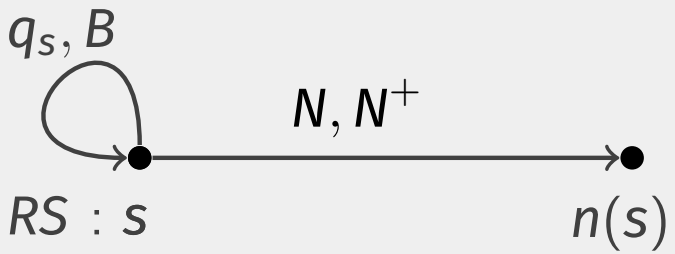
Apply $N(\ell_i, \ell_{i+1}) \rightarrow N^+(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



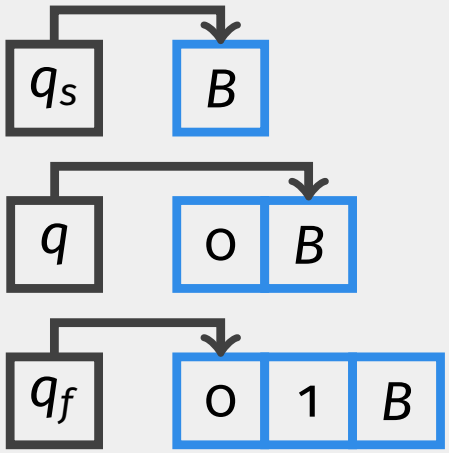
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



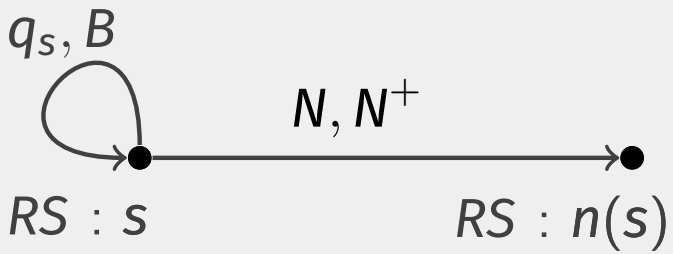
Apply $q_s(\ell_1, \ell_1) \wedge N^+(\ell_1, \ell_i) \rightarrow RS(\ell_i) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



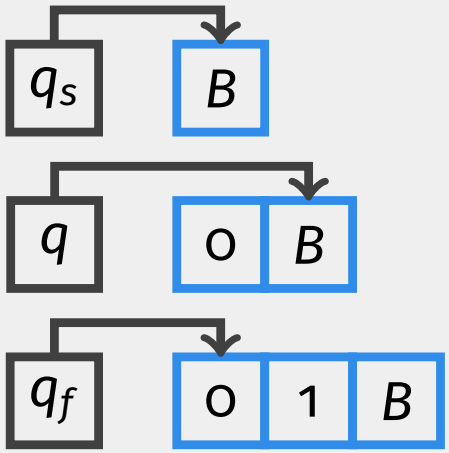
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



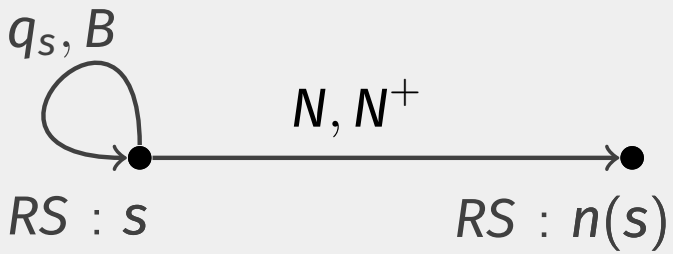
Apply $q_s(\ell_1, \ell_1) \wedge N^+(\ell_1, \ell_i) \rightarrow RS(\ell_i) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



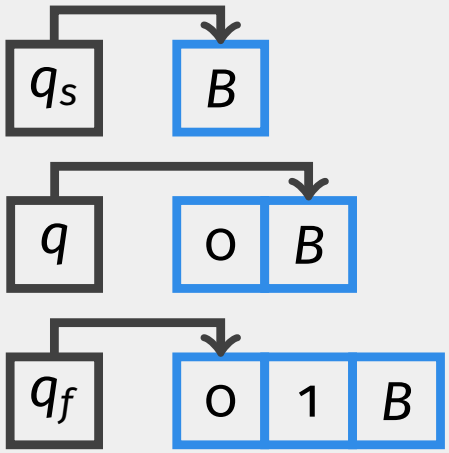
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



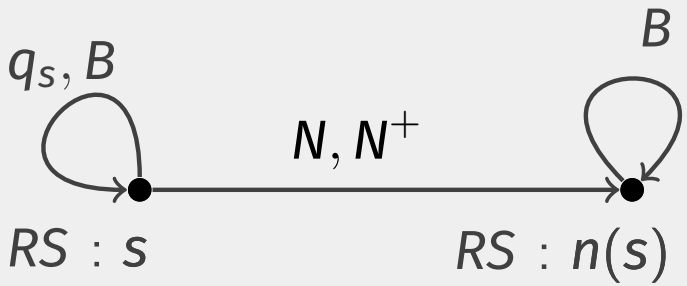
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



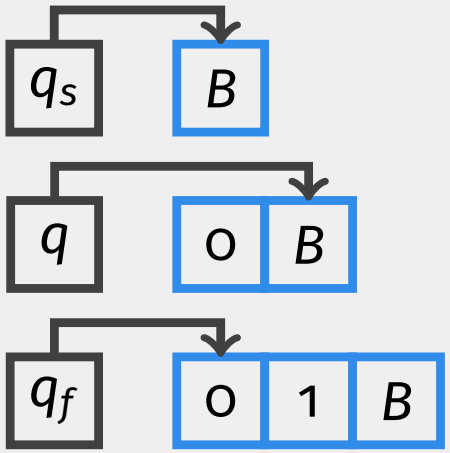
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



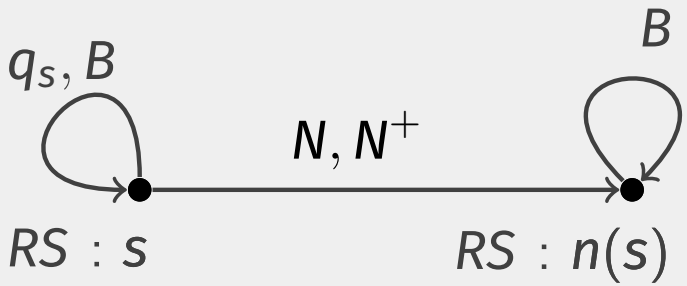
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



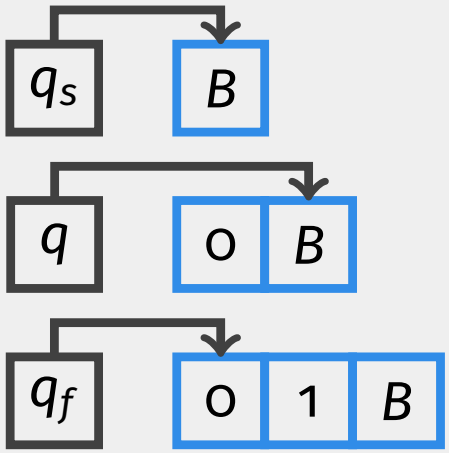
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



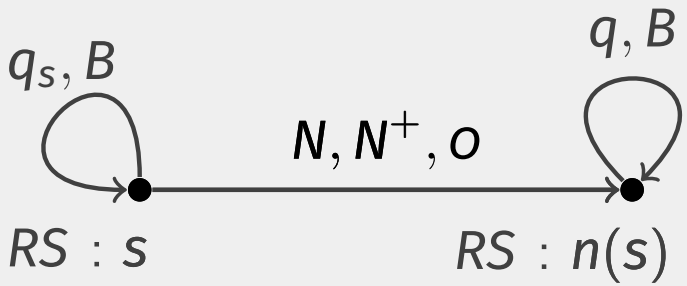
Since $M(q_s, B) = \langle q, 0, R \rangle$, we can apply
$q_s(\ell_i, \ell_i) \wedge B(\ell_i, \ell_i) \wedge N(\ell_i, \ell_{i+1}) \wedge RS(\ell_{i+1}) \rightarrow q(\ell_{i+1}, \ell_{i+1}) \wedge 0(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



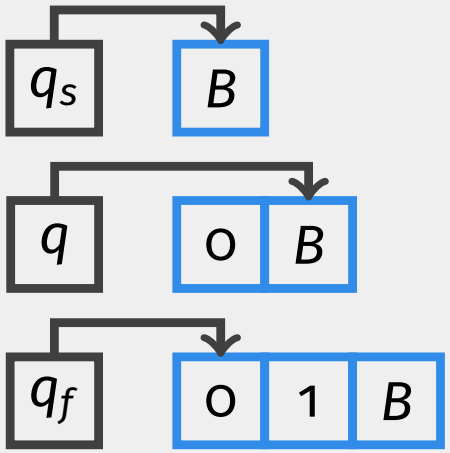
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



