Interactive debugging of knowledge-bases

Keynote @ DX 2014

Gerhard Friedrich

Thanks to my colleagues:

Kostyantyn Shchekotykhin
Patrick Rodler
Phillip Fleiss

Intelligent systems and business informatics
University Klagenfurt
Thanks to my collaborators for the last years:

- Liliana Ardissono
- Luca Console
- Daniele Theseider Dupré
- Andreas Falkner
- Alexander Felfernig
- Gerhard Fleischanderl
- Philipp Fleiss
- Hartmut Freitag
- Melanie Frühstück
- Georg Gottlob
- Alois Haselböck
- Klaus Isak
- Dietmar Jannach
- Jürgen Kuster
- Franz Lackinger
- Monika Mandl
- Wolfgang Mayer
- Vera Mersheeva
- Wolfgang Nejdl
- Giovanna Petrone
- Axel Polleres
- Johannes Retti
- Patrick Rodler
- Christian Russ
- Anna Ryabokon
- Gottfried Schenner
- Herwig Schreiner
- Monika Schubert
- Ralph Schäfer
- Kostyantyn M. Shchekotykhin
- Marius-Calin Silaghi
- Markus Stumptner
- Erich Teppan
- Franz Wotawa
- Markus Zanker

... and all other 76 co-authors
“Since we can never know anything for sure, it is simply not worth searching for certainty; but it is well worth searching for truth; and we do this chiefly by searching for mistakes, so that we have to correct them.”

– Sir Karl Popper, Austrian-British philosopher of science

This talk is about:

Given a “wrong” knowledge-base (theory):
Support the formulation of a valid knowledge-base
Outline

- Motivation
- Basic concepts
- Interactive debugging of knowledge-bases & its application
- Current research topics & results
- Conclusions and future work
Why diagnosis of descriptions?

1986, Siemens:

Broadcasting studio equipment
- Diagnosis system
- Configure-to-order
- Configuration system
- Classical Expert-System approach fails

1987: Highly inspiring work on first-principles/model-based diagnosis by Johan de Kleer, Brian Williams, Raymond Reiter, Randall Davis ('83)

Copyright by Siemens
1991, a conversation at Siemens, Engineering Centre for Electronics:

There is a great, general method for diagnosing failures of machines.

That is interesting. However, our biggest problems are design and software failures.
Let us diagnose descriptions/knowledge-bases

- Prolog
- VHDL
- Configuration KBs
- Constraints
- Recomender Systems
- Description logic KBs
- Software (Java)
- LTL-descriptions
- Workflow instances
- Spreadsheets
- ASP KBs
Test driven development

Tests uncover failures

Tests pass

Refactoring
System overview

**Input**: Background Theory $B$, Positive and Negative Test Cases

1: Diagnoses

Partition Generation

2: Partitions

INTERACTIVE DEBUGGER

3: Query

4: Answer

5: Selected Partition/Answer

Fault Information

6: Test Cases

Test Cases

Knowledge Base

Output: $D^*$
Valid knowledge-base

Given

- A set of positive test cases $P$
- A set of negative test cases $N$

A knowledge-base $KB^*$ is valid iff

$KB^*$ is consistent

$KB^* \models p \quad \forall p \in P$

$KB^* \not\models n \quad \forall n \in N$
Invalid knowledge-bases: repair

- Remove faulty formulas $\mathcal{D}$
- Add correct formulas $\mathcal{EX}$

$$KB' := (KB \setminus \mathcal{D}) \cup \mathcal{B} \cup \mathcal{EX}$$

$\mathcal{B}$: background theory
Model-based diagnosis

\[ ax_1 : I \rightarrow A \quad ax_2 : A \rightarrow B \]
\[ ax_3 : B \rightarrow C \quad ax_4 : C \rightarrow O \]

\[ \mathcal{B} = \{ I, \neg O \} \]

\[ KB \cup \mathcal{B} \]

\[ KB \] is a conflict

- Let \( \mathcal{D} = [ax_1] \), then \( (KB \setminus \mathcal{D}) \cup \mathcal{B} \)
Model-based diagnosis

\[ ax_1 : I \to A \quad ax_2 : A \to B \]
\[ ax_3 : B \to C \quad ax_4 : C \to O \]

