Towards an End-to-End ASP-Based System for Handling Inconsistent Prioritized Data

Meghyn Bienvenu — CNRS & Université de Bordeaux
Camille Bourgaux — CNRS & DI ENS
Katsumi Inoue — NII
Robin Jean — CNRS & Université de Bordeaux
Giuseppe Mazzotta — University of Calabria

July 24, 2026

Inconsistent prioritized data

Inconsistent data

Knowledge base $\mathcal{K}$ = Dataset $\mathcal{D}$ + logical theory $\mathcal{T}$.

We suppose here that $\mathcal{D}$ is inconsistent w.r.t. to $\mathcal{T}$.

Inconsistent data

Knowledge base $\mathcal{K}$ = Dataset $\mathcal{D}$ + logical theory $\mathcal{T}$.

We suppose here that $\mathcal{D}$ is inconsistent w.r.t. to $\mathcal{T}$.

Repair $\mathcal{R}$: $\subseteq$-maximal consistent subset of $\mathcal{D}$
Conflict $\mathcal{C}$: $\subseteq$-minimal inconsistent subset of $\mathcal{D}$

Inconsistent data

Knowledge base $\mathcal{K}$ = Dataset $\mathcal{D}$ + logical theory $\mathcal{T}$.

We suppose here that $\mathcal{D}$ is inconsistent w.r.t. to $\mathcal{T}$.

Repair $\mathcal{R}$: $\subseteq$-maximal consistent subset of $\mathcal{D}$
Conflict $\mathcal{C}$: $\subseteq$-minimal inconsistent subset of $\mathcal{D}$

$q(\vec{a})$ holds under

- brave: in some repair
- AR: in every repair
- IAR: in the intersection of the repairs

Prioritized KB

Priority relation $\succ$: acyclic relation over $\mathcal{D}$,
with $\alpha \succ \beta$ only if α and β occur together in a conflict.

Prioritized KB

Priority relation $\succ$: acyclic relation over $\mathcal{D}$,
with $\alpha \succ \beta$ only if α and β occur together in a conflict.

Prioritized KB $\mathcal{K}_\succ$ = KB $\mathcal{K}$ + priority relation $\succ$

Prioritized KB

Priority relation $\succ$: acyclic relation over $\mathcal{D}$,
with $\alpha \succ \beta$ only if α and β occur together in a conflict.

Prioritized KB $\mathcal{K}_\succ$ = KB $\mathcal{K}$ + priority relation $\succ$

$\rightsquigarrow$ select **optimal repairs**.

Optimal repairs

A repair $\mathcal{R}$ can be improved by a consistent $\mathcal{B} \subseteq \mathcal{D}$:

- Pareto: one $\beta \in \mathcal{B} \setminus \mathcal{R}$ preferred to all of $\mathcal{R} \setminus \mathcal{B}$
- global: each $\alpha \in \mathcal{R} \setminus \mathcal{B}$ is preferred by some $\beta \in \mathcal{B} \setminus \mathcal{R}$

Optimal repairs

A repair $\mathcal{R}$ can be improved by a consistent $\mathcal{B} \subseteq \mathcal{D}$:

- Pareto: one $\beta \in \mathcal{B} \setminus \mathcal{R}$ preferred to all of $\mathcal{R} \setminus \mathcal{B}$
- global: each $\alpha \in \mathcal{R} \setminus \mathcal{B}$ is preferred by some $\beta \in \mathcal{B} \setminus \mathcal{R}$

$\mathcal{R}$ is **Pareto-** or **globally-optimal** if it admits no such improvement, **completion-optimal** if Pareto-optimal w.r.t. some completion of $\succ$.

Optimal repairs

A repair $\mathcal{R}$ can be improved by a consistent $\mathcal{B} \subseteq \mathcal{D}$:

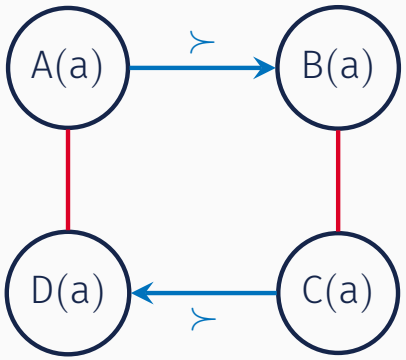
- Pareto: one $\beta \in \mathcal{B} \setminus \mathcal{R}$ preferred to all of $\mathcal{R} \setminus \mathcal{B}$
- global: each $\alpha \in \mathcal{R} \setminus \mathcal{B}$ is preferred by some $\beta \in \mathcal{B} \setminus \mathcal{R}$

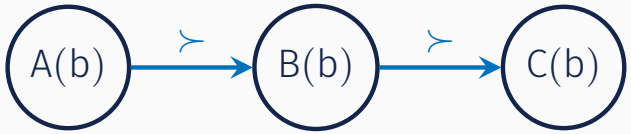
$\mathcal{R}$ is **Pareto-** or **globally-optimal** if it admits no such improvement, **completion-optimal** if Pareto-optimal w.r.t. some completion of $\succ$.