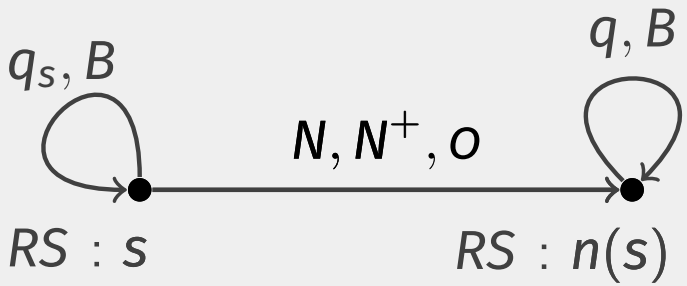
Since $M(q_s, B) = \langle q, \mathrm{o}, R \rangle$, we can apply
$q_s(\ell_i, \ell_i) \wedge B(\ell_i, \ell_i) \wedge N(\ell_i, \ell_{i+1}) \wedge RS(\ell_{i+1}) \rightarrow q(\ell_{i+1}, \ell_{i+1}) \wedge o(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



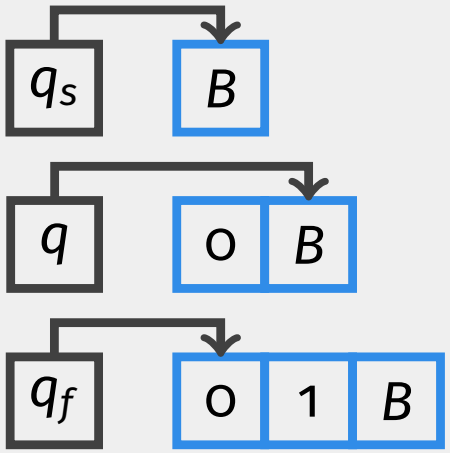
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



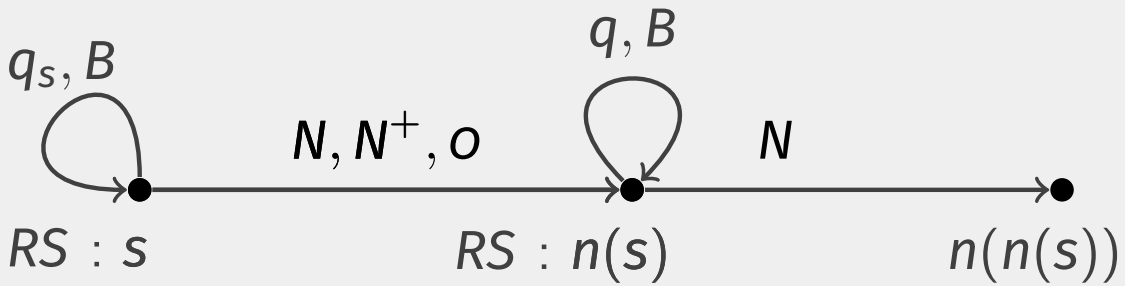
Since q is a non-final state in M, we can apply $q(p, c) \wedge RS(c) \rightarrow \exists n.N(c, n) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



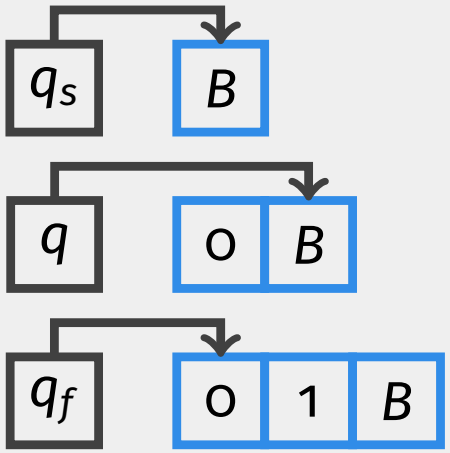
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



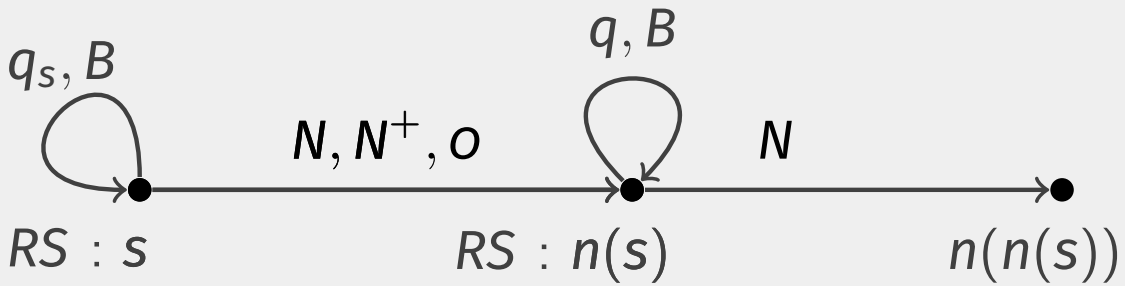
Since q is a non-final state in M, we can apply $q(p, c) \wedge RS(c) \rightarrow \exists n.N(c, n) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



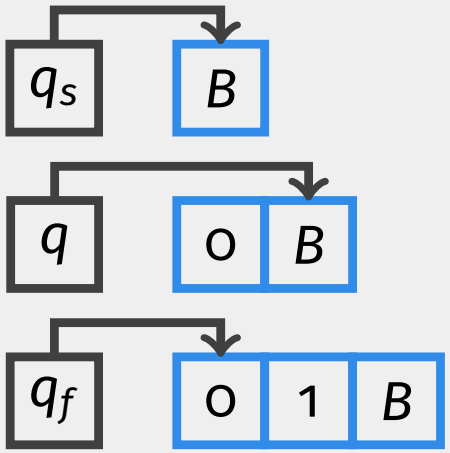
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



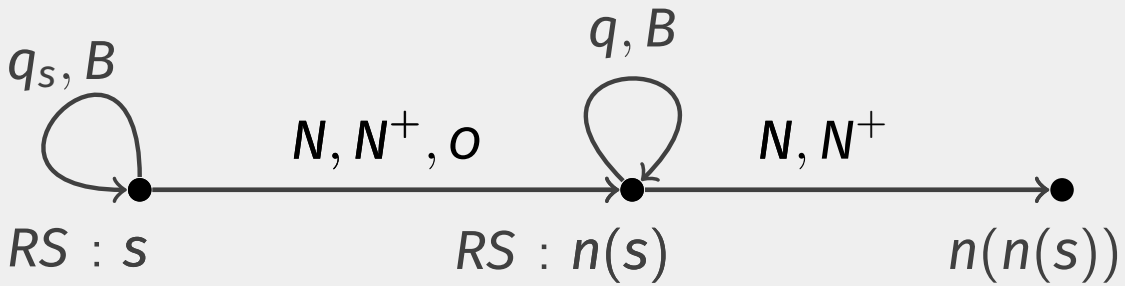
Apply $N(\ell_i, \ell_{i+1}) \rightarrow N^+(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



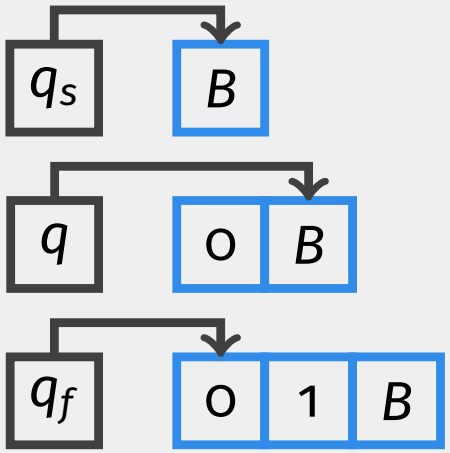
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



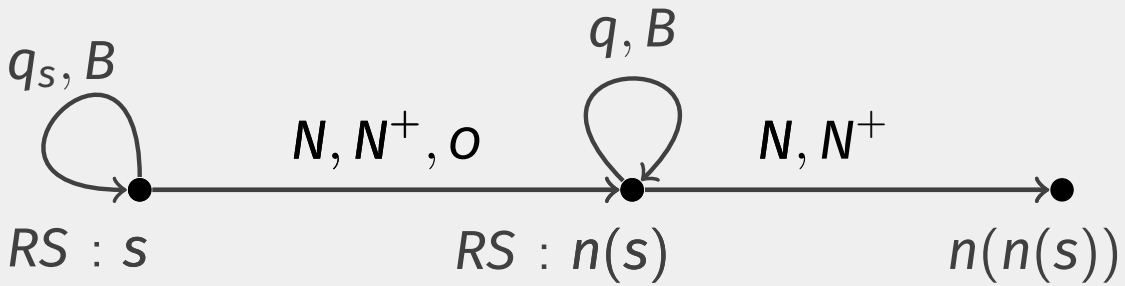
Apply $N(\ell_i, \ell_{i+1}) \rightarrow N^+(\ell_i, \ell_{i+1}) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



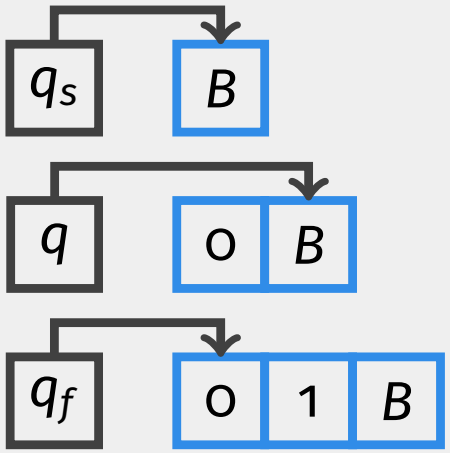
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



