Development of a Belief Merging Framework for dlvhex

Christoph Redl

May 31, 2010

Outline

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base
- In many applications, knowledge sources are often provided by third-parties

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base
- In many applications, knowledge sources are often provided by third-parties
- ... but rarely synchronized

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base
- In many applications, knowledge sources are often provided by third-parties
- ... but rarely synchronized

Combining the contents

- We do not want to restrict ourselves to one source

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base
- In many applications, knowledge sources are often provided by third-parties
- ... but rarely synchronized

Combining the contents

- We do not want to restrict ourselves to one source
- Naive union can introduce contradictions

Motivation

Usage of multiple belief bases

- Usually we have more than one knowledge base
- In many applications, knowledge sources are often provided by third-parties
- ... but rarely synchronized

Combining the contents

- We do not want to restrict ourselves to one source
- Naive union can introduce contradictions
- Many different merging techniques

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: “minimal change”

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: “minimal change”

Belief Merging

- *Aggregation* of sources
- No single formula with absolute priority

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: “minimal change”

Belief Merging

- *Aggregation* of sources
- No single formula with absolute priority
- Variants of AGM postulates are reasonable:
  informally, minimize differences between sources and merged belief base

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: "minimal change"

Belief Merging

- *Aggregation* of sources
- No single formula with absolute priority
- Variants of AGM postulates are reasonable:
  informally, minimize differences between sources and merged belief base

Applications

- judgment aggregation

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: “minimal change”

Belief Merging

- *Aggregation* of sources
- No single formula with absolute priority
- Variants of AGM postulates are reasonable:
  informally, minimize differences between sources and merged belief base

Applications

- judgment aggregation
- merging of decision diagrams

Motivation

Belief Revision

- Incorporation of knowledge into existing belief base
- AGM postulates: “minimal change”

Belief Merging

- *Aggregation* of sources
- No single formula with absolute priority
- Variants of AGM postulates are reasonable:
  informally, minimize differences between sources and merged belief base

Applications

- judgment aggregation
- merging of decision diagrams
- fusion of business databases

Types of Incompatibility

Syntactic Incompatibility

- Sources are written in different formalisms
- Preprocessing step needed

Types of Incompatibility

Syntactic Incompatibility

- Sources are written in different formalisms
- Preprocessing step needed

Examples

- relational databases
- object-orientated databases
- RDF ontologies
- logic programs

Types of Incompatibility

Logic Inconsistencies

Union of data sets leads to contradictions:

$$\bigcup_i KB_i \models \bot$$

Types of Incompatibility

Logic Inconsistencies

Union of data sets leads to contradictions:

$$\bigcup_i KB_i \models \bot$$

likewise

constraint violation: $\bigcup_i KB_i \not\models C$

Types of Incompatibility

Logic Inconsistencies

Union of data sets leads to contradictions:

$$\bigcup_i KB_i \models \bot$$

likewise

constraint violation: $\bigcup_i KB_i \not\models C$

Example

name is the primary key; a constraint forces the height to be unique for each person.

(a)

name	height
Marge	1,78m
Homer	1,82m
Bart	1,67m

(b)

name	height
Marge	1,78m
Homer	1,82m
Bart	1,65m

Data Cleanness

- Remain after logic inconsistencies resolved
- Detection requires advanced algorithms: data cleansing

Data Cleanness

- Remain after logic inconsistencies resolved
- Detection requires advanced algorithms: data cleansing

Undesired artefacts concerning

- Differing naming conventions
  e.g., academic degrees, addresses, ...
- Different entries referring to the same real-world object

Data Cleanness

- Remain after logic inconsistencies resolved

- Detection requires advanced algorithms: data cleansing

Undesired artefacts concerning

- Differing naming conventions
  e.g., academic degrees, addresses, ...