Theorem $CRep(\mathcal{K}_\succ) \subseteq GRep(\mathcal{K}_\succ) \subseteq PRep(\mathcal{K}_\succ) \subseteq SRep(\mathcal{K})$

$\rightsquigarrow$ X-brave, X-AR, X-IAR semantics, X $\in \{$P, G, C$\}$

Example



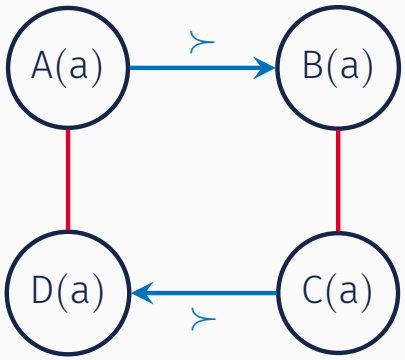


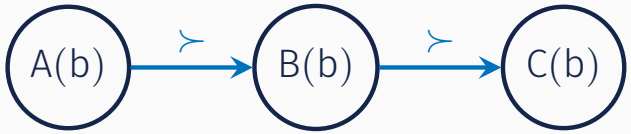
edges = conflicts

— no priority

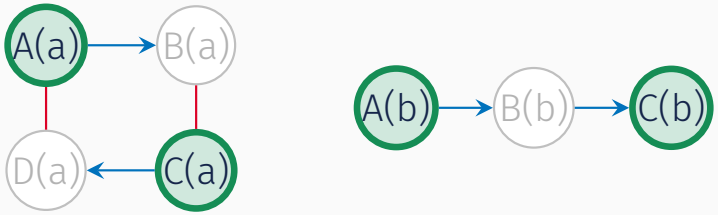
⟶ priority

Example

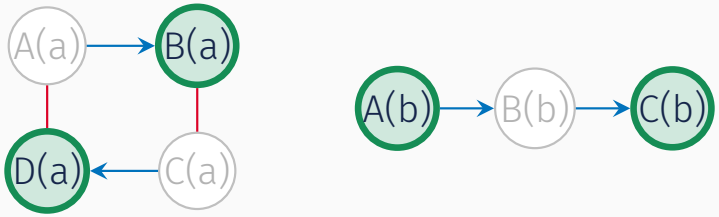




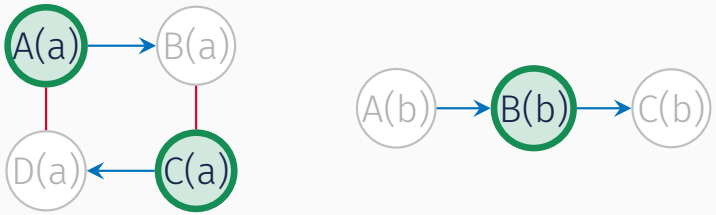
edges = conflicts — no priority ⟶ priority



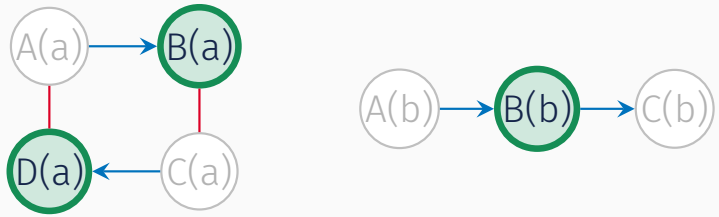
$\mathcal{R}_1$: **P-, G- and C-optimal**



$\mathcal{R}_2$: **P-optimal only**



$\mathcal{R}_3$: not optimal



$\mathcal{R}_4$: not optimal

Complexity landscape (data complexity)

	X-brave	X-AR	X-IAR
$X = S$	AC^0	coNP	AC^0
$X \in \{P, C\}$	NP	coNP	coNP
$X = G$	Σ_2^p	Π_2^p	Π_2^p

Complexity landscape (data complexity)

	X-brave	X-AR	X-IAR
$X = S$	AC^0	coNP	AC^0
$X \in \{P, C\}$	NP	coNP	coNP
$X = G$	Σ_2^p	Π_2^p	Π_2^p

Two gaps towards practical use:

- no simple way for users to **specify** $\succ$
- existing SAT-based system: **no globally-optimal repairs**, binary conflicts only

Specifying priority relations

Where does $\succ$ come from?