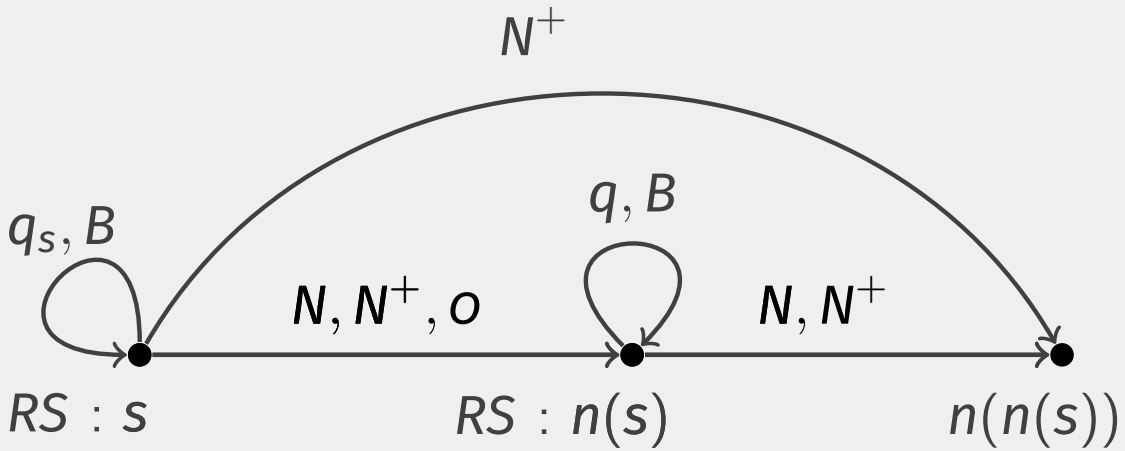
Apply $N(\ell_i, \ell_{i+1}) \wedge N^+(\ell_{i+1}, \ell_j) \rightarrow N^+(\ell_i, \ell_j) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



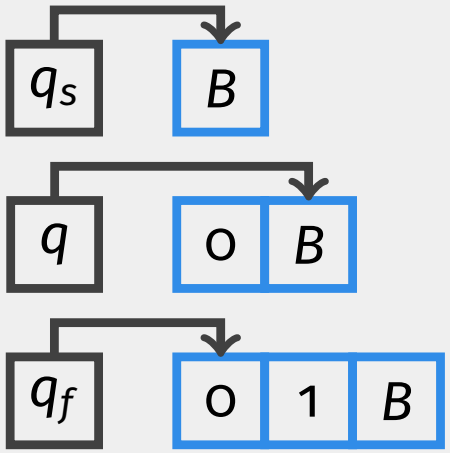
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



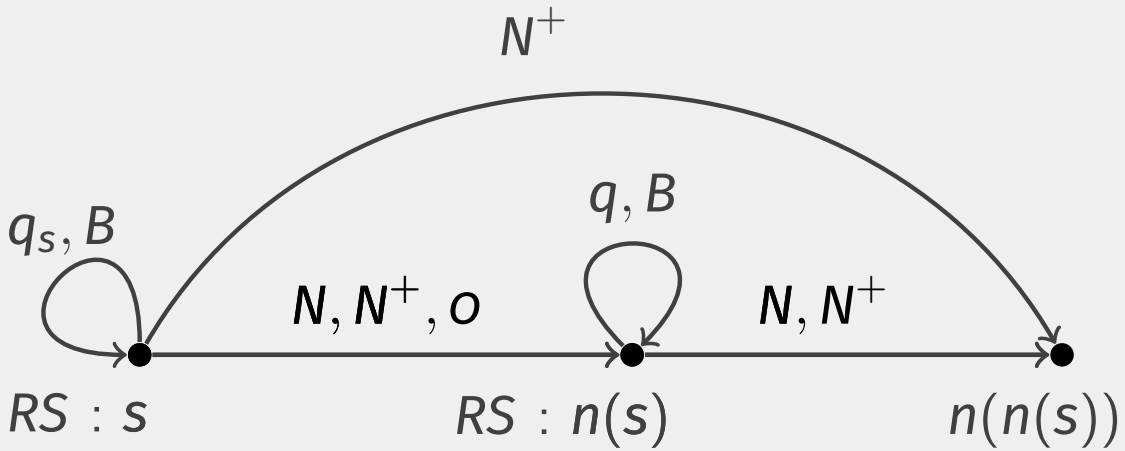
Apply $N(\ell_i, \ell_{i+1}) \wedge N^+(\ell_{i+1}, \ell_j) \rightarrow N^+(\ell_i, \ell_j) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



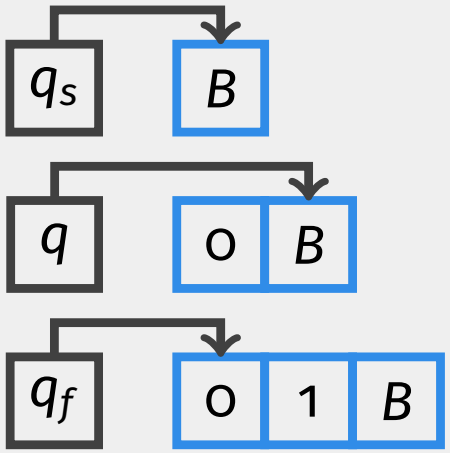
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



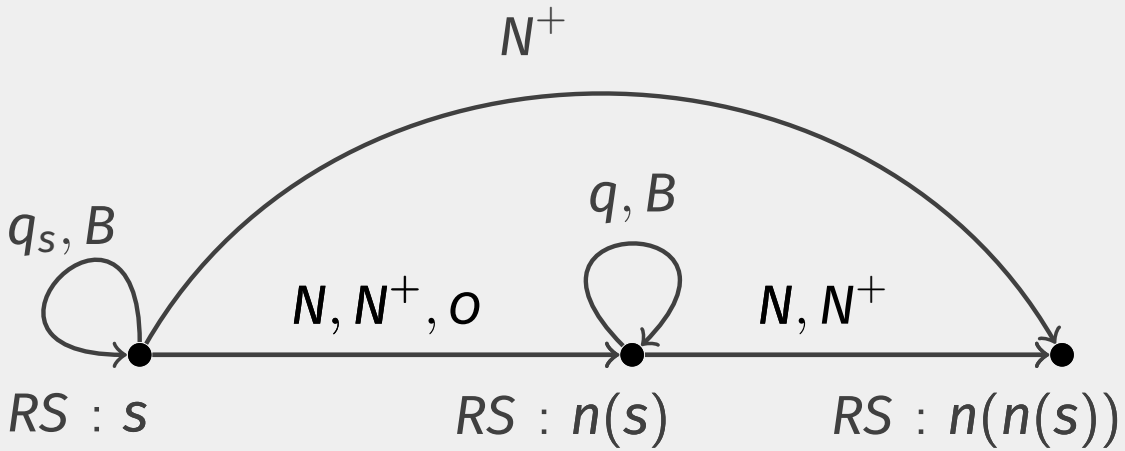
Apply $q_s(\ell_1, \ell_1) \wedge N^+(\ell_1, \ell_i) \rightarrow RS(\ell_i) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



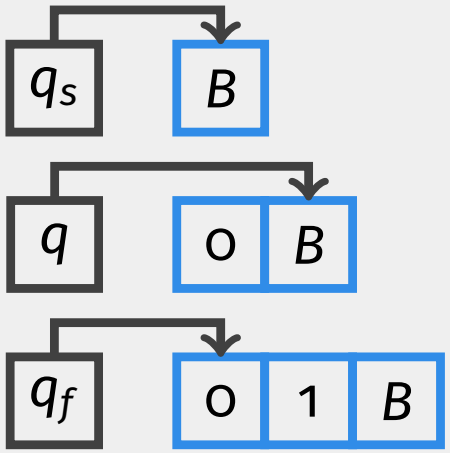
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



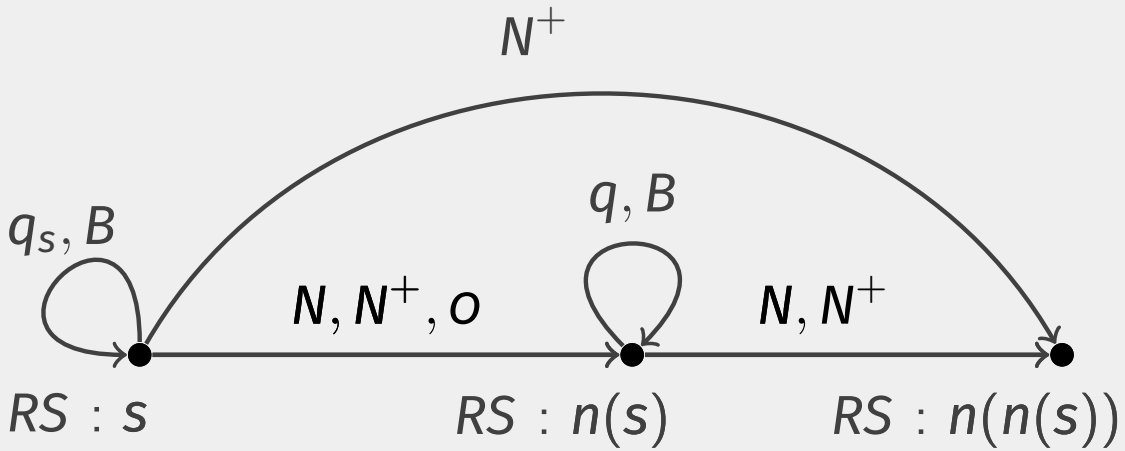
Apply $q_s(\ell_1, \ell_1) \wedge N^+(\ell_1, \ell_i) \rightarrow RS(\ell_i) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



