Belief Change and Answer Set Programming

Odile Papini & Julien Hué & Éric Würbel

LSIS Laboratory UMR CNRS 6168– INCA team
Université d’Aix-Marseille. FRANCE
Université du Sud Toulon Var. FRANCE
Programming and belief change

- Declarative formalism: logic programs with default negation several semantics
  - ASP with stable models semantics [Gelfond et al 1988]:
    - efficient tool for implementing propositional belief bases change operations
    - revision: [Benferhat et al. 2010],
    - fusion: [Hué et al. 2008]

- ASP with SE-models semantics [Turner 2003]:
  - unified formalism for both representing and implementing change operations
  - revision: [Delgrande et al. 2008],
  - fusion: [Delgrande et al. 2009], [Hué et al. 2009]

- several efficient solvers: smodels, DLV, CLASP
1 Introduction

2 Answer Set Programming
   - Syntax
   - Answer Sets
   - Strong negation

3 Merging logic programs

4 ΠRSF a syntactic framework for merging or revising logic programs

5 Conclusion
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \): atoms from a set \( \mathcal{L} \)
- \( \text{not} \): default negation (negation as failure)
- \( "," \): conjunction
- if \( n = 0 \): a fact
- if \( n = m \): a basic rule
- the atom \( a_0 \) is the constant \( \bot \): an integrity constraint
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \): atoms from a set \( \mathcal{L} \)
- \( \text{not} \): default negation (negation as failure)
- \( , \): conjunction
- \( \text{if } n = 0 \): a fact
- \( \text{if } n = m \): a basic rule
- the atom \( a_0 \) is the constant \( \bot \): an integrity constraint
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \) : atoms from a set \( \mathcal{L} \)
- \( \text{not} \) : default negation (negation as failure)
- “,” : conjunction
  - if \( n = 0 \) : a fact
  - if \( n = m \) : a basic rule
- the atom \( a_0 \) is the constant \( \bot \) : an integrity constraint
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \): atoms from a set \( \mathcal{L} \)
- \( \text{not} \): default negation (negation as failure)
- "\,": conjunction
- if \( n = 0 \): a fact
- if \( n = m \): a basic rule
- the atom \( a_0 \) is the constant \( \bot \): an integrity constraint
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \): atoms from a set \( \mathcal{L} \)
- \( \text{not} \): default negation (negation as failure)
- \( \text{"\"\"} \): conjunction
- if \( n = 0 \): a fact
- if \( n = m \): a basic rule
- the atom \( a_0 \) is the constant \( \bot \): an integrity constraint
Normal Logic Program

A set of rules of the form:

\[ a_0 \leftarrow a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \]

- \( a_i \): atoms from a set \( \mathcal{L} \)
- \( \text{not} \): default negation (negation as failure)
- “,”: conjunction
- if \( n = 0 \): a fact
- if \( n = m \): a basic rule
- the atom \( a_0 \) is the constant \( \bot \): an integrity constraint
Definition (Gelfond-Lifschitz Reduction)

Let $\Pi$ be a normal logic program, and $X$ be a set of atoms, the reduct of $\Pi$ relative to $X$, denoted $\Pi^X$ is obtained from $\Pi$ by:

- dropping each rule containing a term $not\ a_i$, with $a_i \in X$ and,
- dropping the negative parts $not\ a_{m+1}, \ldots not\ a_n$ from the bodies of the remaining rules.

Definition

A set of atoms $X$ is an answer set of a normal program $\Pi$ iff $X$ is the minimal Herbrand model of $\Pi^X$.

$AS(\Pi)$ : the set of all answer sets of $\Pi$

example: answer set

$$\Pi = \{ a \leftarrow not\ b. \quad b \leftarrow not\ a. \}$$

$AS(\Pi) = \{\{a\}, \{b\}\}$
Definition (Gelfond-Lifschitz Reduction)

Let $\Pi$ be a normal logic program, and $X$ be a set of atoms, the reduct of $\Pi$ relative to $X$, denoted $\Pi^X$ is obtained from $\Pi$ by:
- dropping each rule containing a term $not\ a_i$, with $a_i \in X$ and,
- dropping the negative parts $not\ a_{m+1}, \ldots not\ a_n$ from the bodies of the remaining rules.

Definition

A set of atoms $X$ is an answer set of a normal program $\Pi$ iff $X$ is the minimal Herbrand model of $\Pi^X$.

$AS(\Pi) :$ the set of all answer sets of $\Pi$

example: answer set

$$\Pi = \{ a \leftarrow not\ b. \quad b \leftarrow not\ a. \}$$

$AS(\Pi) = \{\{a\}, \{b\}\}$
Definition (Gelfond-Lifschitz Reduction)

Let \( \Pi \) be a normal logic program, and \( X \) be a set of atoms, the reduct of \( \Pi \) relative to \( X \), denoted \( \Pi^X \) is obtained from \( \Pi \) by:

- dropping each rule containing a term \( not \ a_i \), with \( a_i \in X \) and,
- dropping the negative parts \( not \ a_{m+1}, \ldots not \ a_n \) from the bodies of the remaining rules.

Definition

A set of atoms \( X \) is an answer set of a normal program \( \Pi \) iff \( X \) is the minimal Herbrand model of \( \Pi^X \).

\( AS(\Pi) \) : the set of all answer sets of \( \Pi \)

example: answer set

\[
\Pi = \{ a \leftarrow not \ b. \quad b \leftarrow not \ a. \}\n\]

\( AS(\Pi) = \{\{a\}, \{b\}\} \)
Definition (SE-model (Turner 2003))

An **SE-interpretation** is a pair \((X, Y)\) of sets of atoms such that \(X \subseteq Y \subseteq \mathcal{L}\). An SE-interpretation \((X, Y)\) is a **SE-model** of a program \(\Pi\) iff \(Y \models \Pi\) and \(X \models \Pi^Y\).

\(SE(\Pi)\) : the set of all SE-models of \(\Pi\)

**Definition**

A set of atoms \(Y\) is an **of a normal logic program** \(\Pi\) iff \((Y, Y) \in SE(\Pi)\) and there is no \((X, Y) \in SE(\Pi)\) such that \(X \subset Y\).

**example:** s

\(\Pi = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a.\}\) \(\quad AS(\Pi) = \{\{a\}, \{b\}\}\)

\(SE(\Pi) = \{\{(a), \{a\}\}, \{(b), \{b\}\}, \{(a), \{a\}\}, (\emptyset, \{a, b\}\),
\(\{(a), \{a, b\}\}, \{(b), \{a, b\}\}, \{(a, b), \{a, b\}\}\)\)
Definition (SE-model (Turner 2003))

An SE-interpretation is a pair \((X, Y)\) of sets of atoms such that \(X \subseteq Y \subseteq \mathcal{L}\). An SE-interpretation \((X, Y)\) is a SE-model of a program \(\Pi\) iff \(Y \models \Pi\) and \(X \models \Pi^Y\).

\(SE(\Pi)\) : the set of all SE-models of \(\Pi\)

Definition

A set of atoms \(Y\) is an of a normal logic program \(\Pi\) iff \((Y, Y) \in SE(\Pi)\) and there is no \((X, Y) \in SE(\Pi)\) such that \(X \subset Y\).

example: s

\[\Pi = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a.\}\]

\[AS(\Pi) = \{\{a\}, \{b\}\}\]

\[SE(\Pi) = \{(\{a\}, \{a\}), (\{b\}, \{b\}), (\{a\}, \{a\}), (\emptyset, \{a, b\}), (\{a\}, \{a, b\}), (\{b\}, \{a, b\}), (\{a, b\}, \{a, b\})\}\]
Definition (SE-model (Turner 2003))

An SE-interpretation is a pair $(X, Y)$ of sets of atoms such that $X \subseteq Y \subseteq \mathcal{L}$. An SE-interpretation $(X, Y)$ is a SE-model of a program $\Pi$ iff $Y \models \Pi$ and $X \models \Pi^Y$.

$SE(\Pi)$: the set of all SE-models of $\Pi$

Definition

A set of atoms $Y$ is an of a normal logic program $\Pi$ iff $(Y, Y) \in SE(\Pi)$ and there is no $(X, Y) \in SE(\Pi)$ such that $X \subset Y$.

example: s

$\Pi = \{a \leftarrow \neg b. \quad b \leftarrow \neg a\} \quad AS(\Pi) = \{\{a\}, \{b\}\}$

$SE(\Pi) = \{\{(a), \{a\}\}, \{(b), \{b\}\}, \{(a), \{a\}\}, (\emptyset, \{a, b\}), \{(a), \{a, b\}\}, \{(b), \{a, b\}\}, \{(a, b), \{a, b\}\}\}$
**Extended program**

An extended program is a normal program where for each rule

\[ l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n. \]

each \( l_i \) is either an atom \( a \in \mathcal{L} \) or its (strong) negation \( \neg a \).

**Consequences**

From the answer set perspective, all definitions remain the same, except that an answer set must be a **consistent** set of literals.
Extended program

An extended program is a normal program where for each rule

\[ l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n. \]

each \( l_i \) is either an atom \( a \in \mathcal{L} \) or its (strong) negation \( \neg a \).

