A Journey to the Frontiers of Query Rewritability

Piotr Ostropolski-Nalewaja, Jerzy Marcinkowski, David Carral, and Sebastian Rudolph

June 7, 2022

The setting: existential rules

The setting: existential rules

Definition (by example)

The setting: existential rules

Definition (by example)

An $\exists$-*rule* or a *Tuple Generating Dependency* is an FO sentence of the following form:

The setting: existential rules

Definition (by example)

An $\exists$-*rule* or a *Tuple Generating Dependency* is an FO sentence of the following form:

$$\forall x\ Human(x) \rightarrow \exists y\ Mother(x, y)$$

The setting: existential rules

Definition (by example)

An $\exists$-*rule* or a *Tuple Generating Dependency* is an FO sentence of the following form:

$$\forall x\ Human(x) \rightarrow \exists y\ Mother(x, y)$$

$$\forall x, y\ Mother(x, y) \rightarrow Human(y)$$

The setting: existential rules

Definition (by example)

An ∃-*rule* or a *Tuple Generating Dependency* is an FO sentence of the following form:

$$\forall x \; Human(x) \rightarrow \exists y \; Mother(x, y)$$

$$\forall x, y \; Mother(x, y) \rightarrow Human(y)$$

Usually omitted

The setting: knowledge representation using ∃-rules

The setting: knowledge representation using ∃-rules

"Everyone has a mother"

$$Human(x) \rightarrow \exists y\ Mother(x, y)$$
$$Mother(x, y) \rightarrow Human(y)$$

The setting: knowledge representation using ∃-rules

"Everyone has a mother"

$$Human(x) \rightarrow \exists y \; Mother(x, y)$$
$$Mother(x, y) \rightarrow Human(y)$$

"E is transitive"

$$E(x, y), E(y, z) \rightarrow E(x, z)$$

The setting: knowledge representation using $\exists$-rules

"Everyone has a mother"

$$Human(x) \rightarrow \exists y\ Mother(x, y)$$
$$Mother(x, y) \rightarrow Human(y)$$

"E is transitive"

$$E(x, y), E(y, z) \rightarrow E(x, z)$$

Query entailment problem

The setting: knowledge representation using $\exists$-rules

"Everyone has a mother"	"E is transitive"
$Human(x) \to \exists y\ Mother(x,y)$ $Mother(x,y) \to Human(y)$	$E(x,y), E(y,z) \to E(x,z)$

Query entailment problem

Given a ruleset $\mathcal{R}$ and a database $\mathcal{D}$ we ask if some query $\mathcal{Q}$ holds in every model of $\mathcal{R}$ and $\mathcal{D}$?

The setting: knowledge representation using ∃-rules

"Everyone has a mother"

$$Human(x) \to \exists y\ Mother(x, y)$$
$$Mother(x, y) \to Human(y)$$

"E is transitive"

$$E(x, y), E(y, z) \to E(x, z)$$

Query entailment problem

Given a ruleset $\mathcal{R}$ and a database $\mathcal{D}$ we ask if some query $\mathcal{Q}$ holds in every model of $\mathcal{R}$ and $\mathcal{D}$?

To denote that a query holds in every model of $\mathcal{R}$ and $\mathcal{D}$ we write

$$\mathcal{R}, \mathcal{D} \models \mathcal{Q}.$$

The main character: BDD class of ∃-rules

The main character: BDD class of ∃-rules

BDD class definition (By FO-rewritings)

The main character: BDD class of $\exists$-rules

BDD class definition (By FO-rewritings)

A ruleset $\mathcal{R}$ admits *Bounded Derivation Depth property (is FO-rewritable)* if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds:

The main character: BDD class of ∃-rules

BDD class definition (By FO-rewritings)

A ruleset $\mathcal{R}$ admits *Bounded Derivation Depth property (is FO-rewritable)* if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds:

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

The main character: BDD class of ∃-rules

BDD class definition (By FO-rewritings)

A ruleset $\mathcal{R}$ admits *Bounded Derivation Depth property (is FO-rewritable)* if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds:

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

entail

The main character: BDD class of ∃-rules

BDD class definition (By FO-rewritings)

A ruleset $\mathcal{R}$ admits *Bounded Derivation Depth property (is FO-rewritable)* if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds:

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

entail holds

The BDD class

The BDD class

If a ruleset is BDD then the entailment problem for conjunctive queries is **decidable**.

The BDD class

If a ruleset is BDD then the entailment problem for conjunctive queries is **decidable**.

BDD is an **undecidable property**.

The BDD class

If a ruleset is BDD then the entailment problem for conjunctive queries is **decidable**.

BDD is an **undecidable property**. Thus, a lot of **decidable subclasses** of BDD were invented, such as:
Linear, Sticky, Sticky-Join, Backward Shy...

The BDD class

If a ruleset is BDD then the entailment problem for conjunctive queries is **decidable**.

BDD is an **undecidable property**. Thus, a lot of **decidable subclasses** of BDD were invented, such as:
Linear, Sticky, Sticky-Join, Backward Shy...

It is thought that **BDD is well understood**.

The BDD class

If a ruleset is BDD then the entailment problem for conjunctive queries is **decidable**.

BDD is an **undecidable property**. Thus, a lot of **decidable subclasses** of BDD were invented, such as:
Linear, Sticky, Sticky-Join, Backward Shy...

It is thought that **BDD is well understood**. Soon, however, we will see that **there is a lot more to learn about it.**

Our contributions

Our contributions

We identify a class of “local theories” subsuming every known decidable subclass of BDD!

Our contributions

We identify a class of "local theories" subsuming every known decidable subclass of BDD!

We show that this class is strictly contained in BDD.

Our contributions

We identify a class of "local theories" subsuming every known decidable subclass of BDD!