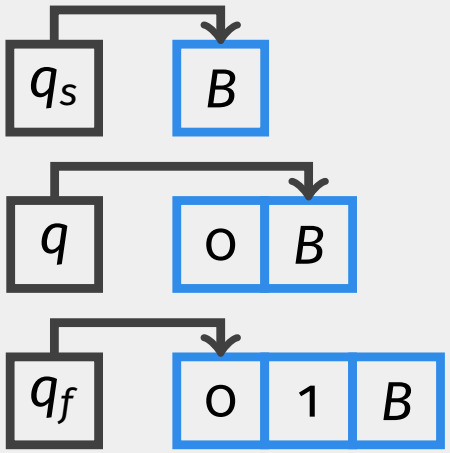
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



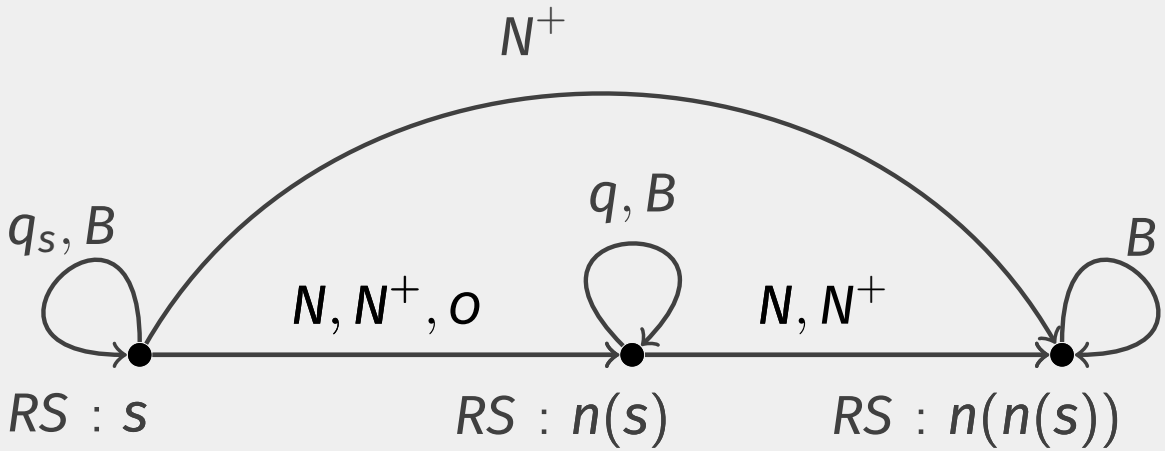
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



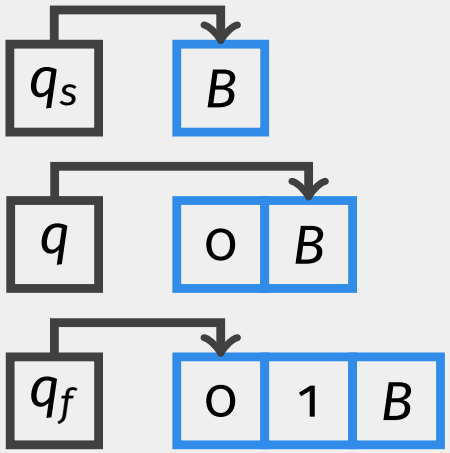
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



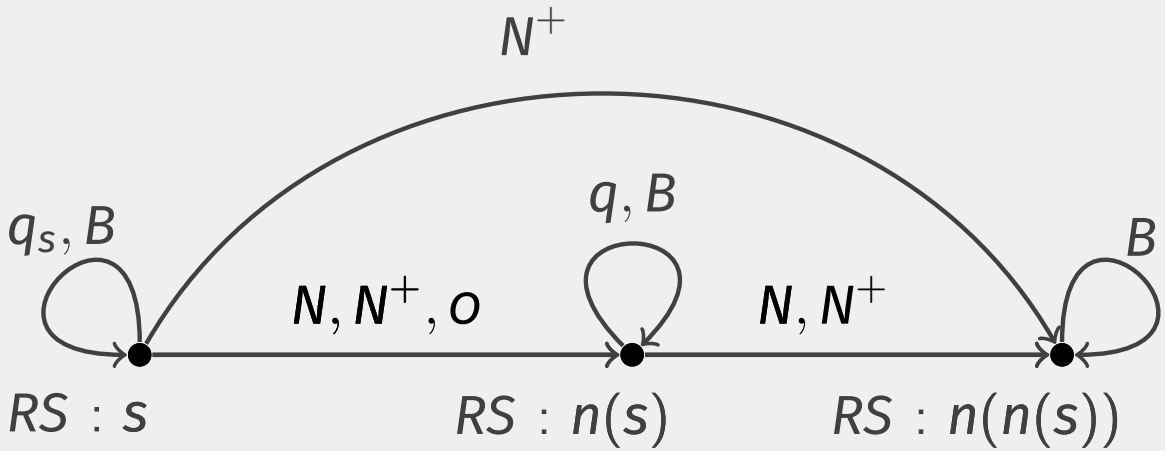
Apply $RS(x) \rightarrow B(x, x) \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

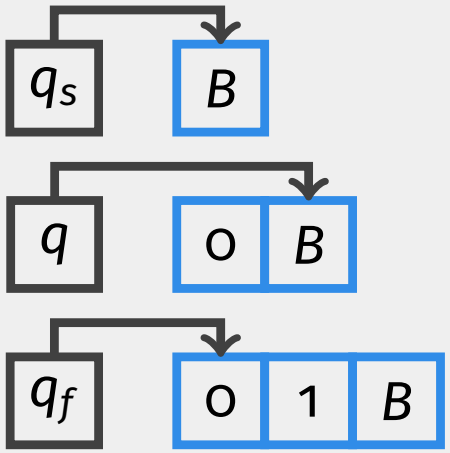


Since $M(q, B) = \langle q_f, 1, L \rangle$, we can apply

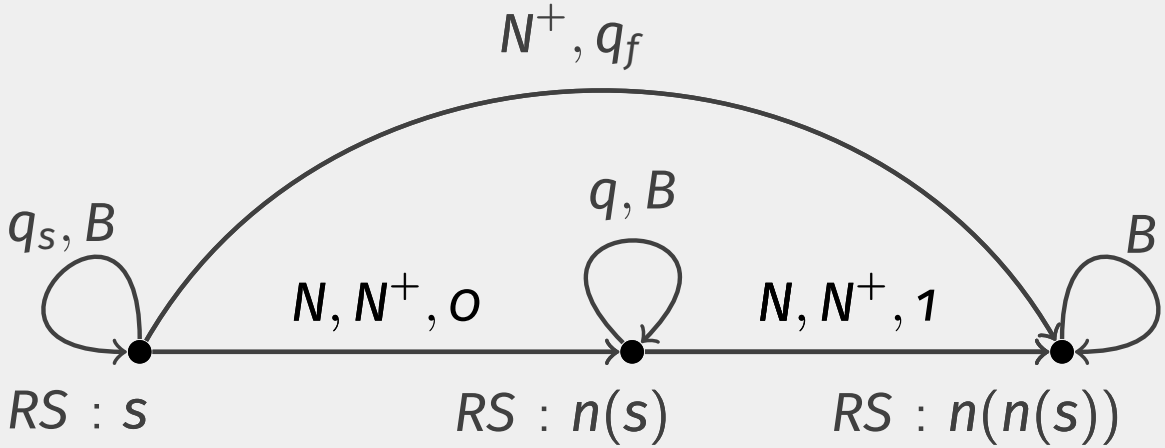
$$q(p_i, c_j) \wedge B(p_i, c_j) \wedge N(p_{i-1}, p_i) \wedge N(c_j, c_{j+1}) \wedge N^+(p_i, c_{j+1}) \wedge RS(p_i) \wedge RS(c_{j+1}) \rightarrow q_f(p_{i-1}, c_{j+1}) \wedge 1(p_i, c_{j+1}) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