Consequences

From the answer set perspective, all definitions remain the same, except that an answer set must be a consistent set of literals.
Propositional belief bases change: ASP implementation

encoding the belief change problem: logic program

- knowledge representation
- potential solutions (guess rules)
- constraints (specific operation) (check rules)

with an ASP solver

- compute answer sets: potential solutions
- preference relation between answer sets according to a strategy: solutions
Propositional belief bases change: ASP implementation

**encoding the belief change problem: logic program**
- knowledge representation
- potential solutions (guess rules)
- constraints (specific operation) (check rules)

**with an ASP solver**
- compute answer sets: potential solutions
- preference relation between answer sets according to a strategy: solutions
Propositional belief bases change: ASP implementation

encoding the belief change problem: logic program
- knowledge representation
- potential solutions (guess rules)
- constraints (specific operation) (check rules)

with an ASP solver
- compute answer sets: potential solutions
- preference relation between answer sets according to a strategy: solutions
Propositional belief bases change: ASP implementation

**encoding the belief change problem: logic program**
- knowledge representation
- potential solutions (guess rules)
- constraints (specific operation) (check rules)

**with an ASP solver**
- compute answer sets: potential solutions
- preference relation between answer sets according to a strategy: solutions
Propositional belief bases change: ASP implementation

**encoding the belief change problem: logic program**
- knowledge representation
- potential solutions (guess rules)
- constraints (specific operation) (check rules)

**with an ASP solver**
- compute answer sets: potential solutions
- preference relation between answer sets according to a strategy: solutions
Introduction

Answer Set Programming

Merging logic programs
  - Goals and representation

ΠRSF a syntactic framework for merging or revising logic programs

Conclusion
Goals

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy
Goals

- obtain a global point of view from different sources
- solve conflicts
  - reduce uncertainty
  - get rid of redundancy

Knowledge representation

- Each source corresponds to a logic program \( \Pi_i \) (a finite set of rules).
- Integrity constraints correspond to a logic program \( \text{IC} \).
- All sources to be merged: a belief profile \( E = (\Pi_1, \ldots, \Pi_n) \) (a multiset of logic programs).
Goals

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

Knowledge representation

- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
- Integrity constraints correspond to a logic program $IC$.
- All sources to be merged is a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs).
Belief Change and Answer Set Programming
Merging logic programs
Goals and representation

Goals
- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

Knowledge representation
- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
- Integrity constraints correspond to a logic program $IC$.
- All sources to be merged: a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs).
Belief Change and Answer Set Programming

Merging logic programs

Goals and representation

**Goals**

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

**Knowledge representation**

- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
- Integrity constraints correspond to a logic program $IC$
- All sources to be merged: a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs)
Goals

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

Knowledge representation

- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
  - Integrity constraints correspond to a logic program $\mathcal{IC}$
  - All sources to be merged: a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs)
Goals

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

Knowledge representation

- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
- Integrity constraints correspond to a logic program $IC$
- All sources to be merged: a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs)
Goals

- obtain a global point of view from different sources
- solve conflicts
- reduce uncertainty
- get rid of redundancy

Knowledge representation

- Each source corresponds to a logic program $\Pi_i$ (a finite set of rules).
- Integrity constraints correspond to a logic program $IC$
- All sources to be merged: a belief profile $E = (\Pi_1, \ldots, \Pi_n)$ (a multiset of logic programs)
Semantic approach

[Delgrande et al. 2009], [Delgrande et al. 2009]

- select SE-models which are closest to the SE-models of the logic programs to be merged.

Syntactic approach

extending Removed Set Fusion to logic programs (ΠRSF)

- select subsets of rules to remove in order to restore consistency according to a fusion strategy \( P (\text{Card}, \Sigma, \text{Max}, \cdots) \)
Belief Change and Answer Set Programming
Merging logic programs
Goals and representation

Semantic approach

[Delgrande et al. 2009], [Delgrande et al. 2009]
- select SE-models which are closest to the SE-models of the logic programs to be merged.

Syntactic approach

extending Removed Set Fusion to logic programs ($\Pi$RSF)
- select subsets of rules to remove in order to restore consistency according to a fusion strategy $P$ ($\text{Card}$, $\Sigma$, $\text{Max}$, $\cdots$)
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

1 Introduction

2 Answer Set Programming

3 Merging logic programs

4 ΠRSF a syntactic framework for merging or revising logic programs
   - Main ideas & definitions
   - Strong merging
   - Problems with strong merging
   - Weak merging
   - Implementation

5 Conclusion
Questions

how to define the consistency of a logic program?

how to define the consequence relation between logic programs?

first solution

consistency:

Let $\Pi$ be a logic program. $\Pi$ is strongly consistent iff $AS(\Pi) \neq \emptyset$.

consequence relation:

$$\Pi_1 \models_s \Pi_2 \text{ iff } AS(\Pi_1) \subseteq AS(\Pi_2)$$
Questions

how to define the consistency of a logic program?

how to define the consequence relation between logic programs?

first solution

consistency:

Let $\Pi$ be a logic program. $\Pi$ is strongly consistent iff $\text{AS}(\Pi) \neq \emptyset$.

consequence relation:

$\Pi_1 \models_s \Pi_2$ iff $\text{AS}(\Pi_1) \subseteq \text{AS}(\Pi_2)$
Questions

how to define the consistency of a logic program?

how to define the consequence relation between logic programs?

first solution

consistency:

Let \( \Pi \) be a logic program. \( \Pi \) is strongly consistent iff \( AS(\Pi) \neq \emptyset \).

consequence relation:

\[
\Pi_1 \models_s \Pi_2 \quad \text{iff} \quad AS(\Pi_1) \subseteq AS(\Pi_2)
\]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Main ideas & definitions

Strong merging

\[ E = \{\Pi_1, \Pi_2, \Pi_3\} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs
Main ideas & definitions

Strong merging

\[ \Pi_1 \cup \Pi_2 \cup \Pi_3 \cup IC \]

\[ AS(\Pi_1 \cup \Pi_2 \cup \Pi_3 \cup IC) = \emptyset \]

\[ E = \{\Pi_1, \Pi_2, \Pi_3\} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Main ideas & definitions

**Strong merging**

\[ E = \{ \Pi_1, \Pi_2, \Pi_3 \} \]

\[ \Delta_{P, IC}^{\Pi_{RSF}, 1} (E) \]

\[ \text{AS}(((\Pi_1 \cup \Pi_2 \cup \Pi_3) \setminus R) \cup IC) \neq \emptyset \]
Special case: revision

\( E = \{ \Pi \} \)
Belief Change and Answer Set Programming
ΠRSF a syntactic framework for merging or revising logic programs
Main ideas & definitions

Special case: revision

\[ E = \{\Pi\} \]

\[ \Pi \cup IC \models \bot \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Main ideas & definitions

Special case: revision

\[ \Delta_{\text{Card}, IC}^{\Pi RSR, 1} \left( E \right) = \Pi \circ_{\Pi RSR} IC \]

\[ (\Pi \setminus R) \cup IC \not\models \bot \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Strong merging

\[ E = \{\Pi_1, \ldots, \Pi_n\} : \text{profile.} \quad \text{IC} : \text{integrity constraints.} \]

**Definition (Strong potential removed set)**

\( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a strong potential removed set of \( E \) iff

\[
\text{AS}(\{(\Pi_1 \cup \ldots \cup \Pi_n) \setminus X \} \cup \text{IC}) \neq \emptyset
\]

**Definition (Strong removed set according to \( P \))**

Let \( P \) be a strategy. \( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a strong removed set of \( E \) according to \( P \) iff

1. \( X \) is a strong potential removed set of \( E \), and
2. there is no strong potential removed set \( Y \) of \( E \) s.t. \( Y \subset X \), and
3. there is no strong potential removed set \( Y \) of \( E \) s.t. \( Y \prec_P X \).

\[ \mathcal{R}^1_P(E) : \text{the set of strong removed sets of } E \text{ according to } P. \]
\[ E = \{\Pi_1, \ldots, \Pi_n\} : \text{profile.} \quad IC : \text{integrity constraints.} \]

**Definition (Strong potential removed set)**

\( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a strong potential removed set of \( E \) iff

\[
\text{AS}(((\Pi_1 \cup \ldots \cup \Pi_n) \setminus X) \cup IC) \neq \emptyset
\]

**Definition (Strong removed set according to \( P \))**

Let \( P \) be a strategy. \( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a strong removed set of \( E \) according to \( P \) iff

1. \( X \) is a strong potential removed set of \( E \), and
2. there is no strong potential removed set \( Y \) of \( E \) s.t. \( Y \subset X \), and
3. there is no strong potential removed set \( Y \) of \( E \) s.t. \( Y <_P X \).