We show that this class is strictly contained in BDD.

And we show that the FUS/FES conjecture (BDD $\cap$ Core Terminating = Uniform BDD) holds for this class.

Our contributions

We identify a class of “local theories” subsuming every known decidable subclass of BDD!

We show that this class is strictly contained in BDD.

And we show that the FUS/FES conjecture
(BDD $\cap$ Core Terminating = Uniform BDD)
holds for this class.
(not a part of this talk)

The chase

The chase

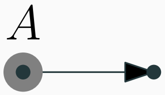
$$E(x, y) \to \exists z\ E(y, z)$$

The chase

$E(x, y) \to \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \to E(x, z)$

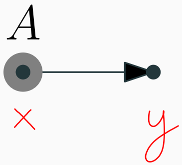
The chase

$E(x, y) \to \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \to E(x, z)$



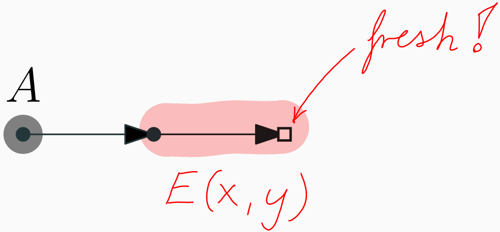
The chase

$$E(x, y) \to \exists z\ E(y, z)$$
$$A(x), E(x, y), E(y, z) \to E(x, z)$$



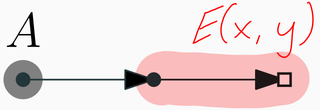
The chase

$E(x, y) \rightarrow \exists z \; E(y, z)$
$A(x), E(x, y), E(y, z) \rightarrow E(x, z)$



The chase

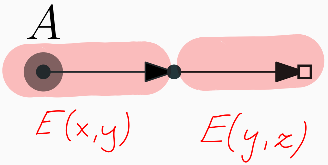
$E(x, y) \rightarrow \exists z \; E(y, z)$
$A(x), E(x, y), E(y, z) \rightarrow E(x, z)$



The chase

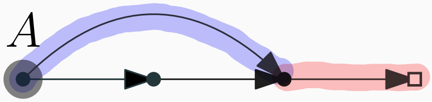
$E(x, y) \to \exists z\, E(y, z)$

$A(x), E(x, y), E(y, z) \to E(x, z)$



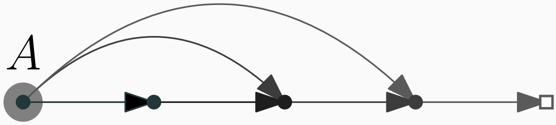
The chase

$E(x, y) \rightarrow \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \rightarrow E(x, z)$



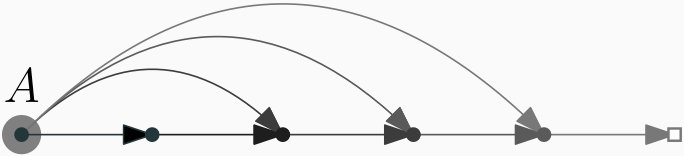
The chase

$E(x, y) \to \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \to E(x, z)$



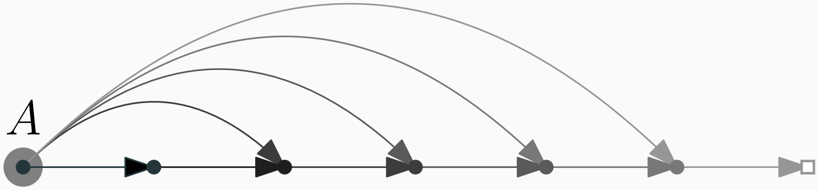
The chase

$E(x, y) \to \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \to E(x, z)$



The chase

$E(x, y) \rightarrow \exists z\ E(y, z)$
$A(x), E(x, y), E(y, z) \rightarrow E(x, z)$



The main character: BDD class of ∃-rules

The main character: BDD class of ∃-rules

BDD definition (by chase)

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

The main character: BDD class of $\exists$-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \;\; \exists k \;\; \forall \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \;\; \exists k \;\; \forall \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

Query

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \;\; \exists k \;\; \forall \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

Query $\quad$ $\in \mathbb{N}$

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \;\; \exists k \;\; \forall \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

Query ($\mathcal{Q}$) — $k \in \mathbb{N}$ — Database ($\mathcal{D}$)

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \ \exists k \ \forall \mathcal{D} \ \ \mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff Chase_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

entail

Query

$\in \mathbb{N}$

Database

The main character: BDD class of ∃-rules

BDD definition (by chase)

A ruleset $\mathcal{R}$ is BDD if:

$$\forall \mathcal{Q} \;\; \exists k \;\; \forall \mathcal{D} \;\; \underbrace{\mathcal{D}, \mathcal{R} \models \mathcal{Q}}_{\text{entail}} \iff Chase_k(\mathcal{R}, \mathcal{D}) \models \mathcal{Q}$$