\[ \mathcal{B} = \{ I, \neg O \} \]
Diagnosis problem

A set of axioms $\mathcal{D} \subseteq KB$ is a candidate diagnosis iff

\[
(\mathcal{KB} \setminus \mathcal{D}) \cup B \cup EX \quad \text{is consistent}
\]

\[
(\mathcal{KB} \setminus \mathcal{D}) \cup B \cup EX \models p \quad \forall p \in P
\]

\[
(\mathcal{KB} \setminus \mathcal{D}) \cup B \cup EX \not\models n \quad \forall n \in N
\]

$\mathcal{D} \subseteq \mathcal{KB}$ is a diagnosis iff there is no such $\mathcal{D}' \subset \mathcal{D}$ that $\mathcal{D}'$ is a candidate diagnosis

Diagnosis problem instance (DPI):

Given $\langle \mathcal{KB}, B, P, N \rangle$ find a diagnosis
Approximation of the extension

- Since $KB^* \models p \quad \forall p \in P$
- Extension is approximated by positive test cases
A set of formulas \( D \subseteq KB \) is a (candidate) diagnosis iff the knowledge-base

\[
KB' := (KB \setminus D) \cup B \cup P
\]

- is consistent
- \( KB' \not\models n \quad \forall n \in N \)
System overview

Input: Background Theory $B$, Positive and Negative Test Cases

Partition Generation

Interactive Debugger

1: Diagnoses
2: Partitions
5: Selected Partition/Answer
4: Answer

Diagnosis Engine

Knowledge Base

Fault Information

6: Test Cases
3: Query

Output: $D^*$
## Test ontologies

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Axioms</th>
<th>Diagnoses</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical</td>
<td>114</td>
<td>6</td>
<td>Elements</td>
</tr>
<tr>
<td>Koala</td>
<td>44</td>
<td>10</td>
<td>Training</td>
</tr>
<tr>
<td>Sweet-JPL</td>
<td>2579</td>
<td>13</td>
<td>Earth-science</td>
</tr>
<tr>
<td>University</td>
<td>50</td>
<td>90</td>
<td>Training</td>
</tr>
<tr>
<td>Tambis</td>
<td>596</td>
<td>147</td>
<td>BioMed</td>
</tr>
<tr>
<td>Economy</td>
<td>1781</td>
<td>864</td>
<td>Mid-level</td>
</tr>
<tr>
<td>Transportation</td>
<td>1300</td>
<td>1782</td>
<td>Mid-level</td>
</tr>
</tbody>
</table>
What to do?

Specify more test cases!

But, which ones?
Generation of test cases

Observation: KBs for different diagnoses

\[ KB_i := (KB \setminus D_i) \cup B \cup P \]

have different sets of entailments

Entailments can be used as test cases!
Sample entailments (1)

\[
\begin{align*}
ax_1 : I & \rightarrow A \\
ax_2 : A & \rightarrow B \\
ax_3 : B & \rightarrow C \\
ax_4 : C & \rightarrow O
\end{align*}
\]

\[
\mathcal{B} = \{ I, \neg O \}
\]

- Four possible diagnoses

\[
\begin{align*}
\mathcal{D}_1 : [ax_1] \\
\mathcal{D}_2 : [ax_2] \\
\mathcal{D}_3 : [ax_3] \\
\mathcal{D}_4 : [ax_4]
\end{align*}
\]
Sample entailments (2)

- Four KBs corresponding to each diagnosis

\[ KB_i := (KB \setminus D_i) \cup B \cup P \]

- Sets of unit clauses entailed by each \( KB_i \)

\[
\begin{align*}
KB_1 & : \{ \neg A, \neg B, \neg C \} \\
KB_2 & : \{ A, \neg B, \neg C \} \\
KB_3 & : \{ A, B, \neg C \} \\
KB_4 & : \{ A, B, C \}
\end{align*}
\]
Sample entailments (3)

\[ KB_1 : \{ \neg A, \neg B, \neg C \} \]
\[ KB_3 : \{ A, B, \neg C \} \]
\[ KB_2 : \{ A, \neg B, \neg C \} \]
\[ KB_4 : \{ A, B, C \} \]

