1. Goal

Using multiple belief sources simultaneously!

Challenges
- Incompatible knowledge representation formalisms
- Different data schemas
- Logical contradictions
- Undesired artefacts prior to data cleansing
  (differing naming conventions, several entries referring to the same
  real-world object, etc.)

Applications
 judgment aggregation, classifier merging, ontology merging,
 fusion of business databases, ...

5. Merging Language

A declarative merging task description consists of

- **Common Signature**
  A set of predicates that is expressive enough to represent the contents
  of all data sources, i.e., a shared vocabulary

- **Mappings**
  Translate sources into the language of the common signature

- **Merging Operators**
  Operators are algorithms which incorporate \( n \) belief sets into one

- **Merging Plan**
  Arranges **merging operators** hierarchically in a tree-like structure
  with the input belief sets in the leaves

2. Application Scenario

Fault Diagnosis: Malfunctioning Full Adder

Suppose we have three experts, each with different hypothesis about the fault:
1. \( \text{xor1} \) is defect!
2. \( \text{xor2} \) is defect!
3. \( \text{xor1} \) is defect, but \( \text{xor2} \) works!

We want to make a group decision such that
- it explains the observation
- it is as similar to the individual decisions as possible

Expected result depends on cost model; one intuitively reasonable possibility: \( \text{xor1} \) is defect since it satisfies two of the three experts.

3. State-of-the-art

Many application-specific solutions which contain a lot of routine tasks
- Calling different sources of computation and parsing results
- Organizing the information flow through different merging procedures
- Extracting the final result

Drawbacks
- Changes in the merging strategy or parameters require lots of
time-consuming repetitive work
- Hence, empirical test of several scenarios in order to find the optimal
  one is difficult

4. New approach

A framework and language for formal description of merging tasks

The implementation is based on an extension of the answer set semantics,
called **HEX semantics**. Then the merging task consists of 3 steps:

**Step 1**: User specifies the merging scenario declaratively; this description
is called **merging task description**. For this purpose, the framework
provides a user-friendly **merging language**

**Step 2**: The merging task description is translated into a semantically
equivalent HEX program by the **merging plan-compiler**

**Step 3**: The generated program is executed by the reasoner **dlvhex** to pro-
duce the new belief set according to the defined merging plan

6. Framework Architecture: Implementation

**Technical Facts**

- **Formalism**: Logic programs under the **HEX** semantics; their
  answer-sets are regarded as derived knowledge, called **belief sets**
- **Reasoner**: dlvhex and DLV
- **Data Sources**: arbitrary, as long as suitable plugins for dlvhex exist
- **Operator Implementations**: C++ classes

7. Solving the Fault Diagnosis Problem

```
% File: diagnosis.mp
[common signature]
[predicate: ab/1]
[belief base]
{name: expert1; source: "e1.dl"; % yields \{ab(cor1)\}}
[belief base]
{name: expert2; source: "e2.dl"; % yields \{ab(cor2)\}}
[belief base]
{name: expert3; source: "e3.dl"; % yields \{ab(cor1), !ab(cor2)\}}
[merging plan]
{operator: dalal; aggregate: "sum";
 constraintfile: "fulladder.dl";
 constraintfile: "fault.obs";
 {expert1}; {expert2}; {expert3};
}

%Command line: $mpcompiler diagnosis.mp | dlvhex --filter=ab --
%Result: \{ab(xor1)\}
```

Advantage of the tool: Easy to exchange operator or modify parameters,
e.g., the cost model!

8. Conclusion

Very flexible due to generic design: useful for many application scenarios!

Advantages
- As of June 2010, most general framework available
- It is easy to change the merging strategy: no repetition of routine tasks
- Merging procedures implemented once and applied in many scenarios
- Implementation for **dlvhex** available; only very few comparable systems
  were actually implemented

Kontakt: christoph.redl@tuwien.ac.at