\( R^1_P(E) : \text{the set of strong removed sets of } E \text{ according to } P. \)
Belief Change and Answer Set Programming

\( \Pi RSF \) a syntactic framework for merging or revising logic programs

**Strong merging**

**Definition**

\[ E = (\Pi_1, \ldots, \Pi_n) \]

\( \mathcal{R}_P^1(E) \) : set of strong removed sets of \( E \) according to \( P \)

\( f \) : selection function

\( \Delta_{P, IC}^{\Pi RSF, 1}(E) = \left\{ ((\Pi_1 \cup \ldots \cup \Pi_n) \setminus f(\mathcal{R}_P^1(E))) \cup IC \right\} \)
Problem

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \quad IC = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

\[ E \cup IC \text{ is strongly consistent: } AS(E \cup IC) = \{\{a, \neg b\}\} \]

\[ \text{for any strategy } P, \Delta^{\Pi_{RSF},1}_{P,IC} = E \cup IC \]

\[ AS(IC) = \{c, \neg b\} \]

IC is contained in the merging result, but

\[ AS(E \cup IC) \not\subseteq AS(IC) \]

Temporary conclusion

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \),

\[ \Delta^{\Pi_{RSF},1}_{P,IC} \not\models_s IC !!! \]
Problem

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \]
\[ IC = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

- \( E \cup IC \) is strongly consistent: \( AS(E \cup IC) = \{\{a, \neg b\}\} \)
- for any strategy \( P \), \( \Delta_{P,IC}^{\PiRSF,1} = E \cup IC \)
- \( AS(IC) = \{c, \neg b\} \).
- \( IC \) is contained in the merging result, but \( AS(E \cup IC) \not\subseteq AS(IC) \).

Temporary conclusion

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \), then \( \Delta_{P,IC}^{\PiRSF,1} \not\models_s IC \) !!!
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Problem

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \]

\[ IC = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

- \( E \cup IC \) is strongly consistent: \( AS(E \cup IC) = \{\{a, \neg b\}\} \)
- for any strategy \( P \), \( \Delta_{\Pi RSF,1}^{P,IC} = E \cup IC \)
- \( AS(IC) = \{c, \neg b\} \).
- \( IC \) is contained in the merging result, but \( AS(E \cup IC) \nsubseteq AS(IC) \).

Temporary conclusion

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \),

then \( \Delta_{\Pi RSF,1}^{P,IC} \nvdash_s IC \) !!!
Problem

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \quad IC = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

- \( E \cup IC \) is strongly consistent: \( AS(E \cup IC) = \{\{a, \neg b\}\} \)
- for any strategy \( P \), \( \Delta_{\Pi^{RSF},1}^{P,IC} = E \cup IC \)
- \( AS(IC) = \{c, \neg b\} \).
- \( IC \) is contained in the merging result, but \( AS(E \cup IC) \not\subseteq AS(IC) \).

Temporary conclusion

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \),
then \( \Delta_{\Pi^{RSF},1}^{P,IC} \not\models_s IC \) !!!
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Problem

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \quad \text{IC} = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

- \( E \cup IC \) is strongly consistent: \( AS(E \cup IC) = \{\{a, \neg b\}\} \)
- for any strategy \( P \), \( \Delta^{\Pi_{RSF},1}_{P,IC} = E \cup IC \)
- \( AS(IC) = \{c, \neg b\} \).
- \( IC \) is contained in the merging result, but \( AS(E \cup IC) \not\subseteq AS(IC) \).

Temporary conclusion

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \),

then \( \Delta^{\Pi_{RSF},1}_{P,IC} \not\models_s IC \) !!!
Belief Change and Answer Set Programming
ΠRSF a syntactic framework for merging or revising logic programs
Problems with strong merging

**Problem**

Unability to define a satisfactory consequence relation based on answer sets.

\[ E = \{\{a.\}\} \quad \text{and} \quad IC = \{c \leftarrow \text{not } a. \quad \neg b.\} \]

- \( E \cup IC \) is strongly consistent: \( AS(E \cup IC) = \{\{a, \neg b\}\} \)
- for any strategy \( P \), \( \Delta_{P,IC}^{\PiRSF,1} = E \cup IC \)
- \( AS(IC) = \{c, \neg b\} \).
- \( IC \) is contained in the merging result, but \( AS(E \cup IC) \not\subseteq AS(IC) \).

**Temporary conclusion**

If \( \Pi_1 \models_s \Pi_2 \) is defined as \( AS(\Pi_1) \subseteq AS(\Pi_2) \), then \( \Delta_{P,IC}^{\PiRSF,1} \not\models_s IC \) !!!
Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a, \quad a \leftarrow b, \quad b \} \] does not have any answer set because its immediate consequence leads to \( \{ a, \neg a, b \} \).
Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a, \ a \leftarrow b, \ b \} \] does not have any answer set because its immediate consequence leads to

\[ \{ a, \neg a, b \} \]

The only way to restore consistency is to remove rules

\[ \Pi' = \{ a \leftarrow \text{not } b, \ b \leftarrow \text{not } c, \ c \leftarrow \text{not } a \} \] does not have any answer set: no set of atoms have a justification, does not lead to any inconsistent set of literals
Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a. \ a \leftarrow b. \ b. \} \]
does not have any answer set because its immediate consequence leads to

\[ \{a, \neg a, b\} \]

The only way to restore consistency is to remove rules

\[ \Pi' = \{ a \leftarrow \text{not } b. \ \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a \} \]
does not have any answer set:

no set of atoms have a justification;

does not lead to any inconsistent set of literals

Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a. \ a \leftarrow b. \ b. \} \]

does not have any answer set because its immediate consequence leads to

\[ \{a, \neg a, b\} \]

The only way to restore consistency is to remove rules

\[ \Pi' = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } c \quad c \leftarrow \text{not } a \} \]

does not have any answer set:

- no set of atoms have a justification,
- does not lead to any inconsistent set of literals
- adding a justification: 
  \[ AS(\Pi' \cup \{a.\}) = \{\{a, b\}\} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Two kinds of programs with no answer set:

\( \Pi = \{ \neg a. \ a \leftarrow b. \ b. \} \) does not have any answer set because its immediate consequence leads to:

\( \{a, \neg a, b\} \)

The only way to restore consistency is to remove rules

\( \Pi' = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } c \ c \leftarrow \text{not } a\} \) does not have any answer set:

- no set of atoms have a justification,
- does not lead to any inconsistent set of literals
- adding a justification:
  \( AS(\Pi' \cup \{a\}) = \{\{a, b\}\} \)
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a. \ a \leftarrow b. \ b. \} \]

does not have any answer set because its immediate consequence leads to

\[ \{ a, \neg a, b \} \]

The only way to restore consistency is to remove rules

\[ \Pi' = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a \} \]

does not have any answer set:

- no set of atoms have a justification,
- does not lead to any inconsistent set of literals
- adding a justification:
  \[ \text{AS}(\Pi' \cup \{ a. \}) = \{ \{ a, b \} \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a. \quad a \leftarrow b. \quad b. \} \]
does not have any answer set because its immediate consequence leads to

\[ \{a, \neg a, b\} \]

The only way to restore consistency is to remove rules

\[ \Pi' = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } c. \quad c \leftarrow \text{not } a\} \]
does not have any answer set:

- no set of atoms have a justification,
- does not lead to any inconsistent set of literals

adding a justification:

\[ AS(\Pi' \cup \{a.\}) = \{\{a, b\}\} \]
Two kinds of programs with no answer set:

\[ \Pi = \{ \neg a. \ a \leftarrow b. \ b. \} \] does not have any answer set because its immediate consequence leads to \[ \{ a, \neg a, b \} \]

The only way to restore consistency is to remove rules.

\[ \Pi' = \{ a \leftarrow not b. \ b \leftarrow not c. \ c \leftarrow not a \} \] does not have any answer set:
- no set of atoms have a justification,
- does not lead to any inconsistent set of literals
- adding a justification: \[ AS(\Pi' \cup \{ a. \}) = \{ \{ a, b \} \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

**Problem**

\[ \Delta_{\Pi_{RSF},1}^{P,IC} \] can remove rules which just lack justification, but can be useful later on.

**Solution**

Use alternative definition for consistency and consequence relation.

**consistency:**

Let \( \Pi \) be a logic program. \( \Pi \) is weakly consistent iff \( SE(\Pi) \neq \emptyset \).

**consequence relation:**

\[
\Pi_1 \models_{w} \Pi_2 \quad \text{iff} \quad SE(\Pi_1) \subseteq SE(\Pi_2)
\]
Problem

$\Delta^{\Pi_{RSF,1}}_{P,IC}$ can remove rules which just lack justification, but can be useful later on.