Nobody will input a **binary relation over millions of facts**.

Where does $\succ$ come from?

Nobody will input a **binary relation over millions of facts**.

Idea: **preference rules**

$$\mathrm{Cond}(x_1, x_2) \rightarrow \mathrm{pref}(x_1, x_2)$$

Where does $\succ$ come from?

Nobody will input a **binary relation over millions of facts**.

Idea: **preference rules**

$$\mathrm{Cond}(x_1, x_2) \rightarrow \mathrm{pref}(x_1, x_2)$$

Cond may refer to

- the dataset: presence or absence of facts
- a meta-database $\mathcal{M} = (\mathbf{id}, \mathcal{F})$: identifiers + timestamps, sources, ...
- the ontology axioms

Example

- $\mathsf{Date}(x_1, y_1) \wedge \mathsf{Date}(x_2, y_2) \wedge <(y_2, y_1) \rightarrow \mathsf{pref}(x_1, x_2)$
- $x_1 = \mathbf{id}(\mathsf{FPr}(y)) \wedge x_2 = \mathbf{id}(\mathsf{APr}(y)) \rightarrow \mathsf{pref}(x_1, x_2)$
- $Y \sqsubseteq \mathsf{Adm} \wedge Z \sqsubseteq \mathsf{Fac} \wedge \neg(\exists z\, \mathsf{Teach}(y, z)) \wedge$
  $x_1 = \mathbf{id}(Y(y)) \wedge x_2 = \mathbf{id}(Z(y)) \rightarrow \mathsf{pref}(x_1, x_2)$

Example

- $\mathsf{Date}(x_1, y_1) \wedge \mathsf{Date}(x_2, y_2) \wedge <(y_2, y_1) \rightarrow \mathsf{pref}(x_1, x_2)$
- $x_1 = \mathbf{id}(\mathsf{FPr}(y)) \wedge x_2 = \mathbf{id}(\mathsf{APr}(y)) \rightarrow \mathsf{pref}(x_1, x_2)$
- $Y \sqsubseteq \mathsf{Adm} \wedge Z \sqsubseteq \mathsf{Fac} \wedge \neg(\exists z\, \mathsf{Teach}(y, z)) \wedge$
  $x_1 = \mathbf{id}(Y(y)) \wedge x_2 = \mathbf{id}(Z(y)) \rightarrow \mathsf{pref}(x_1, x_2)$

prefer the more recent fact

if stated both as a full and associate professor we prioritize the fact that its a full professor

whoever teaches nothing is preferably believed to be admin staff

From preferences to a priority relation

$\succ_\Sigma$ = induced preferences, restricted to pairs occurring in a conflict.

From preferences to a priority relation

$\succ_\Sigma$ = induced preferences, restricted to pairs occurring in a conflict.

But $\succ_\Sigma$ may contain **cycles**:

e.g., prefer recent facts and prefer facts from reliable sources.

From preferences to a priority relation

$\succ_\Sigma$ = induced preferences, restricted to pairs occurring in a conflict.

But $\succ_\Sigma$ may contain **cycles**:
e.g., prefer recent facts and prefer facts from reliable sources.

Two complementary answers:

- **check** that a ruleset can never induce a cycle
- **remove** the cycles

Checking acyclicity

Is $\succ_\Sigma$ acyclic for every dataset and meta-database?

Checking acyclicity

Is $\succ_\Sigma$ acyclic for every dataset and meta-database?

Theorem Undecidable in general.

Checking acyclicity

Is $\succ_\Sigma$ acyclic for every dataset and meta-database?

Theorem Undecidable in general.

Theorem In coNP for DL-Lite ontologies and rule bodies not containing ontology axioms.

Removing cycles

Rules are partitioned into priority levels $\Sigma_1, \ldots, \Sigma_n$.

Removing cycles

Rules are partitioned into priority levels $\Sigma_1, \ldots, \Sigma_n$.

Four cycle resolution techniques:

- **going up** ($\succ^{u}$) add whole levels while acyclic
- **going down** ($\succ^{d}$) drop cyclic preferences, least important first
- **refined going up** ($\succ^{ru}$) add level-wise, only facts not creating cycles
- **grounded** ($\succ^{g}$) fixpoint computation

Removing cycles

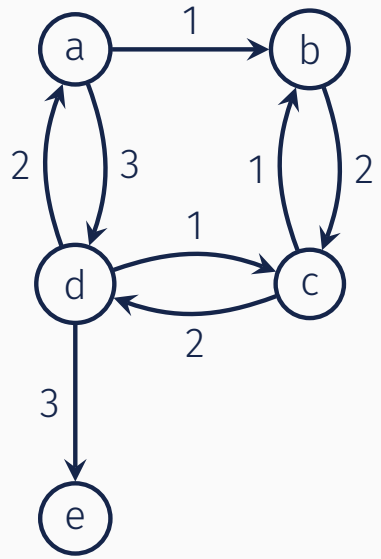
Rules are partitioned into priority levels $\Sigma_1, \ldots, \Sigma_n$.