- Different entries referring to the same real-world object

Example

Merging of address tables, one with and one without abbreviations

Logic Programs as Belief Bases

Given

$\pi = (P_1, \ldots, P_n)$ vector of belief bases

Given as *logic programs* with answer sets $AS(P_i)$

Logic Programs as Belief Bases

Given
$\pi = (P_1, \ldots, P_n)$ vector of belief bases
Given as *logic programs* with answer sets $AS(P_i)$
Answer sets of programs are considered to be the stored knowledge

Logic Programs as Belief Bases

Given

$\pi = (P_1, \ldots, P_n)$ vector of belief bases

Given as *logic programs* with answer sets $AS(P_i)$

Answer sets of programs are considered to be the stored knowledge

To define

$\Sigma^C = (\Sigma^C_c, \Sigma^C_p)$	common signature
$\mu = (\mu_1, \ldots, \mu_n)$	vector of mapping functions
$\omega = (\circ_1, \ldots, \circ_m)$	merging operators
R	merging plan

Common Signature and Mappings

Solve the problem of syntactic incompatibility

Common Signature and Mappings

Solve the problem of syntactic incompatibility

Common Signature

- $\Sigma^C = (\Sigma_c^C, \Sigma_p^C)$
- expressive enough to represent any of sources

Common Signature and Mappings

Solve the problem of syntactic incompatibility

Common Signature

- $\Sigma^C = (\Sigma_c^C, \Sigma_p^C)$
- expressive enough to represent any of sources

Mappings

- Let $\mathcal{A} = 2^{Lit_{\Sigma^C}}$ (set of potential answer sets over Σ^C)

Common Signature and Mappings

Solve the problem of syntactic incompatibility

Common Signature

- $\Sigma^C = (\Sigma_c^C, \Sigma_p^C)$
- expressive enough to represent any of sources

Mappings

- Let $\mathcal{A} = 2^{Lit_{\Sigma^C}}$ (set of potential answer sets over Σ^C)
- $\mu_i : \{AS(P_i)\} \to 2^{\mathcal{A}}$

Common Signature and Mappings

Solve the problem of syntactic incompatibility

Common Signature

- $\Sigma^C = (\Sigma_c^C, \Sigma_p^C)$
- expressive enough to represent any of sources

Mappings

- Let $\mathcal{A} = 2^{Lit_{\Sigma^C}}$ (set of potential answer sets over Σ^C)
- $\mu_i : \{AS(P_i)\} \to 2^{\mathcal{A}}$
- answer sets stay semantically equivalent!

Merging Operators

Resolve logic inconsistencies
(plus: may perform data cleansing tasks)

Merging Operators

Resolve logic inconsistencies
(plus: may perform data cleansing tasks)

$$\circ_i^{n,m} : \underbrace{2^{\mathcal{A}} \times \cdots \times 2^{\mathcal{A}}}_{n\ times} \times \underbrace{\mathcal{D}_1 \times \ldots \times \mathcal{D}_m}_{additional\ parameters} \rightarrow 2^{\mathcal{A}}$$

Merging Operators

Resolve logic inconsistencies
(plus: may perform data cleansing tasks)

$$\circ_i^{n,m} : \underbrace{2^{\mathcal{A}} \times \cdots \times 2^{\mathcal{A}}}_{n \ times} \times \underbrace{\mathcal{D}_1 \times \ldots \times \mathcal{D}_m}_{additional \ parameters} \rightarrow 2^{\mathcal{A}}$$

Example for a merging operator

The union operator $\circ_{\cup}^{2,0}$ is defined as follows:

$$\circ_{\cup}^{2,0} : 2^{\mathcal{A}} \times 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$$

$$\circ_{\cup}^{2,0}(SAS_1, SAS_2) = \{AS_1 \cup AS_2 | AS_1 \in SAS_1, AS_2 \in SAS_2, AS_1 \cup AS_2 \not\models \perp\}$$

($\circ_{\cup}^{2}$ is binary, no additional parameters)

