Inconsistency management in source code with abductive logic programming

Sergio Castro
RELEASed lab
Université catholique de Louvain
sergio.castro@uclouvain.be
Advisor: Kim Mens
An example
An inconsistency in the command design pattern

AbstractAction
- (abstract) perform
- isUndoable ^false.

DrawAction
- perform
- isUndoable ^true.

QueryAction
- perform
- undoAction
- isUndoable ^aBlock value.

Consistency property:
An undoAction method should be provided if and only if the isUndoable method returns true.
An example
An inconsistency in the command design pattern

Consistency property:
An **undoAction** method should be provided if and only if the **isUndoable** method returns true.
An example
An inconsistency in the command design pattern

AbstractAction
- (abstract) perform
- isUndoable ^false.

Experimental Action
- perform
- undoAction

Inconsistency

Consistency property:
An undoAction method should be provided if and only if the isUndoable method returns true.

DrawAction
- perform
- isUndoable ^true.

QueryAction
- perform
- undoAction
- isUndoable ^aBlock value.

Inconsistency
An example
An inconsistency in the command design pattern

**Consistency property:**
An *undoAction* method should be provided if and only if the *isUndoable* method returns true.

AbstractAction
- *(abstract)* perform
- *isUndoable* ^false.

Experimental Action
- perform
- undoAction

DrawAction
- perform
- *isUndoable* ^true.

QueryAction
- perform
- undoAction
- *isUndoable* ^aBlock value.
Intensive

(consistency definition)
Intensive

(consistency definition)
Intensive

(consistency definition)
Intensive
(consistency definition)
IntensiVE
(consistency checking)

Classes that return true in the isUndoable method

Classes that implement the undoAction method
IntensiVE
(consistency checking)

<table>
<thead>
<tr>
<th>Tuples</th>
<th>1 (250 ms)</th>
<th>2 (238 ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>class -&gt; AbstractAddAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddClassification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddObject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddSmartClass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; ClearClassification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; ExperimentalAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; RemoveAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; RenameClassificationAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; TestAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddAlternativeAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddIVGroupAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddIVViewAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddRegularityAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddRelationAction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Class does not return true at the `isUndoable` method

But implements an `undoAction` method
IntensiVE
(diagnosing inconsistencies)

Why the query for finding a true return statement in the `isUndoableMethod` is failing?

OR: Why the query for finding an `undoAction` method is succeeding?
A SLD-tree for the failing query

?-classChainReallyUnderstandMethodName(ExperimentalAction, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)

:-superclassOf(?s, ExperimentalAction),
classChainReallyUnderstandMethodName(?s, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)

:-methodNameInClass(?m, undoAction, ExperimentalAction),
methodWithBooleanReturnStatement(?m, true)
**IntensiVE**

*(diagnosing inconsistencies)*

---

![IntensiVE Interface](image)

**View Consistency**

1. UndoableActionsForIsUndoable
2. UndoableActionsForUndoAction

<table>
<thead>
<tr>
<th>Table View</th>
<th>Text Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>class -&gt; AbstractAddAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddClassificationAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddObjectAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddSmartClassificationAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; ClearClassificationAction</td>
<td></td>
</tr>
<tr>
<td><strong>class -&gt; ExperimentalAction</strong></td>
<td></td>
</tr>
<tr>
<td>class -&gt; RemoveAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; RenameClassificationAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; TestAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddAlternativeAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddIVGroupAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddIVViewAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddRegularityAction</td>
<td></td>
</tr>
<tr>
<td>class -&gt; AddRelationAction</td>
<td></td>
</tr>
</tbody>
</table>

**Tuples**

1. 250 ms
2. 238 ms

- **Inconsistent (8/29)**
  - Full Extension

---

**Actions:**

- Browse
- Inspect
- Sort
- Document as Exception
- Remove exception
- **Explain**
- Correct
A proof tree with our tool
But ...
What does a real SLD tree look like?
What does a real SLD tree look like?
What does a real SLD tree look like?
What does a real SLD tree look like?

Where is the cause of the problem?

How to solve it?
The result of a query can be changed ...

Adding solutions

Transforming failure branches in success
The result of a query can be changed ...

Adding solutions

Transforming failure branches in success
The result of a query can be changed ...
The result of a query can be changed ...
Our solution

• Framework for defining small partial solutions that can be composed.

• The core of this is based on an abductive meta-interpreter.
What is Abduction?

One of the three forms of reasoning according to Pierce

- Deduction
- Induction
- Abduction
Abduction is suitable for:

• Choosing the hypotheses that would, if true, explain an evidence.

• Alternatively, choosing currently true hypotheses that would, if false, explain an evidence (extended abduction).

• These explanations are expressed in terms of some predicates, declared before hand as abducibles.
Defining positive explanations

(abducible predicates)
Defining positive explanations

*abducible predicates*
Defining positive explanations

\textit{(abducible predicates)}
Defining positive explanations

(*abducible* predicates)
Diagnosing & correcting inconsistencies
Diagnosing & correcting inconsistencies

Node marked with positive explanations
Future work

• Choosing a more complex case study (currently analyzing the Starbrowser).

• How to compose partial solutions.

• Filter solutions that will not cause new inconsistencies.

• How to choose among different solutions.
Many Thanks

- Questions?
- Feedback?