entail

Query

$\in \mathbb{N}$

Database

k steps

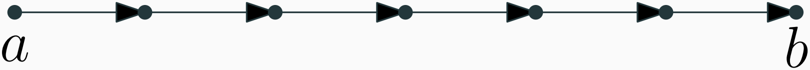
Transitivity is not BDD

Transitivity is not BDD

$E(x, y), E(y, z) \rightarrow E(x, z)$

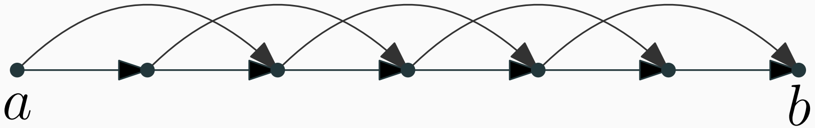
Transitivity is not BDD

$E(x, y), E(y, z) \rightarrow E(x, z)$



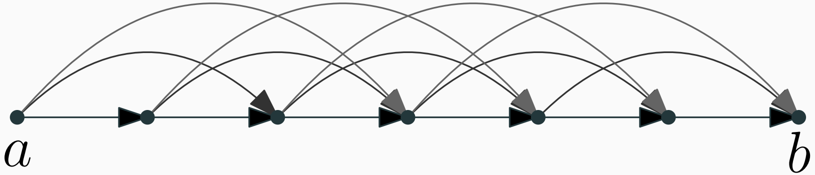
Transitivity is not BDD

$E(x, y), E(y, z) \to E(x, z)$



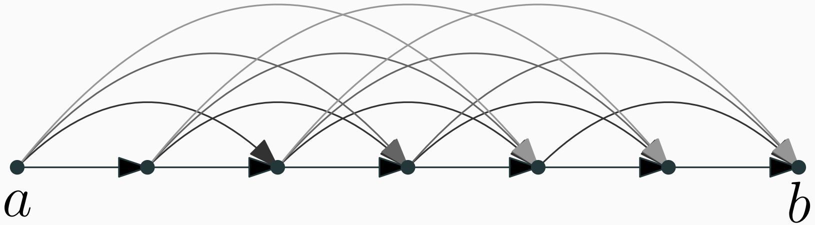
Transitivity is not BDD

$E(x, y), E(y, z) \to E(x, z)$



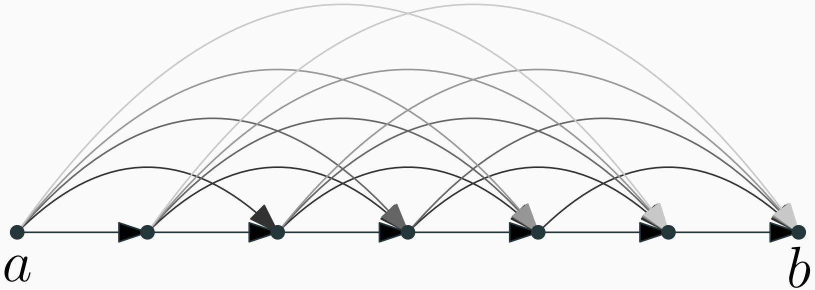
Transitivity is not BDD

$E(x, y), E(y, z) \to E(x, z)$



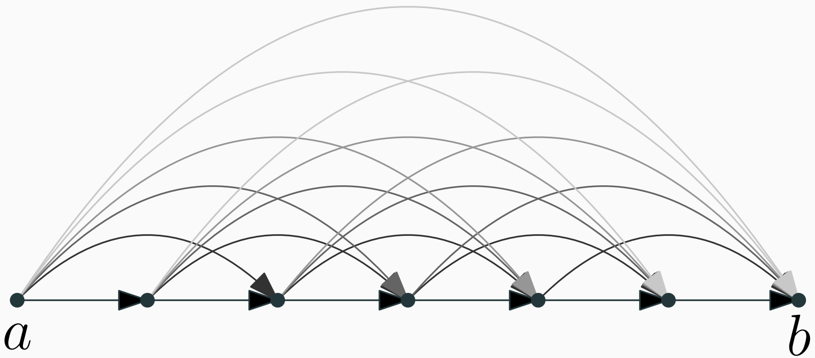
Transitivity is not BDD

$E(x, y), E(y, z) \rightarrow E(x, z)$



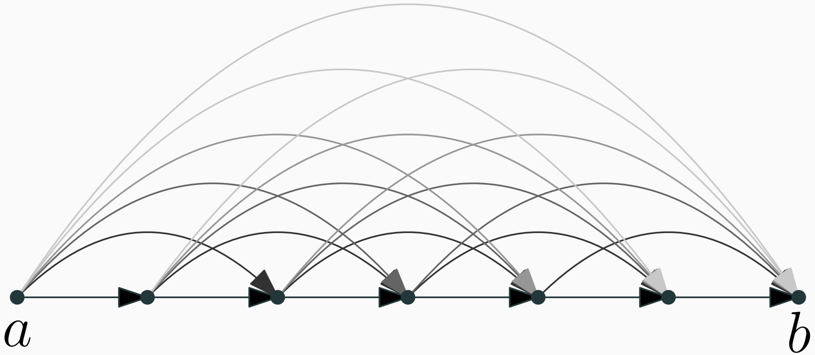
Transitivity is not BDD

$E(x, y), E(y, z) \to E(x, z)$



Transitivity is not BDD

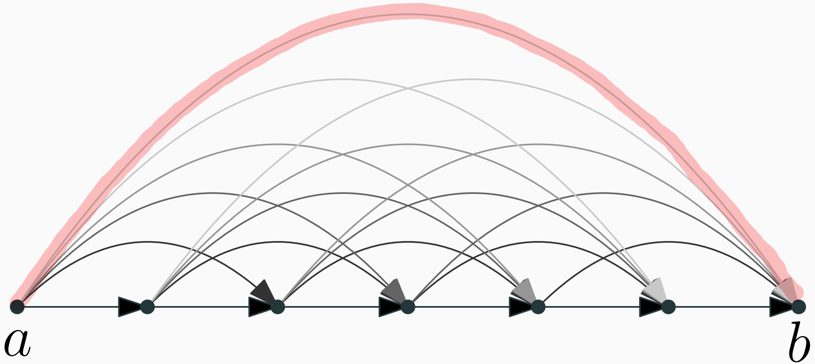
$E(x, y), E(y, z) \rightarrow E(x, z)$



$(\neg \text{ BDD}) \;\; \exists \mathcal{Q} \;\; \forall k \;\; \exists \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \wedge \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \not\models \mathcal{Q}$