Merging Plans

A merging plan is hierarchical and defines

- the order
- of operators
- to be applied on which belief bases

Merging Plans

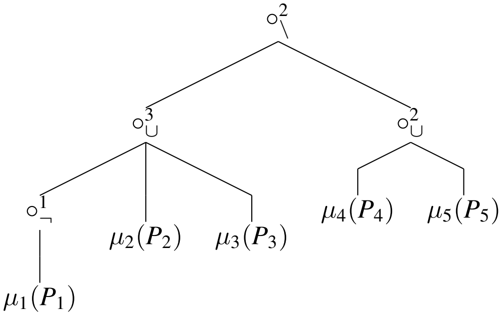
A merging plan is hierarchical and defines

- the order
- of operators
- to be applied on which belief bases

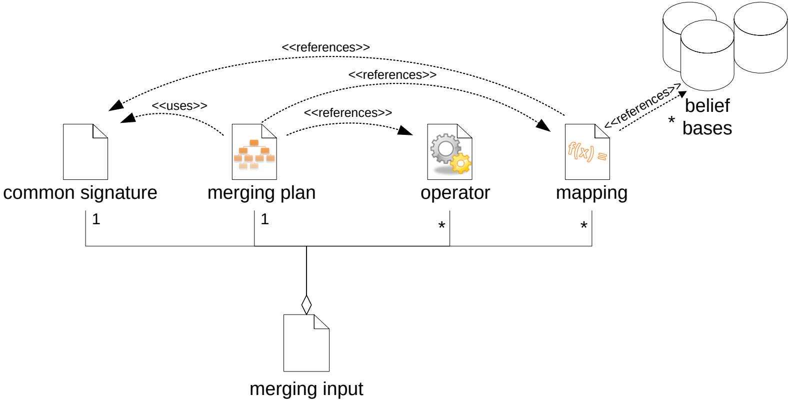
The result

- the set of answer sets delivered by the topmost operator

Example merging plan



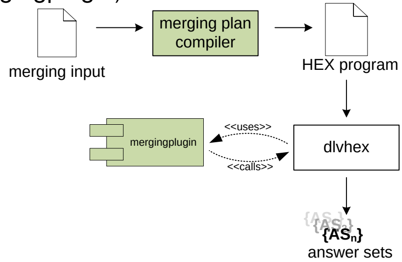
Merging Input



Approach

Steps

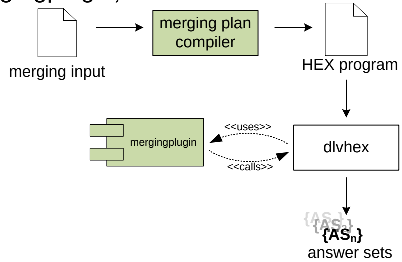
1. designing a merging language
2. implementing the merging plan compiler
3. implementing external atoms (mergingplugin)



Approach

Steps

1. designing a merging language
2. implementing the merging plan compiler
3. implementing external atoms (mergingplugin)



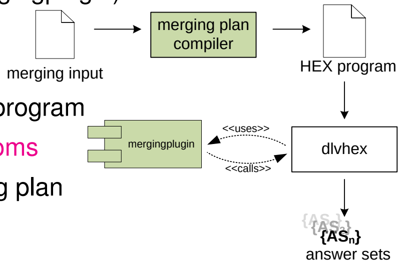
Approach

Steps

1. designing a merging language
2. implementing the merging plan compiler
3. implementing external atoms (mergingplugin)

merging plan compiler

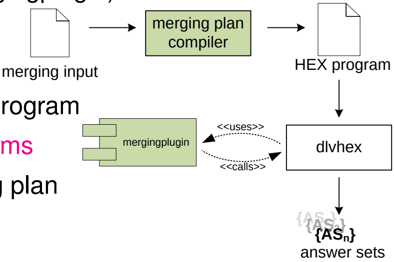
- translates merging plan into HEX program
- program makes use of external atoms
- answer sets = result of the merging plan