Four cycle resolution techniques:

- **going up** ($\succ^{u}$) add whole levels while acyclic
- **going down** ($\succ^{d}$) drop cyclic preferences, least important first
- **refined going up** ($\succ^{ru}$) add level-wise, only facts not creating cycles
- **grounded** ($\succ^{g}$) fixpoint computation

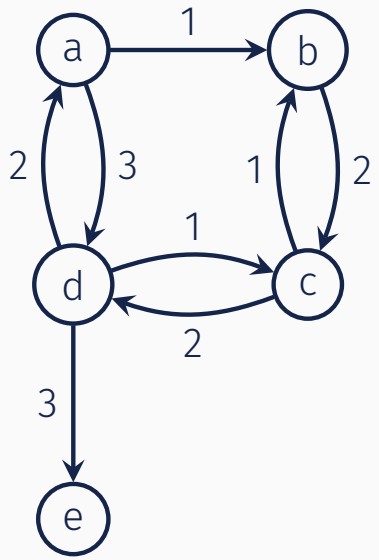
Theorem $\succ^{u} \subseteq \succ^{d} \subseteq \succ^{g}$, $\succ^{u} \subseteq \succ^{d} \subseteq \succ^{ru}$, all in PTIME.

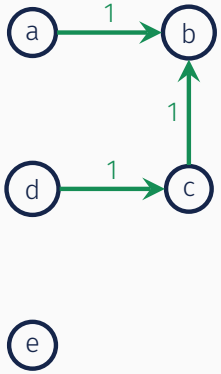
Example



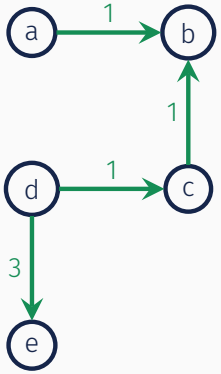
nodes = facts
labels = priority level of the rule inducing the preference

Example

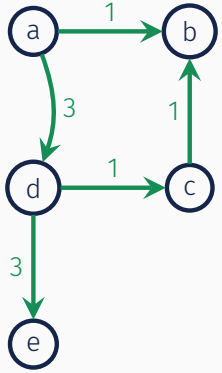




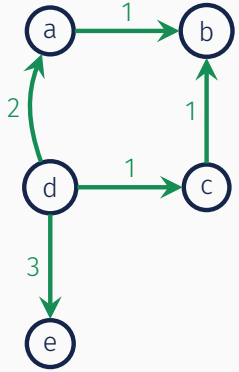
$\prec^{u}$



$\prec^{d}$



$\prec^{ru}$



$\prec^{g}$

Implementations in ASP and ASP(Q)

Answer set programming

ASP: say what the solutions are, not how to compute them.

A problem is encoded as a logic program, its solutions are the answer sets.

Answer set programming

ASP: say what the solutions are, not how to compute them.

A problem is encoded as a logic program, its solutions are the answer sets.

```
fact(a). fact(b). conflict(a,b).   input
{in(X)} :- fact(X).                guess
 :- in(X), in(Y), conflict(X,Y).   check
```

Answer sets: `{in(a)}`, `{in(b)}`

Answer set programming

ASP: say what the solutions are, not how to compute them.

A problem is encoded as a logic program, its solutions are the answer sets.

```
fact(a). fact(b). conflict(a,b).   input
{in(X)} :- fact(X).                guess
 :- in(X), in(Y), conflict(X,Y).   check
```

Answer sets: `{in(a)}`, `{in(b)}`

$\rightsquigarrow$ non-disjunctive ASP encodes any problem in NP.
$\rightsquigarrow$ adding disjunction reach Σ_2^p

Computing the priority relation

Input, as ASP programs: $\mathcal{D}$, $\mathcal{T}$, $\mathcal{M}$, and Σ with its levels.

Computing the priority relation

Input, as ASP programs: $\mathcal{D}$, $\mathcal{T}$, $\mathcal{M}$, and Σ with its levels.

- evaluate the preference rules
- apply the chosen cycle resolution technique

Computing the priority relation

Input, as ASP programs: $\mathcal{D}$, $\mathcal{T}$, $\mathcal{M}$, and Σ with its levels.