Transitivity is not BDD

$E(x, y), E(y, z) \rightarrow E(x, z)$



(¬ BDD) $\exists \mathcal{Q} \ \forall k \ \exists \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \wedge Chase_k(\mathcal{R}, \mathcal{D}) \not\models \mathcal{Q}$

$Q = E(a, b)$

Reachability is not BDD

Reachability is not BDD

$A(x), E(x,y) \rightarrow A(y)$

Reachability is not BDD

$A(x), E(x, y) \rightarrow A(y)$

Reachability is not BDD

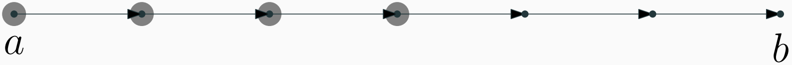
$A(x), E(x, y) \rightarrow A(y)$

Reachability is not BDD

$A(x), E(x, y) \rightarrow A(y)$

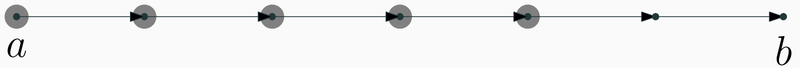
Reachability is not BDD

$A(x), E(x, y) \rightarrow A(y)$



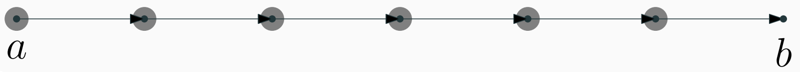
Reachability is not BDD

$A(x), E(x,y) \to A(y)$



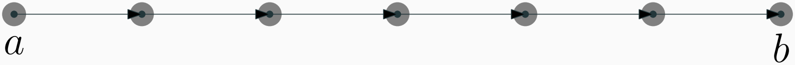
Reachability is not BDD

$A(x), E(x, y) \to A(y)$



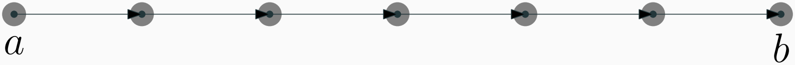
Reachability is not BDD

$A(x), E(x, y) \rightarrow A(y)$



Reachability is not BDD

$A(x), E(x, y) \rightarrow A(y)$

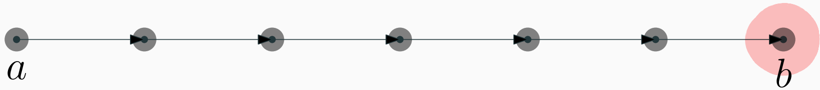


$(\neg \text{ BDD}) \;\; \exists \mathcal{Q} \;\; \forall k \;\; \exists \mathcal{D} \quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \wedge Chase_k(\mathcal{R}, \mathcal{D}) \not\models \mathcal{Q}$

Reachability is not BDD

$A(x), E(x,y) \rightarrow A(y)$

$Q = A(b)$



$(\neg\ \text{BDD})\ \ \exists \mathcal{Q}\ \ \forall k\ \ \exists \mathcal{D}\quad \mathcal{D}, \mathcal{R} \models \mathcal{Q} \wedge \mathit{Chase}_k(\mathcal{R}, \mathcal{D}) \not\models \mathcal{Q}$

Class of local theories

Class of local theories

Definition of the class of local theories

Class of local theories

Definition of the class of local theories

We say that a ruleset $\mathcal{R}$ is local if every atom of the chase can be derived only from a constant number of atoms of the database.

Class of local theories

Definition of the class of local theories

We say that a ruleset $\mathcal{R}$ is local if every atom of the chase can be derived only from a constant number of atoms of the database.

Every local ruleset is BDD!

Class of local theories

Definition of the class of local theories

We say that a ruleset $\mathcal{R}$ is local if every atom of the chase can be derived only from a constant number of atoms of the database.

Every local ruleset is BDD!

Moreover, it contains **every** known decidable subclass of BDD.

Class of local theories

Definition of the class of local theories

We say that a ruleset $\mathcal{R}$ is local if every atom of the chase can be derived only from a constant number of atoms of the database.

Every local ruleset is BDD!

Moreover, it contains **every** known decidable subclass of BDD.

almost

BDD is not local!

BDD is not local!

$E(x, y) \to \exists x', y' \; Box(x, y, x', y')$

BDD is not local!

$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$

BDD is not local!

$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$

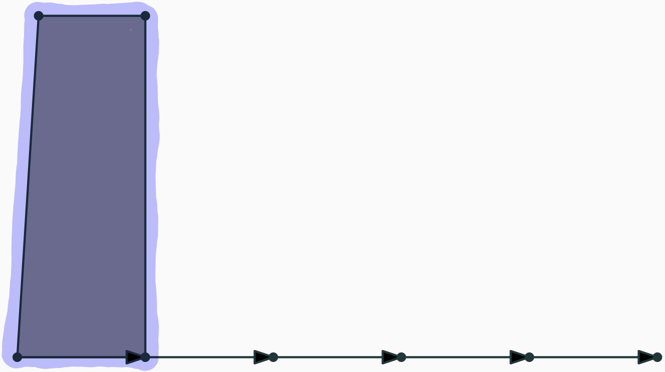
$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$

BDD is not local!

$$E(x,y) \rightarrow \exists x', y' \; Box(x,y,x',y')$$
$$Box(x,y,x',y'), E(x,y), E(y,z) \rightarrow \exists z' \; Box(y,z,y',z')$$

BDD is not local!