Approach

Steps

1. designing a merging language
2. implementing the merging plan compiler
3. implementing external atoms (mergingplugin)



merging plan compiler

- translates merging plan into HEX program
- program makes use of external atoms
- answer sets = result of the merging plan

mergingplugin

defines external atoms for:

- calling of nested HEX programs
- calling of merging operators

Advantages of the Framework

- Rapid prototyping

Advantages of the Framework

- Rapid prototyping
- Routine tasks like information flow management is done automatically

Advantages of the Framework

- Rapid prototyping
- Routine tasks like information flow management is done automatically
- Experiment with different merging plans and operators by parameterizing them

Advantages of the Framework

- Rapid prototyping
- Routine tasks like information flow management is done automatically
- Experiment with different merging plans and operators by parameterizing them
- Develop merging operators once, apply them in many scenarios

Using the Framework

Steps

1. Define your merging task “`merging.mp`”
2. Run the merging plan compiler (`mpcompiler`) on this input
3. Execute the result by dlvhex

Using the Framework

Steps

1. Define your merging task “`merging.mp`”
2. Run the merging plan compiler (`mpcompiler`) on this input
3. Execute the result by dlvhex
4. (filter the output since the translation of merging plans in HEX programs requires the derivation of some intermediate atoms)

Using the Framework

Steps

1. Define your merging task "`merging.mp`"
2. Run the merging plan compiler (`mpcompiler`) on this input
3. Execute the result by dlvhex
4. (filter the output since the translation of merging plans in HEX programs requires the derivation of some intermediate atoms)

Typical call

- Command-line:

```
$ mpcompiler merging.mp | dlvhex --filter=a,b,c --
```

Using the Framework

Steps

1. Define your merging task "`merging.mp`"
2. Run the merging plan compiler (`mpcompiler`) on this input
3. Execute the result by dlvhex
4. (filter the output since the translation of merging plans in HEX programs requires the derivation of some intermediate atoms)

Typical call

- Command-line:
```
$ mpcompiler merging.mp | dlvhex --filter=a,b,c --
```
- Alternatively:
```
$ dlvhex --merging --filter=a,b,c merging.mp
```

Merging plan language: `merging.mp`

```
[common signature]
    predicate: a/0;
    predicate: b/0;
    predicate: c/0;
    predicate: p/1;
    predicate: q/3;

[belief base]
    name:bb1;
    mapping: “some_rule.”;        % query external source here
    mapping: “q(X, Y, Z) :- &rdf[“...”](X, Y, Z).”;

[belief base]
    name:bb2;
    source: “some_program.hex”;     % or within this program
...
```

Merging plan language: `merging.mp` (ctn'd.)

```
[merging plan]
{
    operator: setminus;
        {
            operator: union;
                {
                    operator: neg;
                        {bb1};
                };
                {bb2};
                {bb3};
        };
        {
        operator: union;
            {bb4};
            {bb5};
        };
}
```

Dalal's Operator

Definition of implemented version

- Belief bases $K = (AS(P_1), \ldots, AS(P_n))$
  $\mathcal{A}$ = set of all potential answer sets

Dalal's Operator

Definition of implemented version

- Belief bases $K = (AS(P_1), \ldots, AS(P_n))$
  $\mathcal{A}$ = set of all potential answer sets
- Answer set distance function: $\underline{d} : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$

Dalal's Operator

Definition of implemented version

- Belief bases $K = (AS(P_1), \ldots, AS(P_n))$
  $\mathcal{A}$ = set of all potential answer sets
- Answer set distance function: $\underline{d} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}$
- Answer set to belief base-distances:

$$d(A, P_i) = \min_{J \in AS(P_i)} \underline{d}(A, J)$$

Dalal's Operator

Definition of implemented version

- Belief bases $K = (AS(P_1), \ldots, AS(P_n))$
  $\mathcal{A}$ = set of all potential answer sets
- Answer set distance function: $\underline{d} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}$
- Answer set to belief base-distances:

$$d(A, P_i) = \min_{J \in AS(P_i)} \underline{d}(A, J)$$

- Aggregate function: $D : \mathbb{R}^n \rightarrow \mathbb{R}$

$$D^d(A, K) = D(d(A, P_1), \ldots, d(A, P_n))$$

Dalal's Operator

Definition of implemented version

- Belief bases $K = (AS(P_1), \ldots, AS(P_n))$
  $\mathcal{A}$ = set of all potential answer sets
- Answer set distance function: $\underline{d} : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$
- Answer set to belief base-distances:

$$d(A, P_i) = \min_{J \in AS(P_i)} \underline{d}(A, J)$$

- Aggregate function: $D : \mathbb{R}^n \to \mathbb{R}$

$$D^d(A, K) = D(d(A, P_1), \ldots, d(A, P_n))$$

- $\circ^n(K) = \arg \min_{G \in \mathcal{A}: consistent} D^d(G, K)$

Fault Diagnosis

Finding an explanation for some observation

Definition

Propositional abduction problem (PAP): $\mathscr{P} = \langle V, H, M, T \rangle$

- V is a finite set of propositional variables
- $H \subseteq V$ is a set of hypothesis
- $M \subseteq V$ is the set of manifestations
- T is a consistent theory

Fault Diagnosis

Finding an explanation for some observation

Definition

Propositional abduction problem (PAP): $\mathscr{P} = \langle V, H, M, T \rangle$

- V is a finite set of propositional variables
- $H \subseteq V$ is a set of hypothesis
- $M \subseteq V$ is the set of manifestations
- T is a consistent theory

$S \subseteq H$ is a solution iff $T \cup S$ is consistent and $T \cup S \models M$

Fault Diagnosis

Finding an explanation for some observation

Definition

Propositional abduction problem (PAP): $\mathscr{P} = \langle V, H, M, T \rangle$

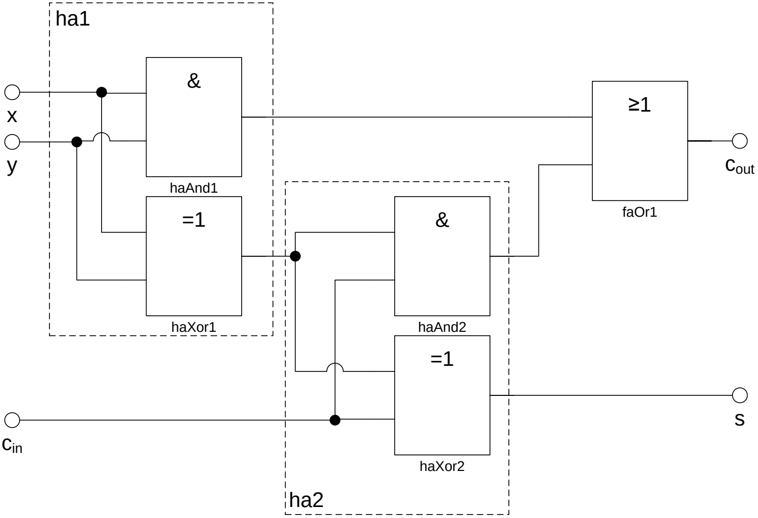
- V is a finite set of propositional variables
- $H \subseteq V$ is a set of hypothesis
- $M \subseteq V$ is the set of manifestations
- T is a consistent theory

$S \subseteq H$ is a solution iff $T \cup S$ is consistent and $T \cup S \models M$