- evaluate the preference rules
- apply the chosen cycle resolution technique

Output: the priority relation $\succ^{x}$, $x \in \{u, d, ru, g\}$.

X-brave and X-AR, $X \in \{P, C\}$

In NP / coNP $\rightsquigarrow$ **plain (non-disjunctive) ASP**, from modular building blocks:

X-brave and X-AR, $X \in \{P, C\}$

In NP / coNP $\rightsquigarrow$ **plain (non-disjunctive) ASP**, from modular building blocks:

1. localize to the facts reachable from the causes (minimal supports)
2. guess a conflict-free subset $\mathcal{R}$ of them
3. enforce that $\mathcal{R}$ is Pareto- (resp. completion-) optimal
4. require that $\mathcal{R}$ contains some cause (brave) or no cause (AR)

X-brave and X-AR, $X \in \{P, C\}$

In NP / coNP $\rightsquigarrow$ **plain (non-disjunctive) ASP**, from modular building blocks:

1. localize to the facts reachable from the causes (minimal supports)
2. guess a conflict-free subset $\mathcal{R}$ of them
3. enforce that $\mathcal{R}$ is Pareto- (resp. completion-) optimal
4. require that $\mathcal{R}$ contains some cause (brave) or no cause (AR)

Conflicts may be of **arbitrary size**.

Why ASP(Q)?

ASP natural up to the first level of the polynomial hierarchy.

Why ASP(Q)?

ASP natural up to the first level of the polynomial hierarchy.

Second level $\Rightarrow$ disjunction $+$ saturation: notoriously hard to write.

Why ASP(Q)?

ASP natural up to the first level of the polynomial hierarchy.

Second level $\Rightarrow$ disjunction + saturation: notoriously hard to write.

ASP(Q): quantifiers over the answer sets of ASP programs

$$\square_1 P_1 \ldots \square_n P_n : C \qquad \square_i \in \{\exists^{st}, \forall^{st}\}$$

$\rightsquigarrow$ compact encodings for the whole polynomial hierarchy.

G-brave and G-AR

Global semantics Σ_2^{p} / $\Pi_2^{\mathsf{p}} \rightsquigarrow$ ASP(Q)

$$\Pi^{\mathsf{G}}_{brave} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{SomeCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$

$$\Pi^{\mathsf{G}}_{AR} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{NoCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$

G-brave and G-AR

Global semantics Σ_2^{p} / $\Pi_2^{\mathsf{p}} \rightsquigarrow$ ASP(Q)

$$\Pi^{\mathsf{G}}_{brave} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{SomeCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$
$$\Pi^{\mathsf{G}}_{AR} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{NoCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$

- $\exists^{\mathsf{st}}$: guess a repair containing some / no cause
- $\forall^{\mathsf{st}}$: guess a global improvement of it
- C: never satisfied

G-brave and G-AR

Global semantics Σ_2^{p} / Π_2^{p} $\rightsquigarrow$ ASP(Q)

$$\Pi^{\mathsf{G}}_{brave} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{SomeCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$
$$\Pi^{\mathsf{G}}_{AR} = \exists^{\mathsf{st}}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{NoCause})\ \forall^{\mathsf{st}}\Pi_{GImp} : \mathsf{C}$$

- $\exists^{\mathsf{st}}$: guess a repair containing some / no cause
- $\forall^{\mathsf{st}}$: guess a global improvement of it
- C: never satisfied

$\rightsquigarrow$ coherent iff $\forall^{\mathsf{st}}$ is **trivially** satisfied, i.e., the repair **has no global improvement**.

Grounded semantics

From abstract argumentation: $\mathcal{K}_{\succ} \rightsquigarrow$ attack relation $\rightsquigarrow$ grounded extension.

Grounded repair $\mathcal{G}$: the facts defended against all their attackers.

Grounded semantics

From abstract argumentation: $\mathcal{K}_{\succ} \rightsquigarrow$ attack relation $\rightsquigarrow$ grounded extension.

Grounded repair $\mathcal{G}$: the facts defended against all their attackers.

Theorem $\mathcal{G} \subseteq \mathcal{R}$ for every $\mathcal{R} \in XRep(\mathcal{K}_{\succ})$, X $\in \{\mathsf{P}, \mathsf{G}, \mathsf{C}\}$.

$\rightsquigarrow$ grounded answers hold under X-IAR, X-AR and X-brave.

Grounded semantics

From abstract argumentation: $\mathcal{K}_{\succ} \rightsquigarrow$ attack relation $\rightsquigarrow$ grounded extension.

Grounded repair $\mathcal{G}$: the facts defended against all their attackers.