Positive literals

\[ A : \{ KB_2, KB_3, KB_4 \} \]
\[ B : \{ KB_3, KB_4 \} \]
\[ C : \{ KB_4 \} \]

Negative literals

\[ \neg A : \{ KB_1 \} \]
\[ \neg B : \{ KB_1, KB_2 \} \]
\[ \neg C : \{ KB_1, KB_2, KB_3 \} \]
Query (1)

Positive literals

\[ A : \{KB_2, KB_3, KB_4\} \]
\[ B : \{KB_3, KB_4\} \]
\[ C : \{KB_4\} \]

Negative literals

\[ \neg A : \{KB_1\} \]
\[ \neg B : \{KB_1, KB_2\} \]
\[ \neg C : \{KB_1, KB_2, KB_3\} \]

- If \( KB^* \not\models B \) then \( \{KB_3, KB_4\} \) are not valid

Consequently \( \{D_3, D_4\} \) are not diagnoses
Query (2)

<table>
<thead>
<tr>
<th>Positive literals</th>
<th>Negative literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A : {KB_2, KB_3, KB_4} )</td>
<td>( \neg A : {KB_1} )</td>
</tr>
<tr>
<td>( B : {KB_3, KB_4} )</td>
<td>( \neg B : {KB_1, KB_2} )</td>
</tr>
<tr>
<td>( C : {KB_4} )</td>
<td>( \neg C : {KB_1, KB_2, KB_3} )</td>
</tr>
</tbody>
</table>

- If \( KB^* \models B \)
  - then \( \{KB_1, KB_2\} \) are not valid

Consequently \( \{D_1, D_2\} \) are not diagnoses
Query (3)

<table>
<thead>
<tr>
<th>Positive literals</th>
<th>Negative literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A : {KB_2, KB_3, KB_4}$</td>
<td>$\neg A : {KB_1}$</td>
</tr>
<tr>
<td>$B : {KB_3, KB_4}$</td>
<td>$\neg B : {KB_1, KB_2}$</td>
</tr>
<tr>
<td>$C : {KB_4}$</td>
<td>$\neg C : {KB_1, KB_2, KB_3}$</td>
</tr>
</tbody>
</table>

- What is true?

\[ KB^* \models B \quad \text{or} \quad KB^* \not\models B \]

- Ask an oracle, if \( KB^* \models B \)?
Query generation (1)

\[
\mathcal{D}^N = \{ \mathcal{D}_1 \} \\
\mathcal{D}^P = \{ \mathcal{D}_3, \mathcal{D}_4 \}
\]

\[
KB_3 \text{ and } KB_4 \models B \\
KB_3 \text{ and } KB_4 \models A
\]

\[
\mathcal{D}^P \cup \{ \mathcal{D}_1 \} \quad \text{if } KB_1 \models B \\
\mathcal{D}^N \cup \{ \mathcal{D}_1 \} \quad \text{if } KB_1 \models \neg B \\
\mathcal{D}^\emptyset \cup \{ \mathcal{D}_1 \} \quad \text{otherwise}
\]
Query generation (2)

\[ \{ \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4 \} \]

\[ \mathbf{D}^N = \{ \mathcal{D}_1, \mathcal{D}_2 \} \]

\[ \mathbf{D}^P = \{ \mathcal{D}_3, \mathcal{D}_4 \} \]

\[
\begin{align*}
KB_3 \text{ and } KB_4 \models B \\
KB_3 \text{ and } KB_4 \models A
\end{align*}
\]

<table>
<thead>
<tr>
<th>Query</th>
<th>( \mathbf{D}^P )</th>
<th>( \mathbf{D}^N )</th>
<th>( \mathbf{D}^\emptyset )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_1 : { A } )</td>
<td>( { \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4 } )</td>
<td>( { \mathcal{D}_1 } )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( Q_2 : { B } )</td>
<td>( { \mathcal{D}_3, \mathcal{D}_4 } )</td>
<td>( { \mathcal{D}_1, \mathcal{D}_2 } )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( Q_3 : { C } )</td>
<td>( { \mathcal{D}_4 } )</td>
<td>( { \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3 } )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
Query generation (3)