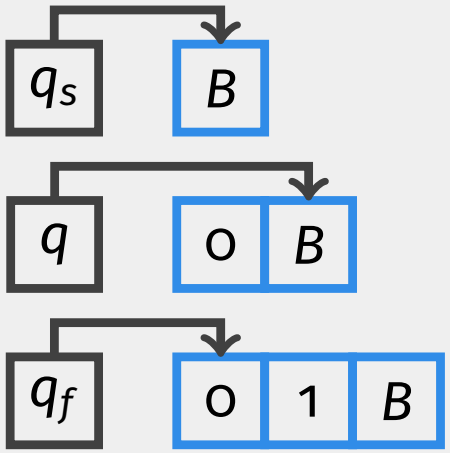


Since $M(q, B) = \langle q_f, 1, L \rangle$, we can apply

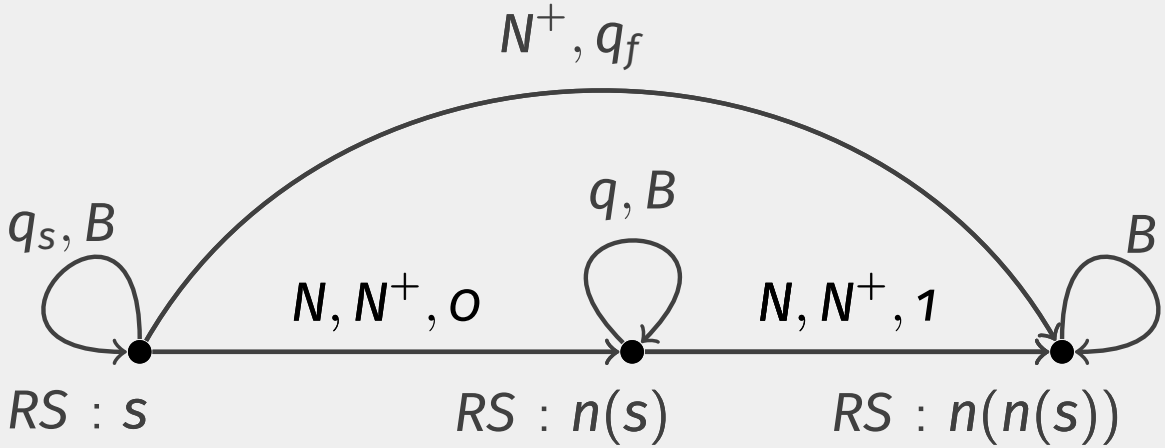
$$q(p_i, c_j) \wedge B(p_i, c_j) \wedge N(p_{i-1}, p_i) \wedge N(c_j, c_{j+1}) \wedge N^+(p_i, c_{j+1}) \wedge RS(p_i) \wedge RS(c_{j+1}) \rightarrow q_f(p_{i-1}, c_{j+1}) \wedge 1(p_i, c_{j+1}) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

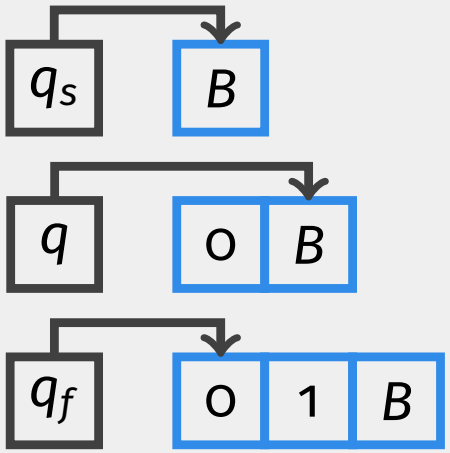


Since q is a non-final state in M, we can apply

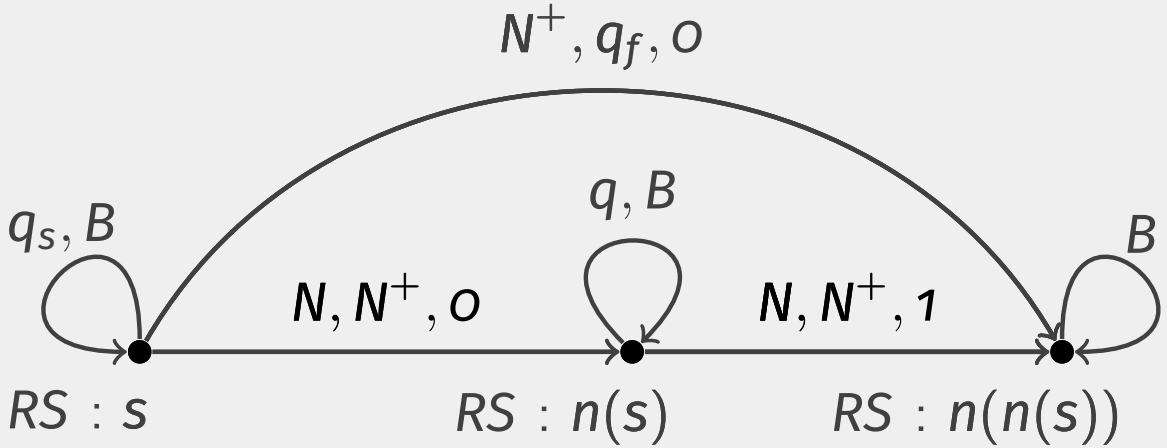
$$0(p_i, c_j) \wedge q(p_k, c_j) \wedge N(c_j, c_{j+1}) \wedge N^+(p_i, p_k) \wedge N^+(p_k, c_{j+1}) \wedge N^+(p_i, c_{j+1}) \wedge RS(p_i) \wedge RS(p_k) \wedge RS(c_{j+1}) \rightarrow 0(p_i, c_{j+1}) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:

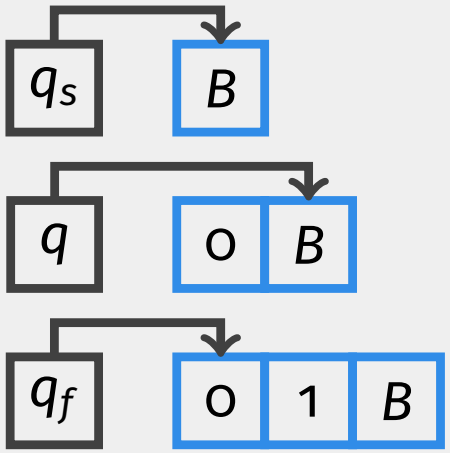


Since q is a non-final state in M, we can apply

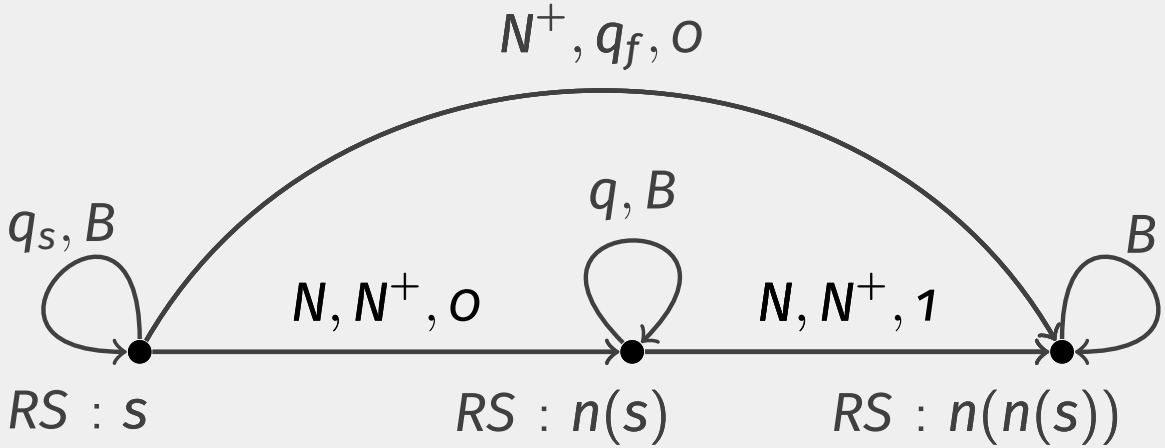
$$o(p_i, c_j) \wedge q(p_k, c_j) \wedge N(c_j, c_{j+1}) \wedge N^+(p_i, p_k) \wedge N^+(p_k, c_{j+1}) \wedge N^+(p_i, c_{j+1}) \wedge RS(p_i) \wedge RS(p_k) \wedge RS(c_{j+1}) \rightarrow o(p_i, c_{j+1}) \in \mathcal{R}_M$$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



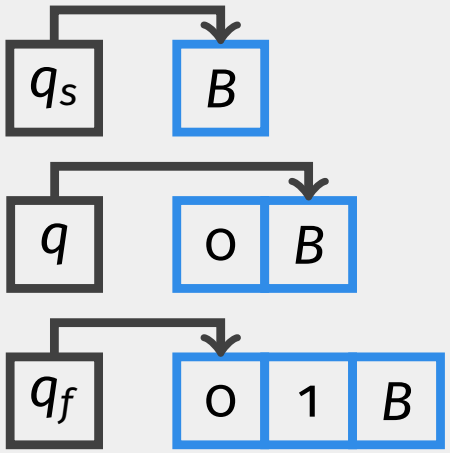
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



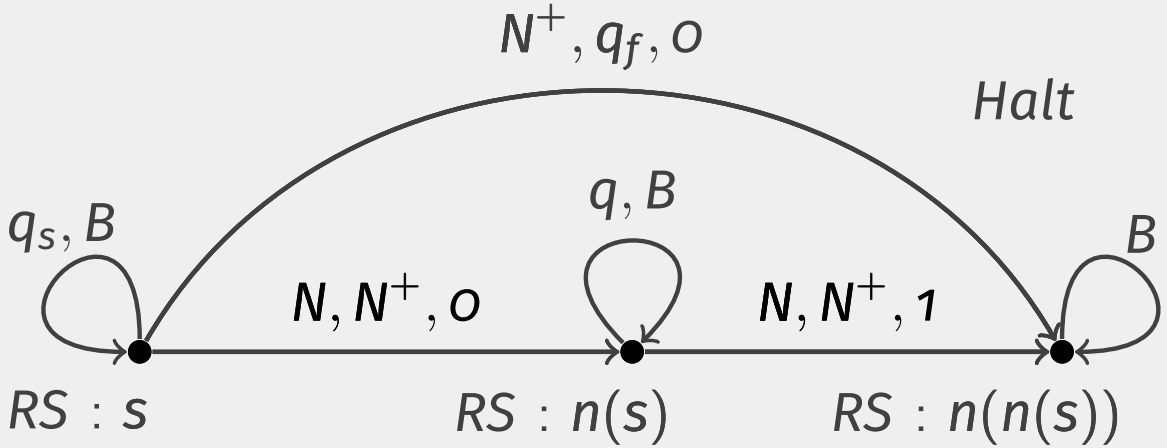
Apply $q_f(p, c) \rightarrow Halt \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