Theorem $\mathcal{G} \subseteq \mathcal{R}$ for every $\mathcal{R} \in XRep(\mathcal{K}_{\succ})$, X $\in \{\mathsf{P}, \mathsf{G}, \mathsf{C}\}$.

$\rightsquigarrow$ grounded answers hold under X-IAR, X-AR and X-brave.

Theorem Grounded query answering is in PTime (for a large class of theories).

X-IAR and cheap approximations

X-IAR: a fact is in all optimal repairs iff it holds under X-AR.

$\rightsquigarrow$ compute $\bigcap_{\mathcal{R} \in XRep(\mathcal{K}_{\succ})} \mathcal{R}$ fact by fact.

X-IAR and cheap approximations

X-IAR: a fact is in all optimal repairs iff it holds under X-AR.

$\rightsquigarrow$ compute $\bigcap_{\mathcal{R} \in XRep(\mathcal{K}_{\succ})} \mathcal{R}$ fact by fact.

Under-approximations of P-IAR, hence of all these semantics:

- the trivially P-IAR answers
- the grounded repair $\mathcal{G}$

X-IAR and cheap approximations

X-IAR: a fact is in all optimal repairs iff it holds under X-AR.

$\rightsquigarrow$ compute $\bigcap_{\mathcal{R} \in XRep(\mathcal{K}_{\succ})} \mathcal{R}$ fact by fact.

Under-approximations of P-IAR, hence of all these semantics:

- the trivially P-IAR answers
- the grounded repair $\mathcal{G}$

$\rightsquigarrow$ answer as many queries as possible **without** the encodings.

Experiments

Setting

CQAPri benchmark: DL-Lite, adapted from LUBM$^{\exists}_{20}$, extended with priority relations.

$+$ a denial constraint yielding conflicts of size 10.

Setting

CQAPri benchmark: DL-Lite, adapted from LUBM$^{\exists}_{20}$, extended with priority relations.

$+$ a denial constraint yielding conflicts of size 10.

Solvers: CASPER for ASP(Q), CLINGO for ASP.

Computing the priority relation

$\succ^{\mathsf{u}}$, $\succ^{\mathsf{d}}$, $\succ^{\mathsf{g}}$ computed on 75K and 463K fact datasets,
even with more than 29% of facts in conflicts.

Computing the priority relation

$\succ^{\mathsf{u}}$, $\succ^{\mathsf{d}}$, $\succ^{\mathsf{g}}$ computed on 75K and 463K fact datasets, even with more than 29% of facts in conflicts.

$\rightsquigarrow \succ^{\mathsf{ru}}$ **never** computable: atom count overflow.

$\rightsquigarrow \succ^{\mathsf{u}}$ often reduced to the **empty** relation.

$\rightsquigarrow \succ^{\mathsf{d}}$ and $\succ^{\mathsf{g}}$ differ by at most 5%, and $\succ^{\mathsf{d}}$ is much faster.

Computing the priority relation

$\succ^{\mathsf{u}}$, $\succ^{\mathsf{d}}$, $\succ^{\mathsf{g}}$ computed on 75K and 463K fact datasets, even with more than 29% of facts in conflicts.

$\rightsquigarrow \succ^{\mathsf{ru}}$ **never** computable: atom count overflow.

$\rightsquigarrow \succ^{\mathsf{u}}$ often reduced to the **empty** relation.

$\rightsquigarrow \succ^{\mathsf{d}}$ and $\succ^{\mathsf{g}}$ differ by at most 5%, and $\succ^{\mathsf{d}}$ is much faster.

$\succ^{\mathsf{d}}$: a good method in practice

Query answering

$\rightsquigarrow$ **localization is crucial**, even on the smallest datasets.

Query answering

$\rightsquigarrow$ **localization is crucial**, even on the smallest datasets.

$\rightsquigarrow$ **grounded is surprisingly effective**: many answers are grounded, $\mathcal{G}$ is cheap, and it covers most of the intersection of the optimal repairs.

Query answering

$\rightsquigarrow$ **localization is crucial**, even on the smallest datasets.

$\rightsquigarrow$ **grounded is surprisingly effective**: many answers are grounded, $\mathcal{G}$ is cheap, and it covers most of the intersection of the optimal repairs.

$\rightsquigarrow$ **globally-optimal repairs are costly**: > 200s for G-AR in 51% of the cases, against ≤ 14s for P-AR.