- Worst case: $O\left(2^{|D|}\right)$ partitions to verify

- Query generation is feasible if the number of diagnoses is fixed

  $|D| \leq n$

- Heuristic algorithms can speed-up computation of a partition with required properties

System overview

Input: Background Theory $B$, Positive and Negative Test Cases

INTERACTIVE DEBUGGER

1: Diagnoses → 2: Partitions → 3: Query → 4: Answer → 5: Selected Partition/Answer → 6: Test Cases

Knowledge Base

Fault Information

Output: $D^*$
Query selection strategies:

- Random: random query
- Split-in-half: query that removes a half of diagnoses regardless of the answer
- Entropy-based: query that minimizes the expected posterior entropy of the set of diagnoses
Entropy strategy

- Specify beliefs in terms of probability

- Expected entropy after querying $Q$

$$H_e(Q) = \sum_{v \in \{yes, no\}} p(Q = v) \times$$

$$- \sum_{D_j \in D} p(D_j|Q = v) \log_2 p(D_j|Q = v)$$
Computing diagnoses

Prerequisite:

- Reasoning system that correctly outputs consistent (rsp. inconsistent) if the knowledge-base is consistent (rsp. inconsistent)

Architecture:

Extend Reiter’s hitting set algorithm for computing hitting set trees

Apply Junker’s QUICKXPLAIN for computing **minimal** conflict sets:

Costs in the **worst case**: $2k \log_2(n/k) + 2k$ consistency checks, where

- $k$ is the length of the minimal conflict
- $n$ number of axioms
Evaluation
## Incoherent ontologies from TONES

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Domain</th>
<th>Axioms</th>
<th>#CS/min/max</th>
<th>#D/min/max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical</td>
<td>Chem. Elements</td>
<td>144</td>
<td>6/5/6</td>
<td>6/1/3</td>
</tr>
<tr>
<td>Koala</td>
<td>Training</td>
<td>44</td>
<td>3/4/4</td>
<td>10/1/3</td>
</tr>
<tr>
<td>Sweet-JPL</td>
<td>Earth Science</td>
<td>2579</td>
<td>1/13/13</td>
<td>13/1/1</td>
</tr>
<tr>
<td>miniTambis</td>
<td>Bio. Science</td>
<td>173</td>
<td>3/2/6</td>
<td>48/3/3</td>
</tr>
<tr>
<td>University</td>
<td>Training</td>
<td>49</td>
<td>4/3/5</td>
<td>90/3/4</td>
</tr>
<tr>
<td>Economy</td>
<td>Mid-level</td>
<td>1781</td>
<td>8/3/4</td>
<td>864/4/8</td>
</tr>
<tr>
<td>Transportation</td>
<td>Mid-level</td>
<td>1300</td>
<td>9/2/6</td>
<td>1782/6/9</td>
</tr>
</tbody>
</table>

Consistency checking is not in NP
Evaluation results: #queries (1)

- Prior fault probabilities of a **good** quality
- Query length: 2-4 facts

![Bar chart showing the average number of queries for different categories: Chemical, Koala, Sweet-JPL, miniTambis, University, Economy, Transportation. The x-axis represents the categories, and the y-axis represents the average number of queries. The chart compares Entropy and Split-in-half methods.](image-url)
Evaluation results: #queries (2)

- Prior fault probabilities of an average quality
Evaluation results: #queries (3)

- Prior fault probabilities of a **bad** quality
Evaluation results: time (4)

- Computation time for 9 diagnoses and a query

![Bar chart showing computation times for various diagnoses and queries. The x-axis represents different diagnoses: Chemical, Koala, Sweet-JPL, miniTabmis, University, Economy, Transportation. The y-axis represents time in seconds, ranging from 0.01 to 10. Each diagnosis has two bars: one for diagnosis and one for query. The bars show the time taken for each operation.]
Scalability test

- Statistics for large ontologies for a random alteration:

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Cton</th>
<th>Opengalen-no-propchains</th>
</tr>
</thead>
<tbody>
<tr>
<td># Axioms</td>
<td>33,203</td>
<td>9,664</td>
</tr>
<tr>
<td># CS/min/max</td>
<td>6/3/7</td>
<td>9/5/8</td>
</tr>
<tr>
<td># D/min/max</td>
<td>15/1/5</td>
<td>110/2/6</td>
</tr>
<tr>
<td>Consistency (sec)</td>
<td>0.005/2/1</td>
<td>0.001/0.1/0.4</td>
</tr>
<tr>
<td>min Conflict (sec)</td>
<td>17/20/38</td>
<td>7/10/12</td>
</tr>
<tr>
<td>min Diagnosis (sec)</td>
<td>0.1/5/38</td>
<td>0.01/1/19</td>
</tr>
<tr>
<td>Overall runtime (sec)</td>
<td>146</td>
<td>119</td>
</tr>
</tbody>
</table>

- #CS/min/max: # min conflict sets, min and max cardinality
- #DS/min/max: # min diagnoses, min and max cardinality
- Runtime in seconds: min/avg/max
Scalability test: time (cont.)

- Using entropy
- Average time in sec for 15 cases
- Number of most probable diagnoses 9
Scalability test: #queries (cont.)

- Average number of queries and their length for 15 cases
- Number of most probable diagnoses 9
Current research / extensions:
How to deal with uncertain probabilities?
Query Selection Strategies (QSS)

Which query should be selected next in order to minimize the overall number of queries?

Two existing paradigms:

- **Split-in-half (SPL):**
  Select query Q such that half of diagnoses in D are definitely eliminated; does **not** consider fault probabilities F
  No risk, but no potential

- **Entropy-based (ENT):**
  Select query Q with highest information gain based on fault probabilities F
  High potential, but high risk
Goal:
- High potential and minimal risk,
- given fault probabilities $F$ of any (good or poor) quality

Reinforcement learning method exploiting benefits of both SPL and ENT:
- Dynamic changes of strategy during debugging session
- Take advantage of $F$ as long as good performance is achieved
  - Behaviour like ENT
- Gradually get more independent from $F$ if suboptimal behaviour is measured
  - Behaviour towards SPL
Four experiments: Each test instance results from alignment of two ontologies produced by different automated matching systems

- **EXP-1, EXP-2: OAEI Conference Track**
  (27 debugging sessions, ~ 700 - 1200 axioms)

- **EXP-3, EXP-4: OAEI Anatomy Track**
  (7 debugging sessions, > 17,000 axioms)

- EXP-1, EXP-3: “Good” probabilities
- EXP-2, EXP-4: “Poor” probabilities
- Available reference mappings used to predefined target diagnosis D*
Evaluation - Results (1)

![Diagram showing results for different systems: HMatch, Falcon-AO, OWL-Ctxmatch, COMA++. The x-axis represents the systems, and the y-axis represents the metric q. EXP-1 and EXP-2 are compared with error bars indicating variability.](image-url)
Evaluation - Results (2)

[Graph showing results for different tools and experimental conditions (EXP-3 and EXP-4)]
Current research / extensions:
High cardinality faults
Direct diagnosis

- Computation of \( m \) “most probable” diagnoses is problematic for large with high cardinality faults KBs
- Modify HS-Tree to use diagnoses as node labels
- Computational complexity of finding a diagnosis

\[
O \left( |\mathcal{D}| \log \frac{|KB|}{|\mathcal{D}|} \right)
\]

calls to a consistency checker
Evaluation results

- Anatomy ontology, > 17,000 axioms, employing Entropy
- Average cardinality of the minimum card. diagnoses is 3
Evaluation results

- Conference ontology, employing Entropy
Evaluation results

- Conference ontology, employing Entropy
Conclusions

- Large number of possible repairs
- Interactive methods:
  - Locates preferred diagnosis by queries
  - Deals with uncertain probabilities
  - Employs direct diagnosis methods for high cardinality faults
- Many diagnosis problem instances are diagnosable (depends on costs of consistency checking)
Current and future work

- Reduction of computational costs
  - Modularization
  - Costs of consistency-checking depends on answers
- Distributed query answering
- Additional requirements: e.g. forbidden deductions
- Non monotonic semantics: ASP
- Repair proposals
Thank you for your attention!

Questions?