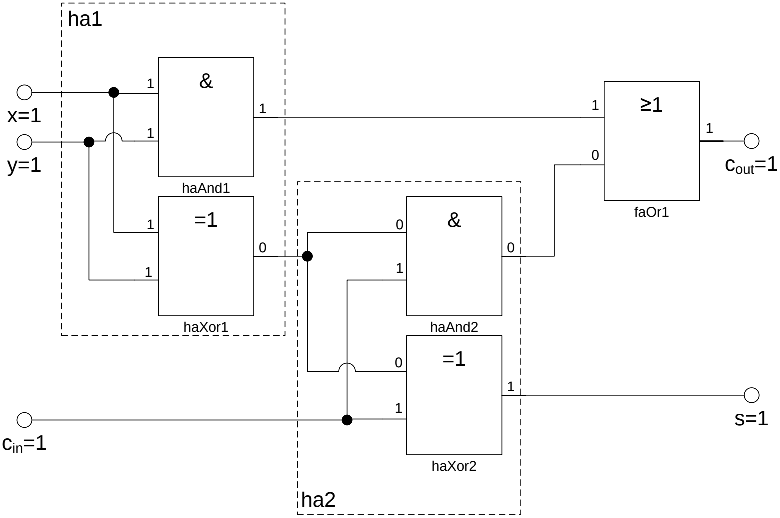
Challenge: *multiple* experts with *different* explanations S_i
Task: Finding a group decision S_G s.t.

- S_G is a solution to $\mathscr{P}$
- S_G is as similar to S_i $\forall i$ as possible

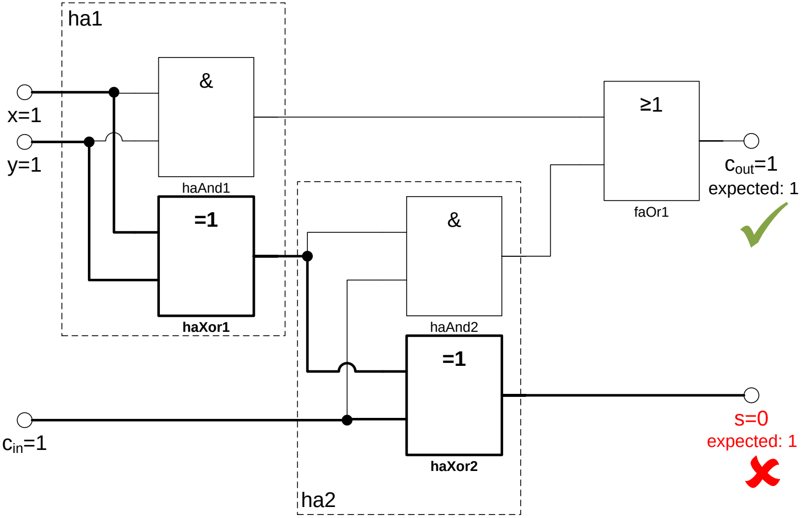
Full Adder



Full Adder - Example interpretation



Full Adder - Malfunctioning



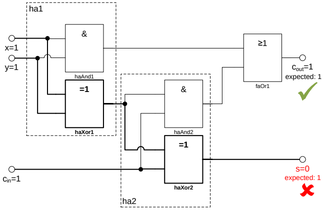
Full Adder

Implemented as logic program “fulladder.dl” (theory) with observations “fault.obs”

Full Adder

Implemented as logic program “fulladder.dl” (theory)
with observations “fault.obs”
Suppose we have 3 experts with different hypotheses

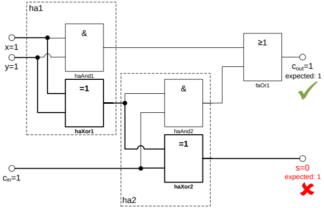
1. $ab(haAnd1).ab(haXor1).ab(haAnd2).ab(haXor2).ab(faOr1).$



Full Adder

Implemented as logic program “fulladder.dl” (theory)
with observations “fault.obs”
Suppose we have 3 experts with different hypotheses

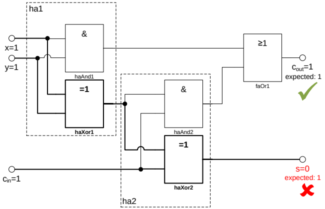
1. $ab(haAnd1).ab(haXor1).ab(haAnd2).ab(haXor2).ab(faOr1).$
2. $ab(haAnd1).ab(haAnd2).ab(haXor2).ab(faOr1).$
   no ab(haXor1)!