Query answering

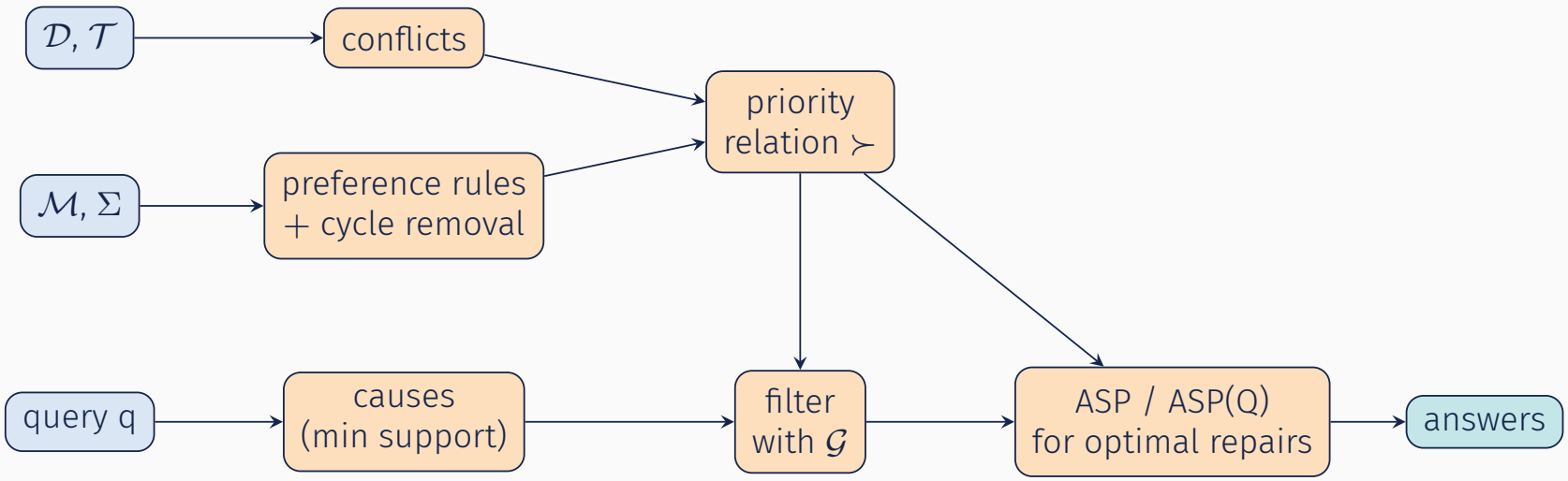
$\rightsquigarrow$ **localization is crucial**, even on the smallest datasets.

$\rightsquigarrow$ **grounded is surprisingly effective**: many answers are grounded, $\mathcal{G}$ is cheap, and it covers most of the intersection of the optimal repairs.

$\rightsquigarrow$ **globally-optimal repairs are costly**: > 200s for G-AR in 51% of the cases, against ≤ 14s for P-AR.

grounded $\rightarrow$ P-AR / P-brave $\rightarrow$ G-encodings only if unresolved

An end-to-end pipeline



Perspectives

Theory

- more theories and rule languages for the static analysis
- when is the optimal repair unique?

Perspectives

Theory

- more theories and rule languages for the static analysis
- when is the optimal repair unique?

Practice

- updating the grounded repair on very large datasets
- further optimization of the encodings
- experimental evaluation of the saturation based encodings

Perspectives

Theory

- more theories and rule languages for the static analysis
- when is the optimal repair unique?

Practice

- updating the grounded repair on very large datasets
- further optimization of the encodings
- experimental evaluation of the saturation based encodings

Thank you for your attention!

Backup: computing $\succ^{X}$ from the preference rules

$\Pi_{\succ^{u}}$	`trans_cl(X,Y,I) :- pref_init(X,Y,I), not blocked(I).`
	`trans_cl(X,Y,I) :- level(I), trans_cl(X,Y,J), J < I, not blocked(I).`
	`trans_cl(X,Y,I) :- pref_init(X,Z,J), trans_cl(Z,Y,I), J <= I, not blocked(I).`
	`cycle(I) :- trans_cl(X,X,I).` `blocked(I) :- level(I), cycle(J), J < I.`
	`pref(X,Y) :- pref_init(X,Y,I), not cycle(I), not blocked(I).`

$\Pi_{\succ^{d}}$	`trans_cl(X,Y,I) :- pref_init(X,Y,I).`
	`trans_cl(X,Y,I) :- pref_init(X,Z,I), trans_cl(Z,Y,J), J <= I.`
	`trans_cl(X,Y,I) :- pref_init(X,Z,J), trans_cl(Z,Y,I), J <= I.`
	`cycle(X,Y,I) :- pref_init(X,Y,I), trans_cl(Y,X,I).`
	`pref(X,Y) :- pref_init(X,Y,I), not cycle(X,Y,I).`