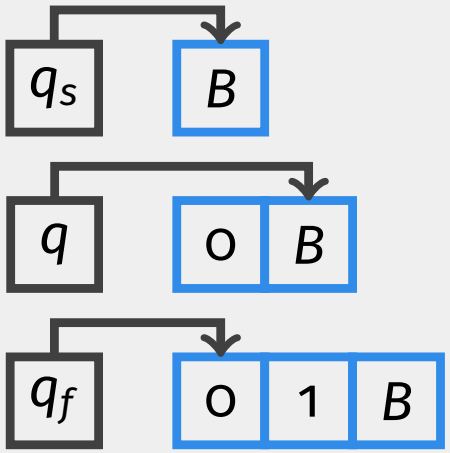
The chase of $\langle \mathcal{R}_M, \emptyset \rangle$:



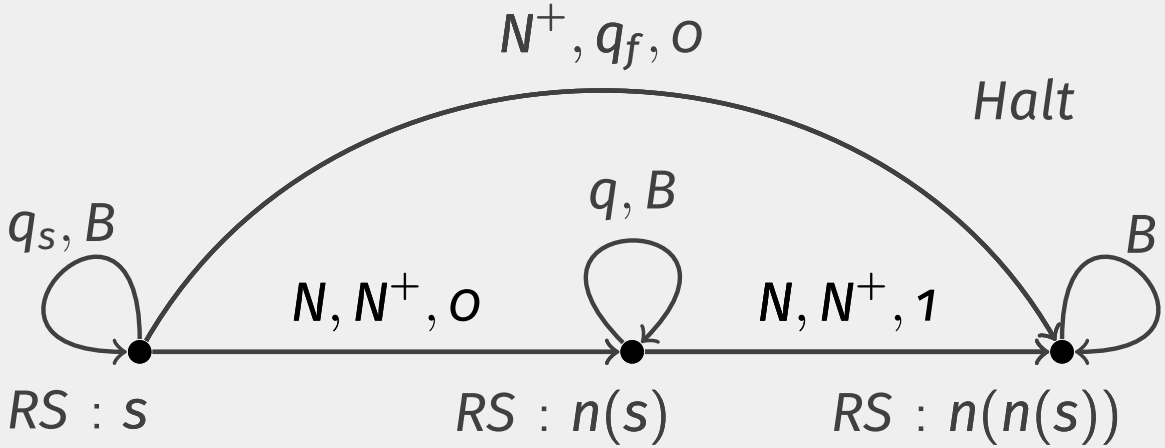
Apply $q_f(p, c) \rightarrow Halt \in \mathcal{R}_M$

Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:

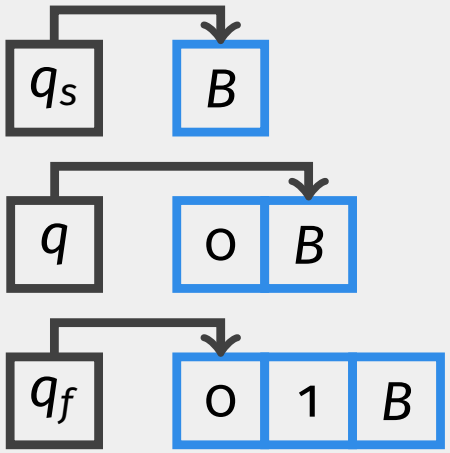


A universal model of $\langle \mathcal{R}_M, \emptyset \rangle$:

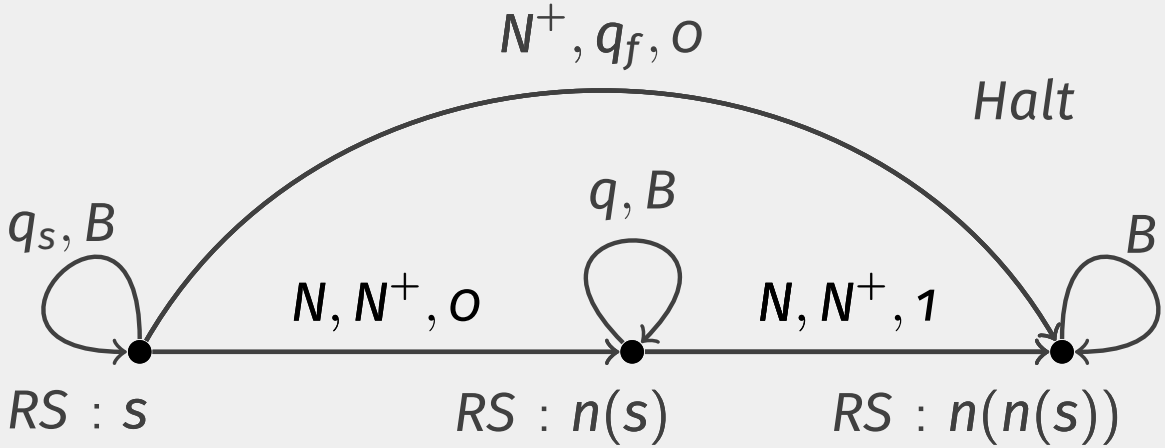


Emulating a Machine M with the KB $\langle \mathcal{R}_M, \emptyset \rangle$

The computation of M on ϵ:



A universal model of $\langle \mathcal{R}_M, \emptyset \rangle$:



Lemma 1. The machine M halts on the empty word if and only if the KB $\langle \mathcal{R}_M, \emptyset \rangle$ entails the fact *Halt*.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

- If M halts on ϵ, then $\langle \{\rightarrow \mathit{Halt}\}, \mathit{Halt} \rangle$ is a datalog-rewriting since $\langle \mathcal{R}_M, \emptyset \rangle \models \mathit{Halt}$ by Lemma 1.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

- If M halts on ϵ, then $\langle \{\rightarrow \mathit{Halt}\}, \mathit{Halt} \rangle$ is a datalog-rewriting since $\langle \mathcal{R}_M, \emptyset \rangle \models \mathit{Halt}$ by Lemma 1.
- If M does not halt on ϵ; then $\langle \mathcal{R}_M^{\forall}, \mathit{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \mathit{Halt} \rangle$, where $\mathcal{R}_M^{\forall}$ is the set of datalog rules in $\mathcal{R}_M$.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

- If M halts on ϵ, then $\langle \{\rightarrow \mathit{Halt}\}, \mathit{Halt} \rangle$ is a datalog-rewriting since $\langle \mathcal{R}_M, \emptyset \rangle \models \mathit{Halt}$ by Lemma 1.
- If M does not halt on ϵ; then $\langle \mathcal{R}_M^{\forall}, \mathit{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \mathit{Halt} \rangle$, where $\mathcal{R}_M^{\forall}$ is the set of datalog rules in $\mathcal{R}_M$. That is, for every fact set $\mathcal{F}$, we show that $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$ iff $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \models \mathit{Halt}$.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

- If M halts on ϵ, then $\langle \{\rightarrow \mathit{Halt}\}, \mathit{Halt} \rangle$ is a datalog-rewriting since $\langle \mathcal{R}_M, \emptyset \rangle \models \mathit{Halt}$ by Lemma 1.
- If M does not halt on ϵ; then $\langle \mathcal{R}_M^\forall, \mathit{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \mathit{Halt} \rangle$, where $\mathcal{R}_M^\forall$ is the set of datalog rules in $\mathcal{R}_M$. That is, for every fact set $\mathcal{F}$, we show that $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$ iff $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \models \mathit{Halt}$.
  - If $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$ since $\mathcal{R}_M^\forall \subseteq \mathcal{R}_M$.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 2

The rule query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is datalog-expressible.

Proof.