Full Adder

Implemented as logic program “fulladder.dl” (theory)
with observations “fault.obs”
Suppose we have 3 experts with different hypotheses

1. $ab(haAnd1).ab(haXor1).ab(haAnd2).ab(haXor2).ab(faOr1).$
2. $ab(haAnd1).ab(haAnd2).ab(haXor2).ab(faOr1).$
   no ab(haXor1)!
3. $ab(haAnd1).ab(haAnd2).ab(haXor2).ab(faOr1).$
   no ab(haXor2)!



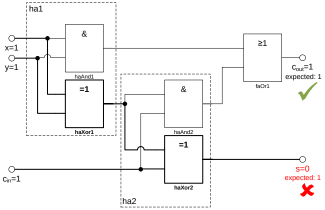
Full Adder

Implemented as logic program "fulladder.dl" (theory)
with observations "fault.obs"
Suppose we have 3 experts with different hypotheses

1. $ab(haAnd1).ab(haXor1).ab(haAnd2).ab(haXor2).ab(faOr1).$
2. $ab(haAnd1).ab(haAnd2).ab(haXor2).ab(faOr1).$
   no ab(haXor1)!
3. $ab(haAnd1).ab(haAnd2).ab(haXor2).ab(faOr1).$
   no ab(haXor2)!

(Minimal) Solutions

1. $AS(P_{J_1}) = \{\{ab(haXor1)\}, \{ab(haXor2)\}\}$
2. $AS(P_{J_2}) = \{\{ab(haXor2)\}\}$
3. $AS(P_{J_3}) = \{\{ab(haXor1)\}\}$



Full Adder - Merging individual decisions

Requirements

The group decision must be an explanation
and should be similar to the individual's

Full Adder - Merging individual decisions

Requirements

The group decision must be an explanation
and should be similar to the individual's

Distance function

I = individual explanation
G = group explanation

1. Penalize ignoring of individual beliefs, i.e.,
   $a \in I \wedge a \notin G$
   $\neg a \in I \wedge \neg a \notin G$

Full Adder - Merging individual decisions

Requirements

The group decision must be an explanation
and should be similar to the individual's

Distance function

I = individual explanation
G = group explanation

1. Penalize ignoring of individual beliefs, i.e.,
   $a \in I \land a \notin G$
   $\neg a \in I \land \neg a \notin G$
2. Penalize group beliefs unfounded for an individual, i.e.,
   $a \in G \land a \notin I$
   $\neg a \in G \land \neg a \notin I$

Full Adder - Merging individual decisions

Requirements

The group decision must be an explanation
and should be similar to the individual's

Distance function

I = individual explanation
G = group explanation

1. Penalize ignoring of individual beliefs, i.e.,
   $a \in I \wedge a \notin G$
   $\neg a \in I \wedge \neg a \notin G$
2. Penalize group beliefs unfounded for an individual, i.e.,
   $a \in G \wedge a \notin I$
   $\neg a \in G \wedge \neg a \notin I$
3. Penalize both (1) and (2), i.e.,
   $|I \Delta G|$

Full Adder

```
[common signature]
predicate:ab/1;

[belief base]
name:juror1;
dlvargs: "-FRmin fulladder.dl abnormal1.hyp fault.obs";

[belief base] name:juror2; ...
[belief base] name:juror3; ...

[merging plan]
{
operator: dalal; aggregate:"sum";
penalize: "ignoring";
constraints: "fulladder.dl"; constraints: "fault.obs";
{juror1}; {juror2}; {juror3};
}
```

Full Adder - Group Decision

Individual explanations

1. $AS(P_{J_1}) = \{\{ab(haXor1)\}, \{ab(haXor2)\}\}$
2. $AS(P_{J_2}) = \{\{ab(haXor2)\}\}$
3. $AS(P_{J_3}) = \{\{ab(haXor1)\}\}$