Solution

Use alternative definition for consistency and consequence relation.

consistency:

Let $\Pi$ be a logic program. $\Pi$ is weakly consistent iff $SE(\Pi) \neq \emptyset$.

consequence relation:

$$\Pi_1 \models_w \Pi_2 \text{ iff } SE(\Pi_1) \subseteq SE(\Pi_2)$$
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Example

\[ \Pi = \{a \leftarrow \neg b. \ b \leftarrow \neg c. \ c \leftarrow \neg a.\} \]

\[ AS(\Pi) = \emptyset \]

\[ SE(\Pi) = \{(\{b\}, \{a, b\}), (\{a, b\}, \{a, b\}), (\{c\}, \{b, c\}), (\{b, c\}, \{b, c\}), (\{a\}, \{a, c\}), (\{a, c\}, \{a, c\}), (\emptyset, \{a, b, c\}), (\{a\}, \{a, b, c\}), (\{b\}, \{a, b, c\}), (\{c\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}), (\{b, c\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\} \]

\[ AS(\Pi \cup \{a.\}) = \{a, b\} \]

\[ SE(\Pi \cup \{a.\}) = \{(\{a, b\}, \{a, b\}), (\{a\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\} \]
Example

\[
\Pi = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } c. \quad c \leftarrow \text{not } a.\} \\
\]

\[
\text{AS}(\Pi) = \emptyset
\]

\[
\text{SE}(\Pi) = \{(\{b\}, \{a, b\}), (\{a, b\}, \{a, b\}), (\{c\}, \{b, c\}), (\{b, c\}, \{b, c\}), (\{a\}, \{a, c\}), (\{a, c\}, \{a, c\}), (\emptyset, \{a, b, c\}), (\{a\}, \{a, b, c\}), (\{b\}, \{a, b, c\}), (\{c\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}), (\{b, c\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\}
\]

\[
\text{AS}(\Pi \cup \{a.\}) = \{a, b\}
\]

\[
\text{SE}(\Pi \cup \{a.\}) = \{(\{a, b\}, \{a, b\}), (\{a\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\}
\]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Problems with strong merging

Example

\[ \Pi = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } c. \quad c \leftarrow \text{not } a. \} \]

\[ \text{AS}(\Pi) = \emptyset \]

\[ \text{SE}(\Pi) = \{ (\{b\}, \{a, b\}), (\{a, b\}, \{a, b\}), (\{c\}, \{b, c\}), (\{b, c\}, \{b, c\}), (\{a\}, \{a, c\}), (\{a, c\}, \{a, c\}), (\emptyset, \{a, b, c\}), (\{a\}, \{a, b, c\}), (\{b\}, \{a, b, c\}), (\{c\}, \{a, b, c\}), (\{a\}, \{a, b, c\}), (\{b, c\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\}) \} \]

\[ \text{AS}(\Pi \cup \{a.\}) = \{a, b\} \]

\[ \text{SE}(\Pi \cup \{a.\}) = \{ (\{a, b\}, \{a, b\}), (\{a\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\}) \} \]
Example

\[ \Pi = \{ a \leftarrow \neg b. \quad b \leftarrow \neg c. \quad c \leftarrow \neg a. \} \]

\[ AS(\Pi) = \emptyset \]

\[ SE(\Pi) = \{(\{b\}, \{a, b\}), (\{a, b\}, \{a, b\}), (\{c\}, \{b, c\}), (\{b, c\}, \{b, c\}),
(\{a\}, \{a, c\}), (\{a, c\}, \{a, c\}), (\emptyset, \{a, b, c\}), (\{a\}, \{a, b, c\}),
(\{b\}, \{a, b, c\}), (\{c\}, \{a, b, c\}), (\{a\}, \{a, b, c\}),
(\{b, c\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\} \]

\[ AS(\Pi \cup \{a\}) = \{a, b\} \]

\[ SE(\Pi \cup \{a\}) = \{(\{a, b\}, \{a, b\}), (\{a\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}),
(\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\} \]
Example

\[ \Pi = \{ a \leftarrow \neg b. \quad b \leftarrow \neg c. \quad c \leftarrow \neg a. \} \]

\[ \text{AS}(\Pi) = \emptyset \]

\[ \text{SE}(\Pi) = \{(\{b\}, \{a, b\}), (\{a, b\}, \{a, b\}), (\{c\}, \{b, c\}), (\{b, c\}, \{b, c\}),
(\{a\}, \{a, c\}), (\{a, c\}, \{a, c\}), (\emptyset, \{a, b, c\}), (\{a\}, \{a, b, c\}),
(\{b\}, \{a, b, c\}), (\{c\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}),
(\{b, c\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\}\]

\[ \text{AS}(\Pi \cup \{a.\}) = \{a, b\} \]

\[ \text{SE}(\Pi \cup \{a.\}) = \{(\{a, b\}, \{a, b\}), (\{a\}, \{a, b, c\}), (\{a, b\}, \{a, b, c\}),
(\{a, c\}, \{a, b, c\}), (\{a, b, c\}, \{a, b, c\})\} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

\[ E = \{ \Pi_1, \ldots, \Pi_n \} \] : profile.

\[ IC \] : integrity constraints.

**Definition (Weak potential removed set)**

\( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a weak potential removed set of \( E \) iff

\[ SE(((\Pi_1 \cup \ldots \cup \Pi_n) \setminus X) \cup IC) \neq \emptyset \]

**Definition (Weak removed set according to \( P \))**

Let \( P \) be a strategy. \( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a weak removed set of \( E \) according to \( P \) iff

1. \( X \) is a weak potential removed set of \( E \), and
2. there is no weak potential removed set \( Y \) of \( E \) s.t. \( Y \subset X \), and
3. there is no weak potential removed set \( Y \) of \( E \) s.t. \( Y <_P X \).

\[ \mathcal{R}^2_P(E) \] : the set of weak removed sets of \( E \) according to \( P \).
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

\[ E = \{\Pi_1, \ldots, \Pi_n\} : \text{profile.} \quad IC : \text{integrity constraints.} \]

**Definition (Weak potential removed set)**

\[ X \subseteq (\Pi_1, \ldots, \Pi_n) \text{ is a weak potential removed set of } E \iff \]

\[ SE(((\Pi_1 \cup \ldots \cup \Pi_n) \setminus X) \cup IC) \neq \emptyset \]

**Definition (Weak removed set according to } P{**

Let \( P \) be a strategy. \( X \subseteq (\Pi_1, \ldots, \Pi_n) \) is a weak removed set of \( E \) according to \( P \) iff

1. \( X \) is a weak potential removed set of \( E \), and
2. there is no weak potential removed set \( Y \) of \( E \) s.t. \( Y \subset X \), and
3. there is no weak potential removed set \( Y \) of \( E \) s.t. \( Y <_P X \).

\[ \mathcal{R}_P^2(E) \text{: the set of weak removed sets of } E \text{ according to } P. \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

**Definition**

\[ E = (\Pi_1, \ldots, \Pi_n) \]

\[ \mathcal{R}_P^2(E) : \text{set of weak removed sets of } E \text{ according to } P \]

\[ f : \text{selection function} \]

Weak merging

\[ \Delta_{P, IC}^{\Pi_{RSF}, 2}(E) = \{((\Pi_1 \cup \ldots \cup \Pi_n) \setminus f(\mathcal{R}_P^2(E))) \cup IC\} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

$E = \{\Pi_1, \Pi_2\}$

$\Pi_1 = \{a \leftarrow \text{not } b. \quad c \leftarrow \text{not } a. \quad e. \quad d \leftarrow f.\}$

$\Pi_2 = \{b \leftarrow \text{not } c. \quad \neg d. \quad d \leftarrow e. \quad f.\}$

$IC = \{\top\}$

Preliminary remarks

- $\Pi_1 \cup \Pi_2$ leads to an inconsistent set $\{d, \neg d, e, f\}$.
- Rules with unjustified consequences do not play any role in this inconsistent set: $\Pi'_1 = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } c. \quad \neg d. \quad c \leftarrow \text{not } a.\}$
- Rules leading to inconsistent set: $\Pi'_2 = \{e. \quad d \leftarrow f. \quad \neg d. \quad d \leftarrow e.\}$
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \{ \begin{array}{l}
    a \leftarrow \text{not } b. \\
    c \leftarrow \text{not } a. \\
    e. \\
    d \leftarrow f. 
  \end{array} \] \]

\[ \Pi_2 = \{ \begin{array}{l}
    b \leftarrow \text{not } c. \\
    \neg d. \\
    d \leftarrow e. \\
    f. 
  \end{array} \] \]

\[ IC = \{ \top \} \]

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{ d, \neg d, e, f \} \).
- Rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{ a \leftarrow \text{not } b. b \leftarrow \text{not } c. c \leftarrow \text{not } a \} \).
- Rules leading to inconsistent set: \( \Pi'_j = \{ e. d \leftarrow f. \neg d. d \leftarrow e. f. \} \).
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{\Pi_1, \Pi_2\} \]
\[ \Pi_1 = \begin{cases} a \leftarrow \text{not } b. & c \leftarrow \text{not } a. \\ e. & d \leftarrow f. \end{cases} \]
\[ \Pi_2 = \begin{cases} b \leftarrow \text{not } c. & \neg d. \\ d \leftarrow e. & f. \end{cases} \]

\[ IC = \{\top\} \]