- If M halts on ϵ, then $\langle \{\rightarrow \mathit{Halt}\}, \mathit{Halt} \rangle$ is a datalog-rewriting since $\langle \mathcal{R}_M, \emptyset \rangle \models \mathit{Halt}$ by Lemma 1.
- If M does not halt on ϵ; then $\langle \mathcal{R}_M^\forall, \mathit{Halt} \rangle$ is a datalog-rewriting of $\langle \mathcal{R}_M, \mathit{Halt} \rangle$, where $\mathcal{R}_M^\forall$ is the set of datalog rules in $\mathcal{R}_M$. That is, for every fact set $\mathcal{F}$, we show that $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$ iff $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \models \mathit{Halt}$.
  - If $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$ since $\mathcal{R}_M^\forall \subseteq \mathcal{R}_M$.
  - To-do: If $\langle \mathcal{R}_M, \mathcal{F} \rangle \models \mathit{Halt}$, then $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \models \mathit{Halt}$.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
  - Because of $N^+(x,x) \to \mathit{Halt}$, $N(x,y) \to N^+(x,y)$, and $N(x,y) \wedge N^+(y,z) \to N^+(x,z)$; there are no N-cycles.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
  - Because of $N^+(x,x) \to \mathit{Halt}$, $N(x,y) \to N^+(x,y)$, and $N(x,y) \wedge N^+(y,z) \to N^+(x,z)$; there are no N-cycles.
  - Because of $N(x,y) \wedge N(x,z) \to y \approx z$ and $N(y,x) \wedge N(z,x) \to y \approx z$, constants have at most one N-successor/predecessor.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \rightarrow x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) \to \mathit{Halt}$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) \to \mathit{Halt}$$
$$q_f(p, c_i) \wedge N^+(c_i, c_j) \to \mathit{Halt}$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) \to \mathit{Halt}$$
$$q_f(p, c_i) \wedge N^+(c_i, c_j) \to \mathit{Halt}$$
$$q(p, c) \wedge q(p', c) \to p \approx p' \qquad \text{for every } q \in Q$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$
\begin{aligned}
N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) &\to \mathit{Halt} \\
q_f(p, c_i) \wedge N^+(c_i, c_j) &\to \mathit{Halt} \\
q(p, c) \wedge q(p', c) &\to p \approx p' && \text{for every } q \in Q \\
q(p, c) \wedge r(p', c) &\to \mathit{Halt} && \text{for every } q \neq r \text{ in } Q
\end{aligned}
$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$
\begin{aligned}
N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) &\to \mathit{Halt} && \\
q_f(p, c_i) \wedge N^+(c_i, c_j) &\to \mathit{Halt} && \\
q(p, c) \wedge q(p', c) &\to p \approx p' && \text{for every } q \in Q \\
q(p, c) \wedge r(p', c) &\to \mathit{Halt} && \text{for every } q \neq r \text{ in } Q \\
s(p, c) \wedge N^+(c, p) &\to \mathit{Halt} && \text{for every } s \in \{0, 1, B\} \cup Q
\end{aligned}
$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain Halt.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- Because of $q_s(x, y) \wedge q_s(z, w) \to x \approx w$, there is at most one *starting chain*. Moreover, because of the following rules, this starting chain encodes encodes a prefix of the computation of M on ϵ:

$$
\begin{aligned}
N^+(\ell_{-i}, \ell_1) \wedge q_s(\ell_1, \ell_1) &\to \mathit{Halt} && \\
q_f(p, c_i) \wedge N^+(c_i, c_j) &\to \mathit{Halt} && \\
q(p, c) \wedge q(p', c) &\to p \approx p' && \text{for every } q \in Q \\
q(p, c) \wedge r(p', c) &\to \mathit{Halt} && \text{for every } q \neq r \text{ in } Q \\
s(p, c) \wedge N^+(c, p) &\to \mathit{Halt} && \text{for every } s \in \{0, 1, B\} \cup Q \\
a(p, c) \wedge b(p, c) &\to \mathit{Halt} && \text{for every } a \neq b \text{ in } \{0, 1, B\}
\end{aligned}
$$

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.

The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ may look a bit like this:

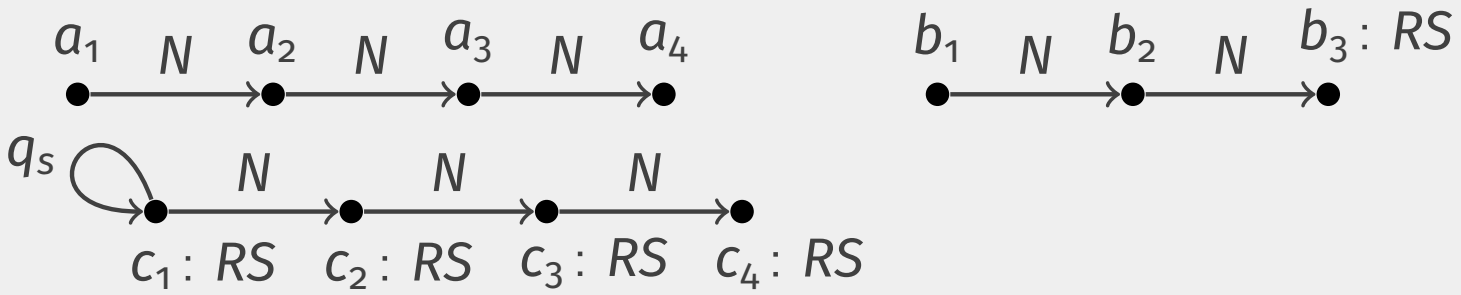
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ may look a bit like this:



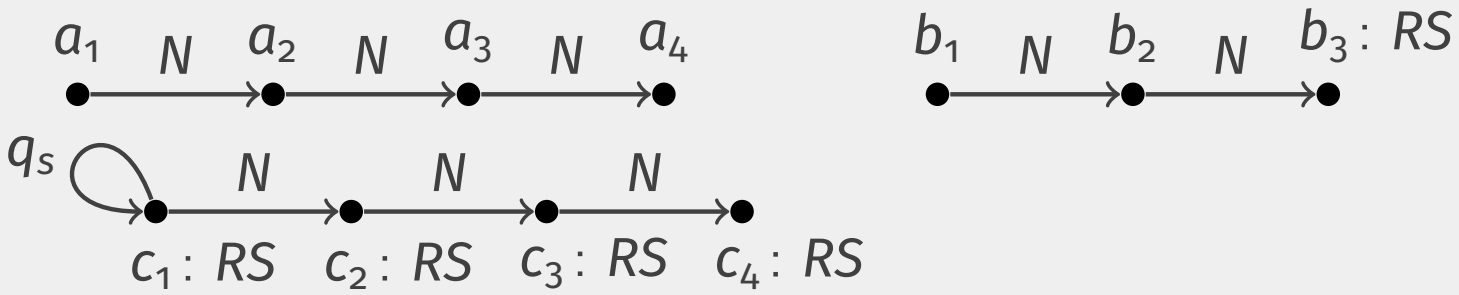
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ may look a bit like this:



We can extend the above into a model of $\langle \mathcal{R}_M, \mathcal{F} \rangle$ by:

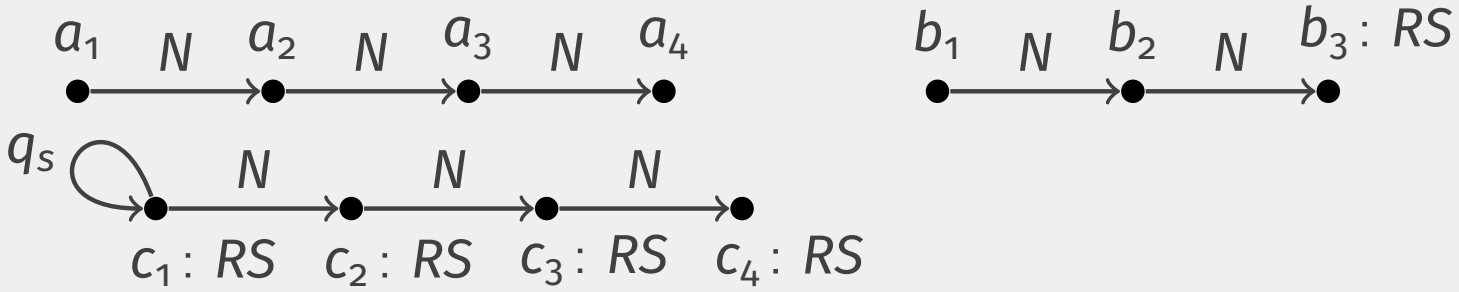
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^{\forall}, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^{\forall}, \mathcal{F})$ may look a bit like this:



We can extend the above into a model of $\langle \mathcal{R}_M, \mathcal{F} \rangle$ by:

- Adding $N(b_3, b_4)$ and $N^+(b_i, b_4)$ for a fresh b_4 and all $1 \leq i \leq 3$.

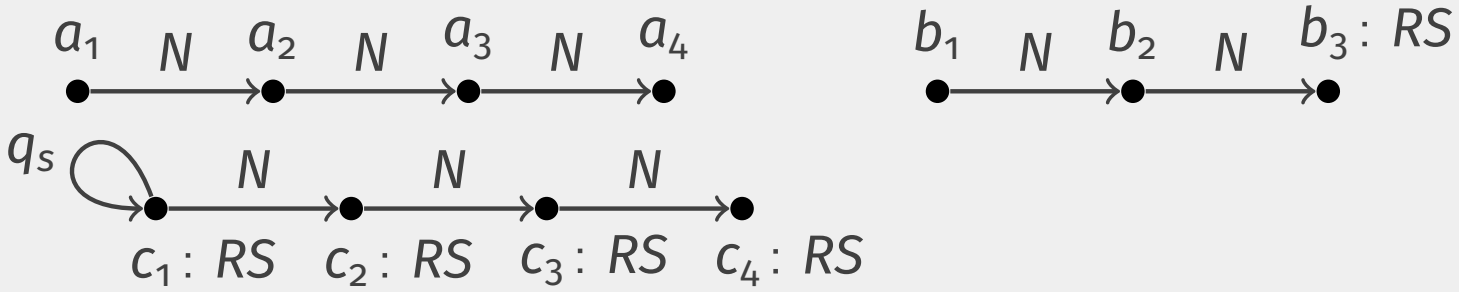
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ may look a bit like this:



We can extend the above into a model of $\langle \mathcal{R}_M, \mathcal{F} \rangle$ by:

- Adding $N(b_3, b_4)$ and $N^+(b_i, b_4)$ for a fresh b_4 and all $1 \leq i \leq 3$.
- Appending the chain in $\mathrm{Chase}(\mathcal{R}_M, \emptyset)$ to the starting chain.