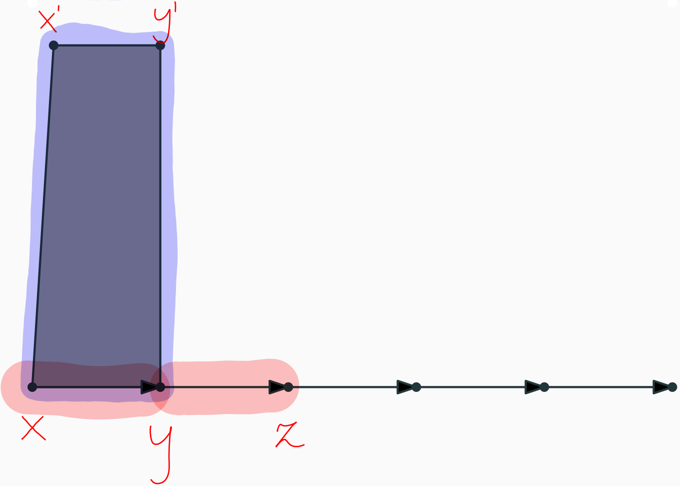
$$E(x,y) \rightarrow \exists x', y' \; Box(x,y,x',y')$$
$$Box(x,y,x',y'), E(x,y), E(y,z) \rightarrow \exists z' \; Box(y,z,y',z')$$



BDD is not local!

$E(x, y) \to \exists x', y' \; Box(x, y, x', y')$

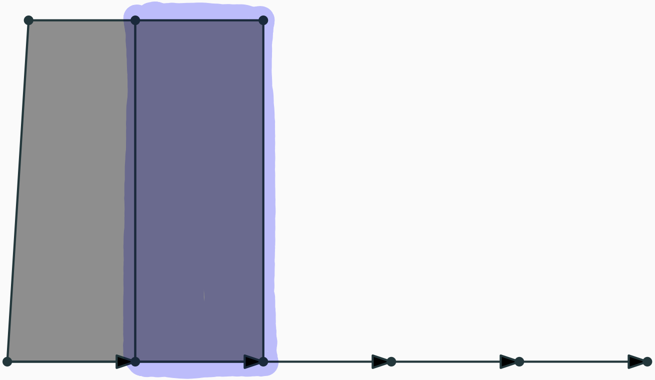
$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z' \; Box(y, z, y', z')$



BDD is not local!

$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$

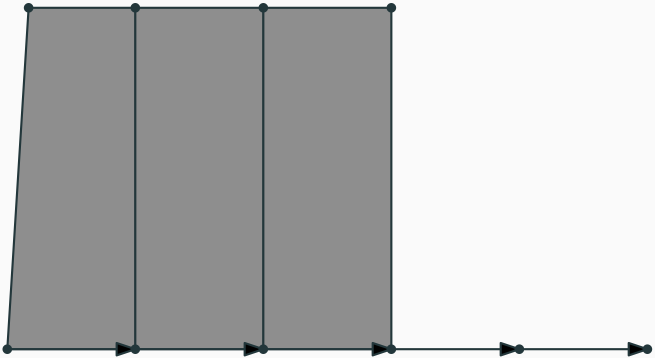
$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$



BDD is not local!

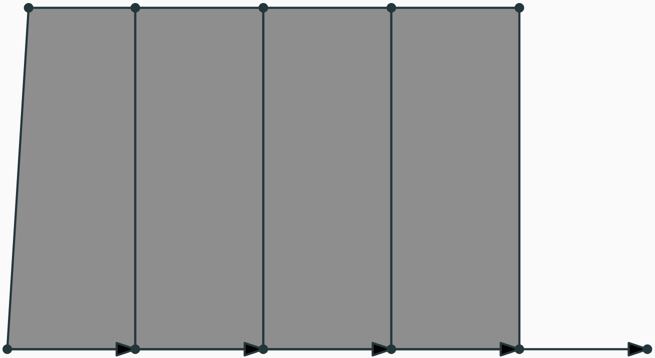
$E(x, y) \to \exists x', y' \ Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z' \ Box(y, z, y', z')$



BDD is not local!

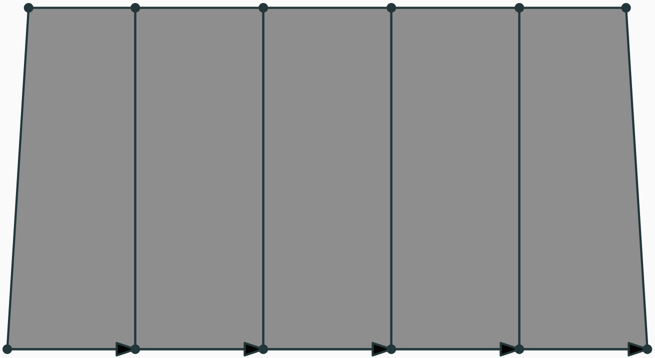
$$E(x, y) \to \exists x', y' \; Box(x, y, x', y')$$
$$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z' \; Box(y, z, y', z')$$



BDD is not local!

$$E(x,y) \to \exists x', y'\ Box(x,y,x',y')$$

$$Box(x,y,x',y'), E(x,y), E(y,z) \to \exists z'\ Box(y,z,y',z')$$



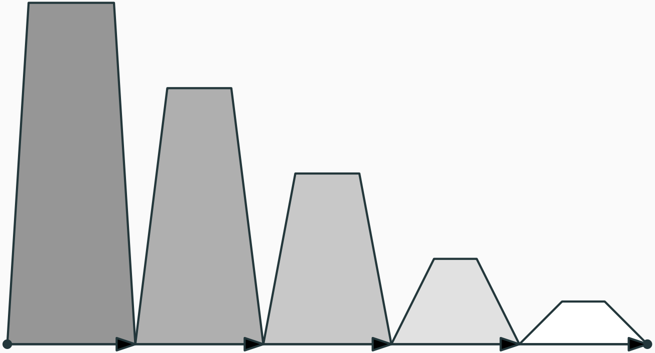
BDD is not local!