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{d, \neg d, e, f\} \).
- Rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a\} \)
- Rules leading to inconsistent set: \( \Pi'_j = \{e. \ d \leftarrow f. \ \neg d. \ d \leftarrow e. \ f.\} \)
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs
Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ \Pi_1 = \{ \begin{align*}
    a & \leftarrow \text{not } b. \\
    c & \leftarrow \text{not } a. \\
    e & . \\
    d & \leftarrow f. 
\end{align*} \} \]
\[ \Pi_2 = \{ \begin{align*}
    b & \leftarrow \text{not } c. \\
    d & \leftarrow e. \\
    f & . 
\end{align*} \} \]

IC = \{ \top \}

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{ d, \neg d, e, f \} \).
- rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a \} \)
- rules leading to inconsistent set: \( \Pi'_j = \{ e. \ d \leftarrow f. \ \neg d. \ d \leftarrow e. \ f. \} \)
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ \Pi_1 = \{ \begin{align*}
  &a \leftarrow \text{not } b. \\
  &c \leftarrow \text{not } a. \\
  &e. \\
  &d \leftarrow f.
\end{align*} \} \]
\[ \Pi_2 = \{ \begin{align*}
  &b \leftarrow \text{not } c. \\
  &\neg d. \\
  &d \leftarrow e. \\
  &f.
\end{align*} \} \]

\[ IC = \{ \top \} \]

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{ d, \neg d, e, f \} \).
- rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a \} \)
- rules leading to inconsistent set:

\[ \Pi'_j = \{ e. \ d \leftarrow f. \ \neg d. \ d \leftarrow e. \ f. \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \{ \begin{array}{l} a \leftarrow \neg b. \ c \leftarrow \neg a. \ e. \ d \leftarrow f. \end{array} \} \]

\[ \Pi_2 = \{ \begin{array}{l} b \leftarrow \neg c. \ \neg d. \ d \leftarrow e. \ f. \end{array} \} \]

IC = \{ \top \}

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{ d, \neg d, e, f \} \).
- rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{ a \leftarrow \neg b. b \leftarrow \neg c. c \leftarrow \neg a \} \)
- rules leading to inconsistent set:
  \[ \Pi'_j = \{ e. \ d \leftarrow f. \ \neg d. \ d \leftarrow e. \ f. \} \]

\[ \Delta_{\Pi_{RSF},1}^{\Pi,IC}(E) \] will pick up in
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{\Pi_1, \Pi_2\} \]
\[ \Pi_1 = \begin{cases} a \leftarrow \text{not } b. & c \leftarrow \text{not } a. \\ e. & d \leftarrow f. \end{cases} \]
\[ \Pi_2 = \begin{cases} b \leftarrow \text{not } c. & \neg d. \\ d \leftarrow e. & f. \end{cases} \]
\[ IC = \{\top\} \]

Preliminary remarks

- \( \Pi_1 \cup \Pi_2 \) leads to an inconsistent set \( \{d, \neg d, e, f\} \).
- Rules with unjustified consequences do not play any role in this inconsistent set: \( \Pi'_i = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } c. \ c \leftarrow \text{not } a\} \)
- Rules leading to inconsistent set: \( \Pi'_j = \{e. \ d \leftarrow f. \ \neg d. \ d \leftarrow e. \ f.\} \)

\[ \Delta^{\Pi_{RSF},2}_{\Pi_1,\Pi_2}(E) \] will pick up in
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{\Pi_1, \Pi_2\} \]

\[
\Pi_1 = \{ a \leftarrow \text{not } b. \quad c \leftarrow \text{not } a. \quad e. \quad d \leftarrow f. \} \quad \Pi_2 = \{ b \leftarrow \text{not } c. \quad \neg d. \quad d \leftarrow e. \quad f. \} \]

IC = \{\top\}

Strong potential removed sets

\[
\begin{align*}
&\{\neg d. \ a \leftarrow \text{not } b.\}, \{\neg d. \ b \leftarrow \text{not } c.\}, \\
&\{\neg d. \ c \leftarrow \text{not } a.\}, \{e. \ f. \ a \leftarrow \text{not } b.\}, \{e. \ f. \ b \leftarrow \text{not } c.\}, \\
&\{e. \ f. \ c \leftarrow \text{not } a.\}, \{e. \ d \leftarrow f. \ a \leftarrow \text{not } b.\}, \\
&\{e. \ d \leftarrow f. \ b \leftarrow \text{not } c.\}, \{e. \ d \leftarrow f. \ c \leftarrow \text{not } a.\}, \\
&\{d \leftarrow e. \ f. \ a \leftarrow \text{not } b.\}, \{d \leftarrow e. \ f. \ b \leftarrow \text{not } c.\}, \\
&\{d \leftarrow e. \ f. \ c \leftarrow \text{not } a.\}, \{d \leftarrow e. \ d \leftarrow f. \ a \leftarrow \text{not } b.\}, \\
&\{d \leftarrow e. \ d \leftarrow f. \ b \leftarrow \text{not } c.\}, \{d \leftarrow e. \ d \leftarrow f. \ c \leftarrow \text{not } a.\} \}
\end{align*}
\]
Belief Change and Answer Set Programming

PRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \{ a \leftarrow \neg b. \quad c \leftarrow \neg a. \quad e. \quad d \leftarrow f. \} \]

\[ \Pi_2 = \{ b \leftarrow \neg c. \quad \neg d. \quad d \leftarrow e. \quad f. \} \]

IC = \{ \top \} 

Strong potential removed sets

\{ \{ \neg d. a \leftarrow \neg b. \}, \{ \neg d. b \leftarrow \neg c. \}, \{ \neg d. c \leftarrow \neg a. \}, \{ e. f. a \leftarrow \neg b. \}, \{ e. f. b \leftarrow \neg c. \}, \{ e. f. c \leftarrow \neg a. \}, \{ e. d \leftarrow f. a \leftarrow \neg b. \}, \{ e. d \leftarrow f. b \leftarrow \neg c. \}, \{ e. d \leftarrow f. c \leftarrow \neg a. \}, \{ d \leftarrow e. f. a \leftarrow \neg b. \}, \{ d \leftarrow e. f. b \leftarrow \neg c. \}, \{ d \leftarrow e. f. c \leftarrow \neg a. \}, \{ d \leftarrow e. d \leftarrow f. a \leftarrow \neg b. \}, \{ d \leftarrow e. d \leftarrow f. b \leftarrow \neg c. \}, \{ d \leftarrow e. d \leftarrow f. c \leftarrow \neg a. \} \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ \Pi_1 = \left\{ \begin{array}{l}
    a \leftarrow \neg b. \\
    c \leftarrow \neg a. \\
    e. \\
    d \leftarrow f. 
\end{array} \right\} \]
\[ \Pi_2 = \left\{ \begin{array}{l}
    b \leftarrow \neg c. \\
    \neg d. \\
    d \leftarrow e. \\
    f. 
\end{array} \right\} \]

\[ IC = \{ \top \} \]

Strong removed sets (Σ and Card)

\{ \{\neg d. a \leftarrow \neg b.\}, \{\neg d. b \leftarrow \neg c.\}, \\
\{\neg d. c \leftarrow \neg a.\}, \{e. f. a \leftarrow \neg b.\}, \{e. f. b \leftarrow \neg c.\}, \\
\{e. f. c \leftarrow \neg a.\}, \{e. d \leftarrow f. a \leftarrow \neg b.\}, \\
\{e. d \leftarrow f. b \leftarrow \neg c.\}, \{e. d \leftarrow f. c \leftarrow \neg a.\}, \\
\{d \leftarrow e. f. a \leftarrow \neg b.\}, \{d \leftarrow e. f. b \leftarrow \neg c.\}, \\
\{d \leftarrow e. f. c \leftarrow \neg a.\}, \{d \leftarrow e. d \leftarrow f. a \leftarrow \neg b.\}, \\
\{d \leftarrow e. d \leftarrow f. b \leftarrow \neg c.\}, \{d \leftarrow e. d \leftarrow f. c \leftarrow \neg a.\} \} \}
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{\Pi_1, \Pi_2\} \]
\[ \Pi_1 = \{a \leftarrow \text{not } b. \ c \leftarrow \text{not } a. \ e. \ d \leftarrow f.\} \]
\[ \Pi_2 = \{b \leftarrow \text{not } c. \ \neg d. \ d \leftarrow e. \ f.\} \]
\[ IC = \{\top\} \]

Weak potential removed sets

\{\{\neg d.\}, \{e. \ f.\}, \{e. \ d \leftarrow f.\}, \{d \leftarrow e. \ f.\}, \{d \leftarrow e. \ d \leftarrow f.\}\} \]
Belief Change and Answer Set Programming
ΠRSF a syntactic framework for merging or revising logic programs

Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ \Pi_1 = \{ a \leftarrow \neg b. \quad c \leftarrow \neg a. \quad e. \quad d \leftarrow f. \} \]
\[ \Pi_2 = \{ b \leftarrow \neg c. \quad \neg d. \quad d \leftarrow e. \quad f. \} \]
\[ IC = \{ \top \} \]

Weak potential removed sets

\{ \{ \neg d. \}, \{ e. \quad f. \}, \{ e. \quad d \leftarrow f. \}, \{ d \leftarrow e. \quad f. \}, \{ d \leftarrow e. \quad d \leftarrow f. \} \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs
Weak merging