Possible group explanations

1. $E_1 = \{ab(haXor1), ab(haXor2)\}$
2. $E_2 = \{ab(haXor1)\}$
3. $E_3 = \{ab(haXor2)\}$

Full Adder - Group Decision

Individual explanations

1. $AS(P_{J_1}) = \{\{ab(haXor1)\}, \{ab(haXor2)\}\}$
2. $AS(P_{J_2}) = \{\{ab(haXor2)\}\}$
3. $AS(P_{J_3}) = \{\{ab(haXor1)\}\}$

Possible group explanations

1. $E_1 = \{ab(haXor1), ab(haXor2)\}$
2. $E_2 = \{ab(haXor1)\}$
3. $E_3 = \{ab(haXor2)\}$

Distances to Individuals

Penalizing ignoring only

	$AS(P_{J_1})$	$AS(P_{J_2})$	$AS(P_{J_3})$	Sum
$\mathbf{E_1}$	0	0	0	$\mathbf{0} \Leftarrow$
E_2	0	1	0	1
E_3	0	0	1	1

Full Adder - Group Decision

Individual explanations

1. $AS(P_{J_1}) = \{\{ab(haXor1)\}, \{ab(haXor2)\}\}$
2. $AS(P_{J_2}) = \{\{ab(haXor2)\}\}$
3. $AS(P_{J_3}) = \{\{ab(haXor1)\}\}$

Possible group explanations

1. $E_1 = \{ab(haXor1), ab(haXor2)\}$
2. $E_2 = \{ab(haXor1)\}$
3. $E_3 = \{ab(haXor2)\}$

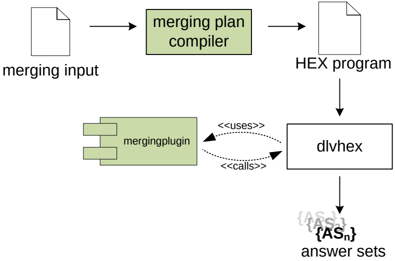
Distances to Individuals

Penalizing ignoring and unfounded group beliefs ($|I \Delta G|$)

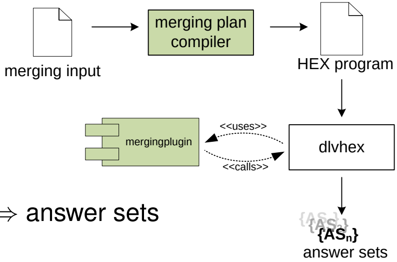
	$AS(P_{J_1})$	$AS(P_{J_2})$	$AS(P_{J_3})$	Sum
E_1	1	1	1	3
$\mathbf{E_2}$	0	2	0	**2** $\Leftarrow$
$\mathbf{E_3}$	0	0	2	**2** $\Leftarrow$

Summary

- Task: Merging of several belief bases

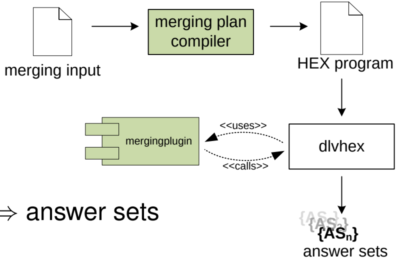


Summary



- Task: Merging of several belief bases
- Approach: Merging language
  - with user-defined merging operators
- merging input ⇒ compiler ⇒ dlvhex ⇒ answer sets

Summary



- Task: Merging of several belief bases
- Approach: Merging language
  - with user-defined merging operators
- merging input ⇒ compiler ⇒ dlvhex ⇒ answer sets

Advantages

- Develop merging operators only once or select one of the preinstalled ones (like Dalal)
- No need for manual re-merging after each change of the setting
- Try out several operators and evaluate which behaves best
- No routine tasks (like information flow between sources)
- User can focus on development and optimization of merging procedures in narrower sense!