$\Pi_{\succ^{ru}}$	`trans_cl(X,Y,I) :- pref_init(X,Y,I).`
	`trans_cl(X,Y,I) :- level(I), rel(X,Y,J), J < I.`
	`trans_cl(X,Y,I) :- pref_init(X,Z,I), trans_cl(Z,Y,I).`
	`trans_cl(X,Y,I) :- trans_cl(X,Z,I), trans_cl(Z,Y,I).`
	`cycle(X,Y,I) :- pref_init(X,Y,I), trans_cl(Y,X,I).`
	`rel(X,Y,I) :- pref_init(X,Y,I), not cycle(X,Y,I).`
	`rel(X,Y,I) :- rel(X,Z,J), rel(Z,Y,I), J <= I.`
	`pref(X,Y) :- pref_init(X,Y,I), rel(X,Y,I).`

Backup: building blocks for X-brave and X-AR

Π_{Attack}	non_attacking(C, A) :- conf(C), inConf(C, A), inConf(C, B), A != B, pref(A, B). attacks(C, A) :- conf(C), inConf(C, A), not non_attacking(C, A).
Π_{Reach}	Π_{Attack} reachable(A) :- cause(C), inCause(C, A). reachable(A) :- conf(C), attacks(C, B), reachable(B), inConf(C, A).
Π_{SubRep}	{inRepair(A)} :- reachable(A). solved(C) :- inConf(C, A), not inRepair(A). :- conf(C), not solved(C).
$\Pi_{SatIfCause}$	violatedCause(C) :- inCause(C, A), not inRepair(A). sat :- cause(C), not violatedCause(C).
$\Pi_{SomeCause}$	$\Pi_{SatIfCause}$:- not sat.
$\Pi_{NoCause}$	$\Pi_{SatIfCause}$:- sat.
Π_{POpt}	Π_{Attack} valid(A) :- reachable(A), inRepair(A). invalid_att(C, A) :- reachable(A), attacks(C, A), inConf(C, B), not inRepair(B), not A = B. valid(A) :- reachable(A), conf(C), not inRepair(A), attacks(C, A), not invalid_att(C, A). :- reachable(A), not valid(A).
Π_{Compl}	pref_comp(A, B) :- reachable(A), reachable(B), pref(A, B). 1{pref_comp(A, B); pref_comp(B, A)}1 :- reachable(A), reachable(B), inConf(C, A), inConf(C, B), not pref(A, B), not pref(B, A), not A = B. trans_cl_comp(A, B) :- pref_comp(A, B). trans_cl_comp(A, B) :- trans_cl_comp(A, Y), pref_comp(Y, B). :- trans_cl_comp(A, A).

Backup: repair and global improvement

Π_{Rep}	Π_{SubRep} safe(C) :- inConf(C, A), inConf(C, B), not A = B, not inRepair(A), not inRepair(B). keepOut(A) :- inConf(C, A), not inRepair(A), not safe(C). :- reachable(A), not inRepair(A), not keepOut(A).
Π_{GImp}	{global_imp(A)} :- reachable(A). solved_global(C) :- inConf(C, A), not global_imp(A). :- conf(C), not solved_global(C). impMinusRepair(A) :- global_imp(A), not inRepair(A). repairMinusImp(A) :- inRepair(A), not global_imp(A). diff :- impMinusRepair(A). diff :- repairMinusImp(A). fake_improvement :- not diff. ok(A) :- repairMinusImp(A), impMinusRepair(B), pref(B, A). fake_improvement :- repairMinusImp(A), not ok(A). global_improvement :- not fake_improvement. :- not global_improvement.

Backup: computing the grounded repair

$\Pi_{\Gamma(\emptyset)}$	`unsafe(A,0) :- attacks(C,A).` `safe(A,0) :- inConf(C,A), not unsafe(A,0).`

$\Pi_{\Gamma\text{incr}}$	`safe(A,t) :- safe(A,t-1).`
	`non_subset(C,A,t) :- conf(C), inConf(C,A), inConf(C,B), A != B, not safe(B,t-1).`
	`subset(C,A,t) :- conf(C), inConf(C,A), not non_subset(C,A,t).`
	`protected(C,A,t) :- attacks(C,A), inConf(C,B), A != B, attacks(C2,B), subset(C2,B,t).`
	`unsafe(A,t) :- attacks(C,A), not protected(C,A,t).`
	`safe(A,t) :- inConf(C,A), not unsafe(A,t).`
	`continue(t) :- safe(A,t), not safe(A,t-1).`

Incremental solving mirrors the fixpoint definition of the grounded semantics.