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ \Pi_1 = \left\{ \begin{array}{l}
  a \leftarrow \text{not } b. \\
  c \leftarrow \text{not } a. \\
  e. \\
  d \leftarrow f. 
\end{array} \right\} \]
\[ \Pi_2 = \left\{ \begin{array}{l}
  b \leftarrow \text{not } c. \\
  d \leftarrow e. \\
  f. 
\end{array} \right\} \]
\[ IC = \{ \top \} \]

Weak removed sets (\(\Sigma\) and \(\text{Card}\))

\[ \{ \{ \neg d. \}, \{ e. \ f. \}, \{ e. \ d \leftarrow f. \}, \{ d \leftarrow e. \ f. \}, \{ d \leftarrow e. \ d \leftarrow f. \} \} \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

\[ \Delta_{\Pi RSF,1}^P, IC(E) \]: Strong Merging

Generate a program \( PL^1(E \cup IC) \) s.t. the answer sets of the program correspond to the strong removed sets.

\[ \Delta_{\Pi RSF,2}^P, IC(E) \]: Weak Merging

Generate a program \( PL^2(E \cup IC) \) s.t. the answer sets of the program correspond to the weak removed sets.
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

\[ \Delta_{P,IC}^{\Pi_{RSF},1}(E) \]: Strong Merging

Generate a program \( PL^1(E \cup IC) \) s.t. the answer sets of the program correspond to the strong removed sets.

\[ \Delta_{P,IC}^{\Pi_{RSF},2}(E) \]: Weak Merging

Generate a program \( PL^2(E \cup IC) \) s.t. the answer sets of the program correspond to the weak removed sets.
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

Building $\mathit{PL}^2(E \cup IC)$: 3 stages

1. generate all sets of atoms which can lead to a $SE$-model of $E \cup IC$
2. detect rules which have to be removed
3. verify integrity constraints
Building $PL^2(E \cup IC)$: 3 stages

1. generate all sets of atoms which can lead to a $SE$-model of $E \cup IC$
2. detect rules which have to be removed
3. verify integrity constraints
Belief Change and Answer Set Programming
πRSF a syntactic framework for merging or revising logic programs
Implementation

Building $PL^2(E \cup IC)$: 3 stages

1. generate all sets of atoms which can lead to a $SE$-model of $E \cup IC$
2. detect rules which have to be removed
3. verify integrity constraints
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

Building $PL^2(E \cup IC)$: 3 stages

1. generate all sets of atoms which can lead to a $SE$-model of $E \cup IC$
2. detect rules which have to be removed
3. verify integrity constraints
Stage 1: generate candidate SE-models

- Let \( Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\} \). Each \( a' \) represents \( \neg a \) in \( E \cup IC \).
- \( Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\} \).
  \( \tilde{a} \) stands for \( not \ a \).
- for each \( a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC) \), add the 4 rules:
  \( a_S \leftarrow not \, \tilde{a}_S \). \( \tilde{a}_S \leftarrow not \, a_S \). \( a_E \leftarrow not \, \tilde{a}_E \). \( \tilde{a}_E \leftarrow not \, a_E \).
- avoid inconsistent sets of literals :
  \( \bot \leftarrow a_S, a'_S \). \( \bot \leftarrow a_E, a'_E \).
- withdraw \((X, Y)\) s.t. \( X \not\subseteq Y \).
  \( \bot \leftarrow a_S, \tilde{a}_E \).
Stage 1: generate candidate $SE$-models

Let $Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\}$. Each $a'$ represents $\neg a$ in $E \cup IC$.

- $Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\}$. $\tilde{a}$ stands for $\neg a$.
- For each $a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)$, add the 4 rules: $a_S \leftarrow \neg \tilde{a}_S$. $\tilde{a}_S \leftarrow \neg a_S$. $a_E \leftarrow \neg \tilde{a}_E$. $\tilde{a}_E \leftarrow \neg a_E$.
- Avoid inconsistent sets of literals: $\bot \leftarrow a_S, a'_S$. $\bot \leftarrow a_E, a'_E$.
- Withdraw $(X, Y)$ s.t. $X \not\subseteq Y$. $\bot \leftarrow a_S, \tilde{a}_E$. 
Stage 1: generate candidate SE-models

- Let $Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\}$. Each $a'$ represents $\neg a$ in $E \cup IC$.

- $Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\}$. $\tilde{a}$ stands for not $a$.

- for each $a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)$, add the 4 rules:
  
  $a_S \leftarrow \neg \tilde{a}_S$. $\tilde{a}_S \leftarrow \neg a_S$. $a_E \leftarrow \neg \tilde{a}_E$. $\tilde{a}_E \leftarrow \neg a_E$.

- avoid inconsistent sets of literals:
  
  $\bot \leftarrow a_S, a'_S$. $\bot \leftarrow a_E, a'_E$.

- withdraw $(X, Y)$ s.t. $X \nsubseteq Y$:
  
  $\bot \leftarrow a_S, \tilde{a}_E$. 
Stage 1: generate candidate SE-models

- Let $Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\}$. Each $a'$ represents $\neg a$ in $E \cup IC$.

- $Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\}$.
  $\tilde{a}$ stands for not $a$.

- for each $a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)$, add the 4 rules:
  $a_S \leftarrow \neg \tilde{a}_S$. $\tilde{a}_S \leftarrow \neg a_S$. $a_E \leftarrow \neg \tilde{a}_E$. $\tilde{a}_E \leftarrow \neg a_E$.

- avoid inconsistent sets of literals:
  $\bot \leftarrow a_S, a'_S$. $\bot \leftarrow a_E, a'_E$.

- withdraw $(X, Y)$ s.t. $X \not\subseteq Y$.
  $\bot \leftarrow a_S, \tilde{a}_E$. 
Stage 1: generate candidate SE-models

- Let \( Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\} \). Each \( a' \) represents \( \neg a \) in \( E \cup IC \).
- \( Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\} \).
  \( \tilde{a} \) stands for \( \text{not } a \).
- for each \( a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC) \), add the 4 rules: 
  \( a_S \leftarrow \text{not } \tilde{a}_S \). 
  \( \tilde{a}_S \leftarrow \text{not } a_S \). 
  \( a_E \leftarrow \text{not } \tilde{a}_E \). 
  \( \tilde{a}_E \leftarrow \text{not } a_E \).
- avoid inconsistent sets of literals:
  \( \bot \leftarrow a_S, a'_S \). 
  \( \bot \leftarrow a_E, a'_E \).
- withdraw \((X, Y)\) s.t. \( X \nsubseteq Y \).
  \( \bot \leftarrow a_S, \tilde{a}_E \).
Stage 1: generate candidate $SE$-models

- Let $Atoms'(E \cup IC) = \{a' \mid a \in E \cup IC\}$. Each $a'$ represents $\neg a$ in $E \cup IC$.
- $Atoms(PL^2(E \cup IC)) = \{a_S, a_E, \tilde{a}_S, \tilde{a}_E \mid a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)\}$. \(\tilde{a}\) stands for $not\ a$.
- For each $a \in Atoms(E \cup IC) \cup Atoms'(E \cup IC)$, add the 4 rules: $a_S \leftarrow not \ \tilde{a}_S$. $\tilde{a}_S \leftarrow not \ a_S$. $a_E \leftarrow not \ \tilde{a}_E$. $\tilde{a}_E \leftarrow not \ a_E$.
- Avoid inconsistent sets of literals:
  \(\bot \leftarrow a_S, a'_S\). \(\bot \leftarrow a_E, a'_E\).
- Withdraw $(X, Y)$ s.t. $X \nsubseteq Y$.
  \(\bot \leftarrow a_S, \tilde{a}_E\).
Stage 2: detect rules which have to be removed

- for each $r \in \Pi_1 \cup \ldots \cup \Pi_n$, introduce a new atom $rm_r$, whose presence in an answer set stands for the presence of $r$ in a potential removed set. $R^+ = \{rm_r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n\}$
- $S \subseteq \text{Atoms} \left( PL^2 \left( E \cup IC \right) \right)$, $\text{Rules}(S) = \{r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n, \; rm_r \in S\}$
- for each $rm_r$, add the rules:
  $rm_r \leftarrow \text{head}(\tilde{r}_E), \text{body}^+(r_E), \text{body}^-(\tilde{r}_E)$.
  $rm_r \leftarrow \text{head}(\tilde{r}_S), \text{body}^+(r_S), \text{body}^-(\tilde{r}_E)$.

$r_S$ (resp. $r_E$): from the rule $r$, each atom $a$ being replaced by $a_S$ (resp. $a_E$).

$\tilde{r}$: from the rule $r$, each atom $a$ being replaced by $\tilde{a}$. 
Stage 2: detect rules which have to be removed

- for each $r \in \Pi_1 \cup \ldots \cup \Pi_n$, introduce a new atom $rm_r$, whose presence in an answer set stands for the presence of $r$ in a potential removed set. $R^+ = \{ rm_r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n \}$

- $S \subseteq \text{Atoms}(PL^2(E \cup IC))$, 
  $\text{Rules}(S) = \{ r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n, \, \text{rm}_r \in S \}$

- for each $\text{rm}_r$, add the rules:
  $\text{rm}_r \leftarrow \text{head}(\tilde{r}_E), \, \text{body}^+(r_E), \, \text{body}^-(\tilde{r}_E)$.
  $\text{rm}_r \leftarrow \text{head}(\tilde{r}_S), \, \text{body}^+(r_S), \, \text{body}^-(\tilde{r}_E)$.