$E(x, y) \to \exists x', y' \; Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z' \; Box(y, z, y', z')$

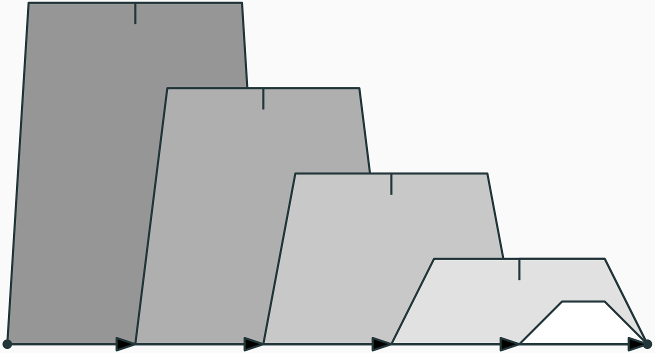
BDD is not local!

$$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$$
$$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$$



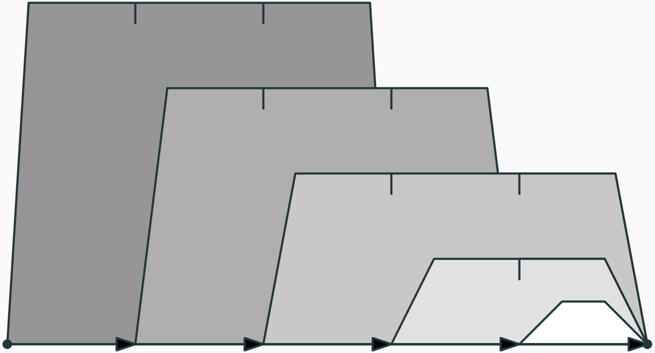
BDD is not local!

$$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$$
$$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$$



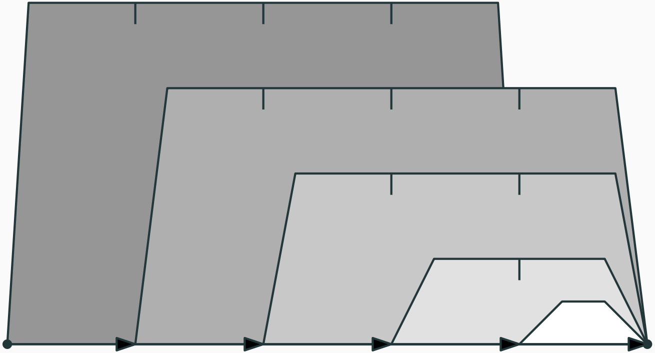
BDD is not local!

$E(x, y) \rightarrow \exists x', y'\ Box(x, y, x', y')$
$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z'\ Box(y, z, y', z')$



BDD is not local!

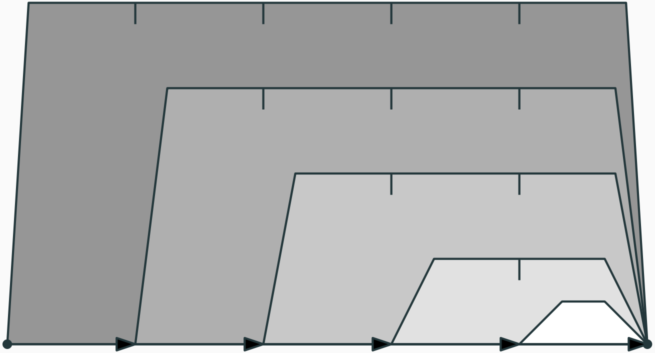
$$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$$
$$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$$



BDD is not local!

$E(x, y) \to \exists x', y' \; Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z' \; Box(y, z, y', z')$



BDD is not local!

$E(x,y) \rightarrow \exists x', y' \; Box(x,y,x',y')$

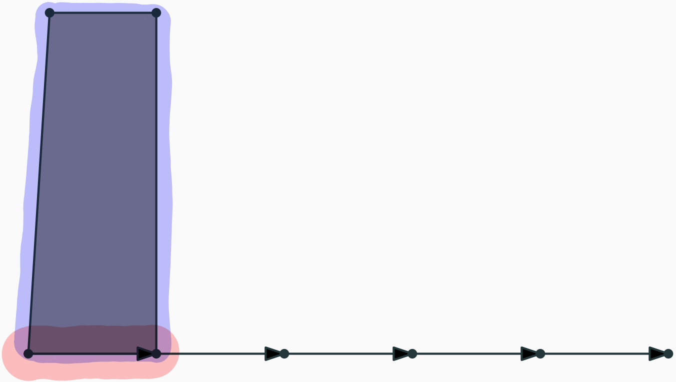
$Box(x,y,x',y'), E(x,y), E(y,z) \rightarrow \exists z' \; Box(y,z,y',z')$

This ruleset is not local and is BDD!

BDD is not local!

$E(x, y) \to \exists x', y'\ Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \to \exists z'\ Box(y, z, y', z')$

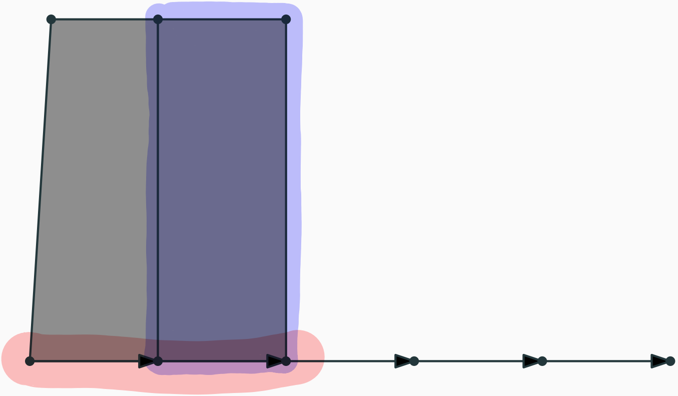


This ruleset is not local and is BDD!

