Project Discription:

Over the course of this project the MCS-IE-System was extended by another way of calculation, the calculation over explanations. The aim was to find the explanations directly by looking on the contexts (so far only calculated from found diagnosis), and then calculating the diagnosis from those found explanations.

The calculation over explanations was implemented and it works, but there are some flaws which are worth mentioning:

- ✗ Out of the explanations we only get the minimal diagnosis, not all diagnosis.
- ✗ Equilibria cannot be calculated
- ✗ The calcualtion over Explanations takes very long. (Med Example takes half a Day).

Refactoring & Codeextensions:

In order to extend the MCSI-System properly several classes have been refactored:

The Class InputConverter has been refactored, and therefore been split in 4 classes. The class InputConverter.cpp stays the Interface to DLV, but the functionality has been shifted to the following two classes: InputConverterDiagnosis.cpp, InputConverterExplanations.cpp. InputConverterDiagnosis contains just the same functionality the old Inputconverter used to have, while InputConverterExplanations contains the same basic structure but adapted to the Calculation over explanations. Methods used by both clases (generating or parsing the tree), have been put in a seperate class called InputConverterHelper.cpp.

The Class BridgeRule.cpp no longer contains the method named „write“, because both types of caluclation (over diagnosis & explanations) need different such methodes. The method write is now contained in the class InputconverterDiagnosis.cpp without changing the functionality. The class InputConverterExplanations.cpp contains a similair method adepted to the calculation over explanations.

The class Outputrewriter.cpp which is in charge of handling the generated output has been extended by code to handle the calculation over explanationoutput. Another method has been added: „getDiagnosis“, which, with the Help of an DLV Program, gathers the Diagnosis from the found minimal Explanations.

The Class ExplanationPrintVisitor.cpp has been created. This class is used for outprinting the found explanations from the calculation over explanation.

In order to control the two differen calculations another parameter has been integrated:
--iecompoverex
this parameter initializes the computation over explanations by setting the new variable computeoverExplanations in global.cpp.

How the Program works

Example call from Directory „examples“:

dlvhex –plugindir=../src --ieenable --ieexplain=E --iecompoverex benchmark1/dlvctx_master.hex

The parameter --iecompoverex initiates the calculation over Explanations. When this parameter is set, the class InputConverterExplanations.cpp will generate a DLVHex-Program that then calculates the Explanations from the given Contexts with the algorthmen from Antonius Winzierl, presented in the file „naive_explanation.pdf“ in the same folder.

When asking for minimal Diagnosis, the Diagnosis are caluclated from the found minimal Explanations by this DLV-Program:

```
//for every Bridgerule:
rule(bi).

//for each explanation Expi and all contained bridgerules in E1
inExpi_E1(bj).

//for each explanation Expi and all contained bridgerules in E2
inExpi_E2(bj).

//a bridgerule can be in d1 or not
d1(R) v nd1(R) :- rule(R).

//a bridgerule can be in d2 or not
d2(R) v nd2(R) :- rule(R).

//a bridgerule cannot be in d1 and d2 the same time
:- d1(R), d2(R).

//d1OK follows if the intersection of d1 with all E1 is empty
d1OK :- inExpi_E1(Ri), d1(Ri).

//d2OK follows if the intersection of d2 with all E2 is empty
d2OK :- inExpi_E2(Ri), d2(Ri).

//if neither d1OK nor d2OK is calculated, it is no DIagnosis
:- not d1OK, not d2OK.
```

Remark: The found Diagnosis are not correct, but if we filter the minimal Diagnosis from them we get the right minimal diagnosis. So we can calculate Dm but we cannot calculate the correct D.

Finally the class Outputrewriter.cpp prints out the generted explanations/diagnosis.

Explanatory Notes:

In the calculation of the explanations following rules were used in the constructed DLV-Program.

//for every bridgerule
r1(ri) :- e1(bi).
r1(bi) v nr1(bi) :- ne1(bi).
r2(bi) v nr2(bi) :- ne2(bi).

Rather than:

r1(R) :- e1(R).
r1(R) v nr1(R) :- ne1(R).
r2(R) v nr2(R) :- ne2(R).

This is due to the fact, that the calculation is twelve times as fast with the grounded rules than with the ungrounded. The cause to this phenomen is unknown. Probably a minor bug in DLVHex.