$r_S$ (resp. $r_E$): from the rule $r$, each atom $a$ being replaced by $a_S$ (resp. $a_E$).

$\tilde{r}$: from the rule $r$, each atom $a$ being replaced by $\tilde{a}$. 
Stage 2: detect rules which have to be removed

- for each \( r \in \Pi_1 \cup \ldots \cup \Pi_n \), introduce a new atom \( rm_r \), whose presence in an answer set stands for the presence of \( r \) in a potential removed set. \( R^+ = \{ rm_r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n \} \)

- \( S \subseteq Atoms(PL^2(E \cup IC)) \),
  
  \[ \text{Rules}(S) = \{ r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n, \; rm_r \in S \} \]

- for each \( rm_r \), add the rules:
  
  \[ rm_r \leftarrow \text{head}(\tilde{r}_E), \; \text{body}^+(r_E), \; \text{body}^-(\tilde{r}_E). \]
  
  \[ rm_r \leftarrow \text{head}(\tilde{r}_S), \; \text{body}^+(r_S), \; \text{body}^-(\tilde{r}_E). \]

\( r_S \) (resp. \( r_E \)) : from the rule \( r \), each atom \( a \) being replaced by \( a_S \) (resp. \( a_E \)).

\( \tilde{r} \) : from the rule \( r \), each atom \( a \) being replaced by \( \tilde{a} \).
Stage 2: detect rules which have to be removed

- For each \( r \in \Pi_1 \cup \ldots \cup \Pi_n \), introduce a new atom \( rm_r \), whose presence in a answer set stands for the presence of \( r \) in a potential removed set. \( R^+ = \{ rm_r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n \} \)

- \( S \subseteq Atoms(PL^2(E \cup IC)) \),
  \( Rules(S) = \{ r \mid r \in \Pi_1 \cup \ldots \cup \Pi_n, \; rm_r \in S \} \)

- For each \( rm_r \), add the rules:
  \( rm_r \leftarrow \text{head}(\tilde{r}_E), \text{body}^+(r_E), \text{body}^-(\tilde{r}_E). \)
  \( rm_r \leftarrow \text{head}(\tilde{r}_S), \text{body}^+(r_S), \text{body}^-(\tilde{r}_E). \)

\( r_S \) (resp. \( r_E \)) : from the rule \( r \), each atom \( a \) being replaced by \( a_S \) (resp. \( a_E \)).

\( \tilde{r} \) : from the rule \( r \), each atom \( a \) being replaced by \( \tilde{a} \).
Stage 3: verify integrity constraints

For each $r \in IC$, add the rules:

\[ \bot \leftarrow \text{head}(\tilde{r}_E), \text{body}^+(r_E), \text{body}^-(\tilde{r}_E). \]

\[ \bot \leftarrow \text{head}(\tilde{r}_S), \text{body}^+(r_S), \text{body}^-(\tilde{r}_E). \]
Proposition

$S$ is a answer set of $PL^2(E \cup IC)$ iff $Rules(S)$ is a weak potential removed set of $E$. 
Belief Change and Answer Set Programming

Encoding the strategy $P$

The merging strategy is encoded exactly as for the propositional merging operators using additional rules and optimization statements and leads to a logic program $\prod_{E,IC}^P$.

Proposition

$S$ is an answer set of $PL^2(E \cup IC) \cup \prod_{E,IC}^P$ iff $Rules(S)$ is a weak removed set of $E$ according to $P$. 
### Example

\[\begin{align*}
E &= \{\Pi_1, \Pi_2\} \\
\Pi_1 &= \begin{cases} 
  r_1 : a &\leftarrow \text{not } b. \\
  r_2 : c &\leftarrow \text{not } a. \\
  r_3 : e. \\
  r_4 : d &\leftarrow f. 
\end{cases} \\
\Pi_2 &= \begin{cases} 
  r_5 : b &\leftarrow \text{not } c. \\
  r_6 : \neg d. \\
  r_7 : d &\leftarrow e. \\
  r_8 : f.
\end{cases} \\
IC &= \{\top\}
\end{align*}\]

### Stage 1: atoms

\[
\text{Atoms}(PL^2(E \cup IC)) = \{a_S, a_E, b_S, b_E, c_S, c_E, d_S, d_E, \\
  d'_S, d'_E, e_S, e_E, f_S, f_E, \\
  \tilde{a}_S, \tilde{a}_E, \tilde{b}_S, \tilde{b}_E, \tilde{c}_S, \tilde{c}_E, \tilde{d}_S, \tilde{d}_E, \\
  \tilde{d'}_S, \tilde{d'}_E, \tilde{e}_S, \tilde{e}_E, \tilde{f}_S, \tilde{f}_E, \\
  rm_1, rm_2, rm_3, rm_4, rm_5, rm_6, rm_7, rm_8\}\]
**Belief Change and Answer Set Programming**

ΠRSF a syntactic framework for merging or revising logic programs

**Implementation**

**Example**

\[ E = \{\Pi_1, \Pi_2\}\]

\[ IC = \{\top\}\]

\[ \Pi_1 = \{ \]

\[ \begin{array}{l}
  r_1 : a \leftarrow \text{not } b. \\
  r_2 : c \leftarrow \text{not } a. \\
  r_3 : e. \\
  r_4 : d \leftarrow f.
\end{array} \]

\[ \Pi_2 = \{ \]

\[ \begin{array}{l}
  r_5 : b \leftarrow \text{not } c. \\
  r_6 : \neg d. \\
  r_7 : d \leftarrow e. \\
  r_8 : f.
\end{array} \]

**Stage 1: atoms**

\[ \text{Atoms}(PL^2(E \cup IC)) = \{ a_S, a_E, b_S, b_E, c_S, c_E, d_S, d_E, \]

\[ d'_S, d'_E, e_S, e_E, f_S, f_E, \]

\[ \tilde{a}_S, \tilde{a}_E, \tilde{b}_S, \tilde{b}_E, \tilde{c}_S, \tilde{c}_E, \tilde{d}_S, \tilde{d}_E, \]

\[ \tilde{d'}_S, \tilde{d'}_E, \tilde{e}_S, \tilde{e}_E, \tilde{f}_S, \tilde{f}_E, \]

\[ rm_1, rm_2, rm_3, rm_4, rm_5, rm_6, rm_7, rm_8 \} \]
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \left\{ \begin{array}{l}
  r_1 : a \leftarrow \text{not } b. \\
  r_2 : c \leftarrow \text{not } a. \\
  r_3 : e. \\
  r_4 : d \leftarrow f.
\end{array} \right\} \]

\[ \Pi_2 = \left\{ \begin{array}{l}
  r_5 : b \leftarrow \text{not } c. \\
  r_6 : \text{not } d. \\
  r_7 : d \leftarrow e. \\
  r_8 : f.
\end{array} \right\} \]

\[ IC = \{ \top \} \]

Stage 1: candidate SE-models & inconsistency avoidance

\[ a_S \leftarrow \text{not } \tilde{a}_S. \]
\[ b_S \leftarrow \text{not } \tilde{b}_S. \]
\[ c_S \leftarrow \text{not } \tilde{c}_S. \]
\[ d_S \leftarrow \text{not } \tilde{d}_S. \]
\[ e_S \leftarrow \text{not } \tilde{e}_S. \]
\[ f_S \leftarrow \text{not } \tilde{f}_S. \]
\[ d'_S \leftarrow \text{not } \tilde{d'}_S. \]
\[ \bot \leftarrow d_E, d'_E. \]
\[ \bot \leftarrow a_S, \tilde{a}_E. \]
\[ \bot \leftarrow e_S, \tilde{e}_E. \]
\[ a_E \leftarrow \text{not } \tilde{a}_E. \]
\[ b_E \leftarrow \text{not } \tilde{b}_E. \]
\[ c_E \leftarrow \text{not } \tilde{c}_E. \]
\[ d_E \leftarrow \text{not } \tilde{d}_E. \]
\[ e_E \leftarrow \text{not } \tilde{e}_E. \]
\[ f_E \leftarrow \text{not } \tilde{f}_E. \]
\[ d'_E \leftarrow \text{not } \tilde{d'}_E. \]
\[ \bot \leftarrow c_S, \tilde{c}_E. \]
\[ \bot \leftarrow d_S, \tilde{d}_E. \]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

Example

\[ E = \{ \Pi_1, \Pi_2 \} \]
\[ IC = \{ \top \} \]

\[ \Pi_1 = \{ 
  r_1 : a \leftarrow \text{not} \ b. 
  r_2 : c \leftarrow \text{not} \ a. 
  r_3 : e. 
  r_4 : d \leftarrow f. 
\} \]

\[ \Pi_2 = \{ 
  r_5 : b \leftarrow \text{not} \ c. 
  r_6 : \neg d. 
  r_7 : d \leftarrow e. 
  r_8 : f. 
\} \]