BDD is not local!

$E(x, y) \rightarrow \exists x', y'\ Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z'\ Box(y, z, y', z')$

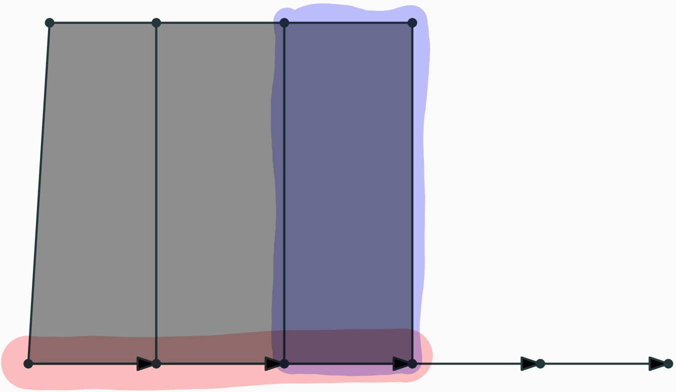


This ruleset is not local and is BDD!

BDD is not local!

$E(x, y) \rightarrow \exists x', y'\ Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z'\ Box(y, z, y', z')$

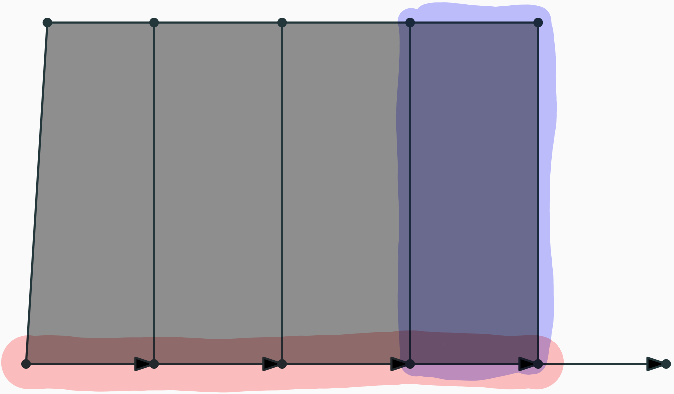


This ruleset is not local and is BDD!

BDD is not local!

$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$

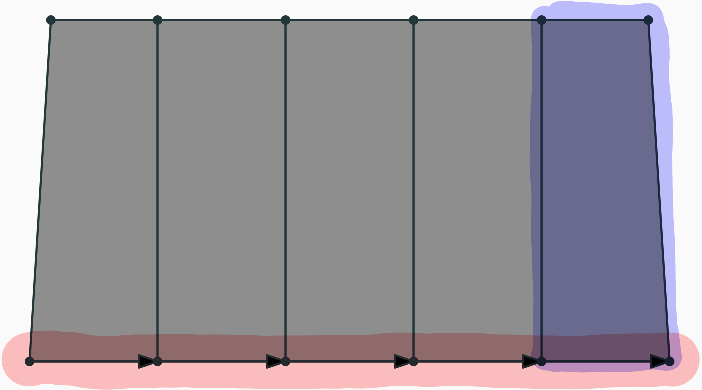


This ruleset is not local and is BDD!

BDD is not local!

$E(x, y) \rightarrow \exists x', y' \; Box(x, y, x', y')$

$Box(x, y, x', y'), E(x, y), E(y, z) \rightarrow \exists z' \; Box(y, z, y', z')$



This ruleset is not local and is BDD!

Linear-width rewritings

Linear-width rewritings

BDD definition (by FO-rewritings)

A ruleset $\mathcal{R}$ is BDD (FO-rewritable) if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

Linear-width rewritings

BDD definition (by FO-rewritings)

A ruleset $\mathcal{R}$ is BDD (FO-rewritable) if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

The linear-width rewritings class

The *width* of a UCQ is the maximal size of its disjuncts.

Linear-width rewritings

BDD definition (by FO-rewritings)

A ruleset $\mathcal{R}$ is BDD (FO-rewritable) if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

The linear-width rewritings class

The *width* of a UCQ is the maximal size of its disjuncts.

We say that a ruleset $\mathcal{R}$ admits *linear-width rewritings* if it is BDD and for every CQ its UCQ rewriting is of linear size width.

Linear-width rewritings

BDD definition (by FO-rewritings)

A ruleset $\mathcal{R}$ is BDD (FO-rewritable) if for every CQ $\mathcal{Q}$ there exists a UCQ $\mathcal{Q}'$ such that for every database $\mathcal{D}$ the following holds

$$\mathcal{D}, \mathcal{R} \models \mathcal{Q} \iff \mathcal{D} \models \mathcal{Q}'$$

The linear-width rewritings class

The *width* of a UCQ is the maximal size of its disjuncts.

We say that a ruleset $\mathcal{R}$ admits *linear-width rewritings* if it is BDD and for every CQ its UCQ rewriting is of linear size width.

Every known decidable subclass of BDD admits linear-width rewritings!