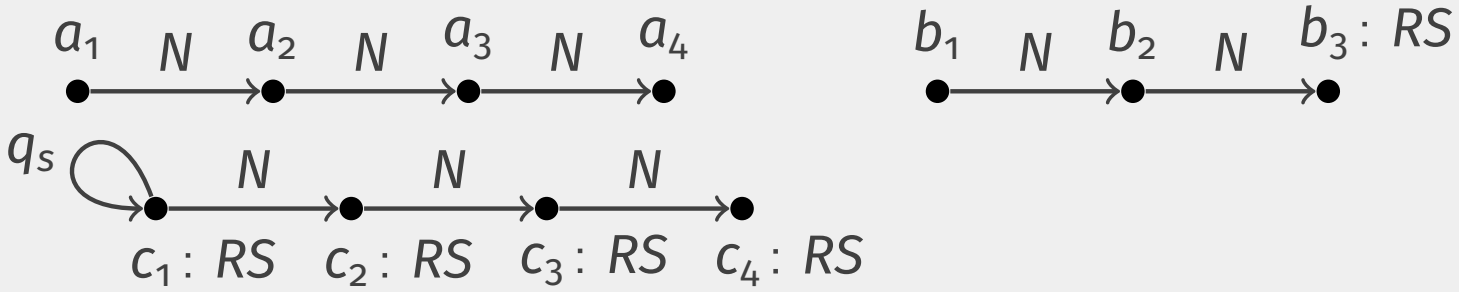
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ may look a bit like this:



We can extend the above into a model of $\langle \mathcal{R}_M, \mathcal{F} \rangle$ by:

- Adding $N(b_3, b_4)$ and $N^+(b_i, b_4)$ for a fresh b_4 and all $1 \leq i \leq 3$.
- Appending the chain in $\mathrm{Chase}(\mathcal{R}_M, \emptyset)$ to the starting chain.

If M does not halt on ϵ, the resulting fact set does not contain *Halt*.

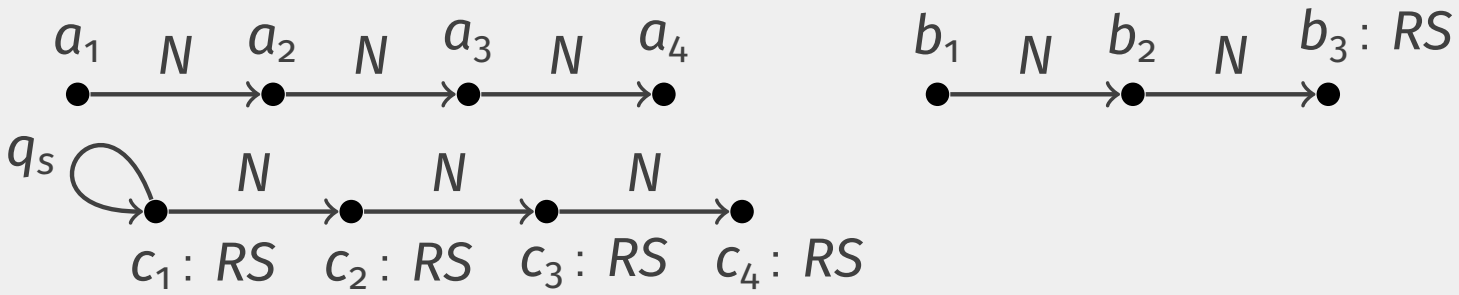
The Rule Query $\langle \mathcal{R}_M, \mathit{Halt} \rangle$ is Datalog-Expressible

Lemma 3.

If M does not halt on ϵ and $\langle \mathcal{R}_M^\forall, \mathcal{F} \rangle \not\models \mathit{Halt}$, then $\langle \mathcal{R}_M, \mathcal{F} \rangle \not\models \mathit{Halt}$.

Proof. Suppose that the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ does not contain *Halt*.

- The constants in $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ form a disjoint union of N-chains.
- There is at most one *starting chain*, which encodes a prefix of the computation of M on ϵ.
- Hence, the fact set $\mathrm{Chase}(\mathcal{R}_M^\forall, \mathcal{F})$ may look a bit like this:



We can extend the above into a model of $\langle \mathcal{R}_M, \mathcal{F} \rangle$ by:

- Adding $N(b_3, b_4)$ and $N^+(b_i, b_4)$ for a fresh b_4 and all $1 \leq i \leq 3$.
- Appending the chain in $\mathrm{Chase}(\mathcal{R}_M, \emptyset)$ to the starting chain.

If M does not halt on ϵ, the resulting fact set does not contain *Halt*. □

Future Work and Conclusions

Future Work

OpenProblem 1

- Is the class of all RPQ-expressible queries RPQ-rewritable?

Future Work

OpenProblem 1

- Is the class of all RPQ-expressible queries RPQ-rewritable?
- Is there a procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is RPQ-expressible?

Future Work

OpenProblem 1

- Is the class of all RPQ-expressible queries RPQ-rewritable?
- Is there a procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is RPQ-expressible?

Remark

Open Problem 1 can be instantiated for other languages such as monadic datalog, UCRPQs, languages based on CFGs...

Future Work

OpenProblem 1

- Is the class of all RPQ-expressible queries RPQ-rewritable?
- Is there a procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is RPQ-expressible?

Remark

Open Problem 1 can be instantiated for other languages such as monadic datalog, UCRPQs, languages based on CFGs...

Open Problem 2

Consider a KB $\langle \mathcal{R}, \mathcal{F} \rangle$, a (possibly non-Boolean) CQ φ, and a list $\vec{a}$ of constants occurring in $\mathcal{F}$.

Future Work

OpenProblem 1

- Is the class of all RPQ-expressible queries RPQ-rewritable?
- Is there a procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is RPQ-expressible?

Remark

Open Problem 1 can be instantiated for other languages such as monadic datalog, UCRPQs, languages based on CFGs...

Open Problem 2

Consider a KB $\langle \mathcal{R}, \mathcal{F} \rangle$, a (possibly non-Boolean) CQ φ, and a list $\vec{a}$ of constants occurring in $\mathcal{F}$.

- Is there a procedure to check if $\langle \mathcal{R}, \mathcal{F} \rangle \models \varphi[\vec{a}]$ that is sound, complete, and terminating if $\mathcal{R}$ is datalog-expressible?

Conclusions

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Conclusions

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Corollary

The class of all datalog-expressible rule queries is not datalog-rewritable.

Conclusions

Theorem

There is no procedure to check if a KB $\langle \mathcal{R}, \mathcal{F} \rangle$ entails a fact φ that is sound, complete, and terminates if $\langle \mathcal{R}, \varphi \rangle$ is datalog-expressible.

Corollary

The class of all datalog-expressible rule queries is not datalog-rewritable.

Thanks for your attention!

References I


Shqiponja Ahmetaj, Magdalena Ortiz, and Mantas Simkus, *Rewriting guarded existential rules into small datalog programs*, 21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria (Benny Kimelfeld and Yael Amsterdamer, eds.), LIPIcs, vol. 98, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 4:1–4:24.


Michael Benedikt, Maxime Buron, Stefano Germano, Kevin Kappelmann, and Boris Motik, *Rewriting the infinite chase*, Proc. VLDB Endow. **15** (2022), no. 11, 3045–3057.


Vince Bárány, Michael Benedikt, and Balder ten Cate, *Rewriting guarded negation queries*, Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings (Krishnendu Chatterjee and Jirí Sgall, eds.), Lecture Notes in Computer Science, vol. 8087, Springer, 2013, pp. 98–110.

References II


Gerald Berger, Georg Gottlob, Andreas Pieris, and Emanuel Sallinger, *The space-efficient core of vadalog*, ACM Trans. Database Syst. **47** (2022), no. 1, 1:1–1:46.


David Carral, Irina Dragoste, and Markus Krötzsch, *The combined approach to query answering in horn-alchoiq*, Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018 (Michael Thielscher, Francesca Toni, and Frank Wolter, eds.), AAAI Press, 2018, pp. 339–348.


David Carral, Larry González, and Patrick Koopmann, *From horn-sriq to datalog: A data-independent transformation that preserves assertion entailment*, The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in

References III

Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 2736–2743.

Georg Gottlob, Sebastian Rudolph, and Mantas Simkus, *Expressiveness of guarded existential rule languages*, Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014 (Richard Hull and Martin Grohe, eds.), ACM, 2014, pp. 27–38.

Georg Gottlob and Thomas Schwentick, *Rewriting ontological queries into small nonrecursive datalog programs*, Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, eds.), AAAI Press, 2012.

References IV

- Ullrich Hustadt, Boris Motik, and Ulrike Sattler, *Reasoning in description logics by a reduction to disjunctive datalog*, J. Autom. Reason. **39** (2007), no. 3, 351–384.
- Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo, *Sound, complete and minimal ucq-rewriting for existential rules*, Semantic Web **6** (2015), no. 5, 451–475.
- Bruno Marnette, *Resolution and Datalog rewriting under value invention and equality constraints*, CoRR **abs/1212.0254** (2012).
- Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler, *Type-elimination-based reasoning for the description logic shiqbs using decision diagrams and disjunctive datalog*, Log. Methods Comput. Sci. **8** (2012), no. 1.