Stage 2: detect rules to be removed

\[ rm_1 \leftarrow \tilde{a}_E, \tilde{b}_E \]
\[ rm_2 \leftarrow \tilde{c}_E, \tilde{a}_E \]
\[ rm_3 \leftarrow \tilde{e}_E \]
\[ rm_4 \leftarrow \tilde{d}_E, f_E \]
\[ rm_5 \leftarrow \tilde{b}_E, \tilde{c}_E \]
\[ rm_6 \leftarrow \tilde{d}_E' \]
\[ rm_7 \leftarrow \tilde{d}_E, \tilde{e}_E \]
\[ rm_8 \leftarrow f_E \]
\[ rm_1 \leftarrow \tilde{a}_S, \tilde{b}_E \]
\[ rm_2 \leftarrow \tilde{c}_S, \tilde{a}_E \]
\[ rm_3 \leftarrow \tilde{e}_S \]
\[ rm_4 \leftarrow \tilde{d}_S, f_S \]
\[ rm_5 \leftarrow \tilde{b}_S, c_E \]
\[ rm_6 \leftarrow \tilde{d}_S' \]
\[ rm_7 \leftarrow \tilde{d}_S, \tilde{e}_S \]
\[ rm_8 \leftarrow f_S \]

No stage 3 in this example: no integrity constraints.
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ IC = \{ \top \} \]

\[ \Pi_1 = \{ \begin{align*}
    r_1 & : a \leftarrow \neg b. \\
    r_2 & : c \leftarrow \neg a. \\
    r_3 & : e. \\
    r_4 & : d \leftarrow f.
\end{align*} \} \]

\[ \Pi_2 = \{ \begin{align*}
    r_5 & : b \leftarrow \neg c. \\
    r_6 & : \neg d. \\
    r_7 & : d \leftarrow e. \\
    r_8 & : f.
\end{align*} \} \]

Stage 2: detect rules to be removed

\[ \begin{align*}
    rm_1 & \leftarrow \tilde{a}_E, \tilde{b}_E. \\
    rm_2 & \leftarrow \tilde{c}_E, \tilde{a}_E. \\
    rm_3 & \leftarrow \tilde{e}_E. \\
    rm_4 & \leftarrow \tilde{d}_E, f_E. \\
    rm_5 & \leftarrow \tilde{b}_E, \tilde{c}_E. \\
    rm_6 & \leftarrow \tilde{d}^\prime_E. \\
    rm_7 & \leftarrow \tilde{d}_E, \tilde{e}_E. \\
    rm_8 & \leftarrow f_E. \\
    rm_1 & \leftarrow \tilde{a}_S, \tilde{b}_E. \\
    rm_2 & \leftarrow \tilde{c}_S, \tilde{a}_E. \\
    rm_3 & \leftarrow \tilde{e}_S. \\
    rm_4 & \leftarrow \tilde{d}_S, f_S. \\
    rm_5 & \leftarrow \tilde{b}_S, c_E. \\
    rm_6 & \leftarrow \tilde{d}^\prime_S. \\
    rm_7 & \leftarrow \tilde{d}_S, \tilde{e}_S. \\
    rm_8 & \leftarrow f_S.
\end{align*} \]

No stage 3 in this example: no integrity constraints.
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

Results

- 1620 answer sets
- after projection on $R^+$: 5 set inclusion minimal weak potential removed sets.

\[
\begin{align*}
\{ rm_3, rm_8 \} & \quad \{ e. \ f. \} \\
\{ rm_7, rm_8 \} & \quad \{ d \leftarrow e. \ f. \} \\
\{ rm_4, rm_7 \} & \quad \{ d \leftarrow f. \ d \leftarrow e. \} \\
\{ rm_3, rm_4 \} & \quad \{ e. \ d \leftarrow f. \} \\
\{ rm_6 \} & \quad \{ \neg d. \}
\end{align*}
\]
1620 answer sets

- after projection on $R^+$: 5 set inclusion minimal weak potential removed sets.

\[
\begin{align*}
\{rm_3, rm_8\} & \quad \{e.\ f.\} \\
\{rm_7, rm_8\} & \quad \{d \leftarrow e.\ f.\} \\
\{rm_4, rm_7\} & \quad \{d \leftarrow f.\ d \leftarrow e.\} \\
\{rm_3, rm_4\} & \quad \{e.\ d \leftarrow f.\} \\
\{rm_6\} & \quad \{\neg d.\}
\end{align*}
\]
Results

- 1620 answer sets
- after projection on $R^+$: 5 set inclusion minimal weak potential removed sets.

\[
\begin{align*}
\{rm_3, rm_8\} & \quad \{e. \ f.\} \\
\{rm_7, rm_8\} & \quad \{d \leftarrow e. \ f.\} \\
\{rm_4, rm_7\} & \quad \{d \leftarrow f. \ d \leftarrow e.\} \\
\{rm_3, rm_4\} & \quad \{e. \ d \leftarrow f.\} \\
\{rm_6\} & \quad \{\neg d.\}
\end{align*}
\]
Belief Change and Answer Set Programming

ΠRSF a syntactic framework for merging or revising logic programs

Implementation

Results

- 1620 answer sets
- after projection on $R^+$: 5 set inclusion minimal weak potential removed sets.

\[
\begin{align*}
\{rm_3, rm_8\} & \quad \{e. \ f.\} \\
\{rm_7, rm_8\} & \quad \{d \leftarrow e. \ f.\} \\
\{rm_4, rm_7\} & \quad \{d \leftarrow f. \ d \leftarrow e.\} \\
\{rm_3, rm_4\} & \quad \{e. \ d \leftarrow f.\} \\
\{rm_6\} & \quad \{\neg d.\}
\end{align*}
\]
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \begin{cases} 
    r_1 & a \leftarrow \text{not } b. \\
    r_2 & c \leftarrow \text{not } a. \\
    r_3 & e. \\
    r_4 & d \leftarrow f. 
\end{cases} \]

\[ \Pi_2 = \begin{cases} 
    r_5 & b \leftarrow \text{not } c. \\
    r_6 & \neg d. \\
    r_7 & d \leftarrow e. \\
    r_8 & f. 
\end{cases} \]

\[ IC = \{ \top \} \]

Σ & Card strategy

\{ #\text{minimize}\{ rm_1, rm_2, rm_3, rm_4, rm_5, rm_6, rm_7, rm_8 \}. \}

1 Optimized answer set.

\{ rm_6 \}
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ IC = \{ \top \} \]

\[ \Pi_1 = \{ \]

\[ r_1 : a \leftarrow \text{not } b. \]

\[ r_2 : c \leftarrow \text{not } a. \]

\[ r_3 : e. \]

\[ r_4 : d \leftarrow f. \]

\[ \Pi_2 = \{ \]

\[ r_5 : b \leftarrow \text{not } c. \]

\[ r_6 : \neg d. \]

\[ r_7 : d \leftarrow e. \]

\[ r_8 : f. \]

\[ \Sigma & \text{ Card strategy} \]

\[ \{ \# \text{minimize}\{ r_{m1}, r_{m2}, r_{m3}, r_{m4}, r_{m5}, r_{m6}, r_{m7}, r_{m8} \} \}. \]

1 Optimized answer set.

\[ \{ r_{m6} \} \]
Example

\[ E = \{ \Pi_1, \Pi_2 \} \]

\[ \Pi_1 = \{ \begin{align*}
  r_1 &: \ a \leftarrow \text{not} \ b. \\
  r_2 &: \ c \leftarrow \text{not} \ a. \\
  r_3 &: \ e. \\
  r_4 &: \ d \leftarrow \ f. 
\end{align*} \} \]

\[ \Pi_2 = \{ \begin{align*}
  r_5 &: \ b \leftarrow \text{not} \ c. \\
  r_6 &: \ \neg d. \\
  r_7 &: \ d \leftarrow \ e. \\
  r_8 &: \ f. 
\end{align*} \} \]

\[ IC = \{ \top \} \]

\[ \Sigma \ & \text{Card} \ \text{strategy} \]

\[ \{ \# \text{minimize} \{ rm_1, rm_2, rm_3, rm_4, rm_5, rm_6, rm_7, rm_8 \} \}. \]

1 Optimized answer set.

\[ \{ rm_6 \} \]
Conclusion

Belief change: revision, fusion

nature of change under consideration [Delgrande et al. 2006]

example: revision

1) revision as defeasible inference
2) changing the plausibility ordering in presence of new evidence
3) revising generic knowledge

ASP: unified formalism for both representing and implementing change operations also allows for 3)
AGM, KM and KP postulates:

- focus on 2)
- new postulates for revising, merging logic programs?
Conclusion

- semantic characterization of ΠRSF
- extension of ΠRSF to logic programs with preferences
- experimental study
- necessity of benchmarks? : (ASPARAGUS)