BDD is not about linear-width rewritings

BDD is not about linear-width rewritings

$\rightarrow \exists x \; H(x,x), V(x,x)$

BDD is not about linear-width rewritings

$$\rightarrow \exists x\, H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\, H(x,y), V(x,z)$$

BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$

BDD is not about linear-width rewritings

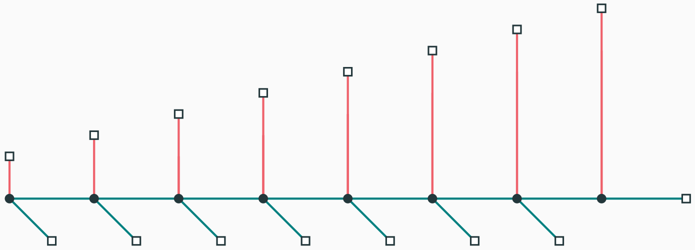
$$\to \exists x\ H(x,x), V(x,x)$$
$$\top(x) \to \exists y,z\ H(x,y), V(x,z)$$
$$H(x,y), H(y,z), V(x,x') \to \exists z'\ V(z,z'), H(x',z')$$

BDD is not about linear-width rewritings

$$\rightarrow \exists x \; H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z \; H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z' \; V(z,z'), H(x',z')$$

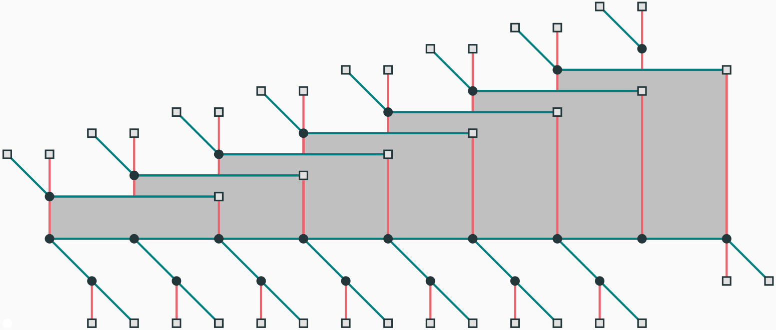


BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$



BDD is not about linear-width rewritings

$$\to \exists x \; H(x,x), V(x,x)$$

$$\top(x) \to \exists y,z \; H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \to \exists z' \; V(z,z'), H(x',z')$$

BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$

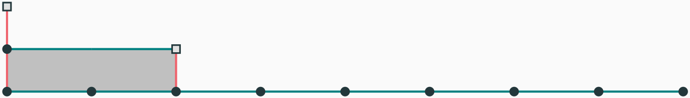
$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$

BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$

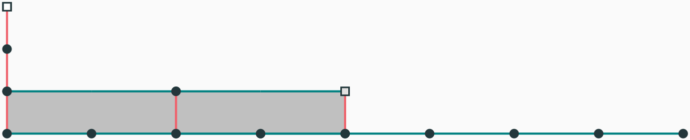


BDD is not about linear-width rewritings

$$\rightarrow \exists x \; H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z \; H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z' \; V(z,z'), H(x',z')$$

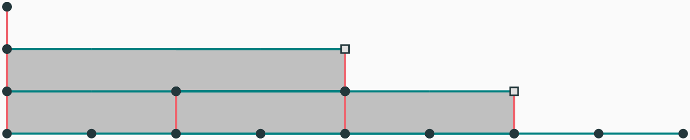


BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

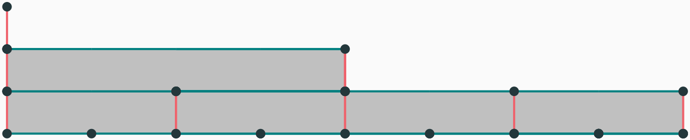
$$\top(x) \rightarrow \exists y, z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$



BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$
$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$
$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$

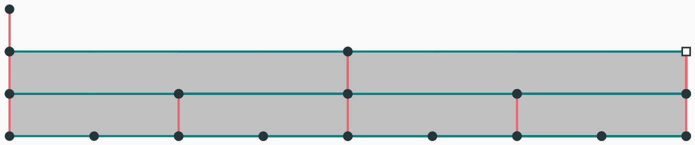


BDD is not about linear-width rewritings

$$\rightarrow \exists x\ H(x,x), V(x,x)$$

$$\top(x) \rightarrow \exists y,z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \rightarrow \exists z'\ V(z,z'), H(x',z')$$

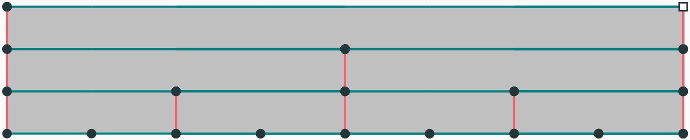


BDD is not about linear-width rewritings

$$\to \exists x\ H(x,x), V(x,x)$$

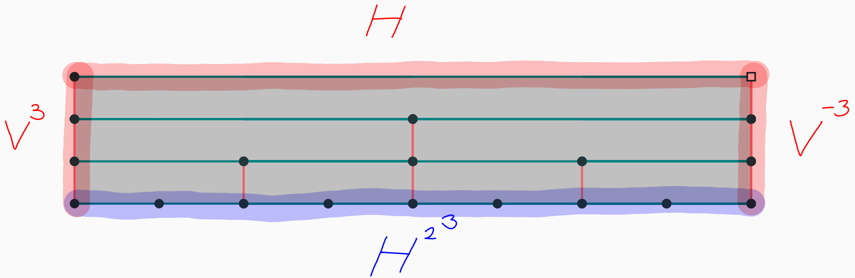
$$\top(x) \to \exists y,z\ H(x,y), V(x,z)$$

$$H(x,y), H(y,z), V(x,x') \to \exists z'\ V(z,z'), H(x',z')$$



BDD is not about linear-width rewritings

$$\to \exists x\ H(x,x), V(x,x)$$
$$\top(x) \to \exists y,z\ H(x,y), V(x,z)$$
$$H(x,y), H(y,z), V(x,x') \to \exists z'\ V(z,z'), H(x',z')$$



Rewriting of $V^n H V^{-n}$ contains a disjunct H^{2^n}!

Why is this important?

Why is this important?

It shows deficiency in the understanding of an impactful class of existential rules.

Why is this important?

It shows deficiency in the understanding of an impactful class of existential rules.

How can we use it?

Why is this important?

It shows deficiency in the understanding of an impactful class of existential rules.

How can we use it?

It gives a concrete example of behaviour that can be formalized and used for more expressive knowledge representation.

Thank You