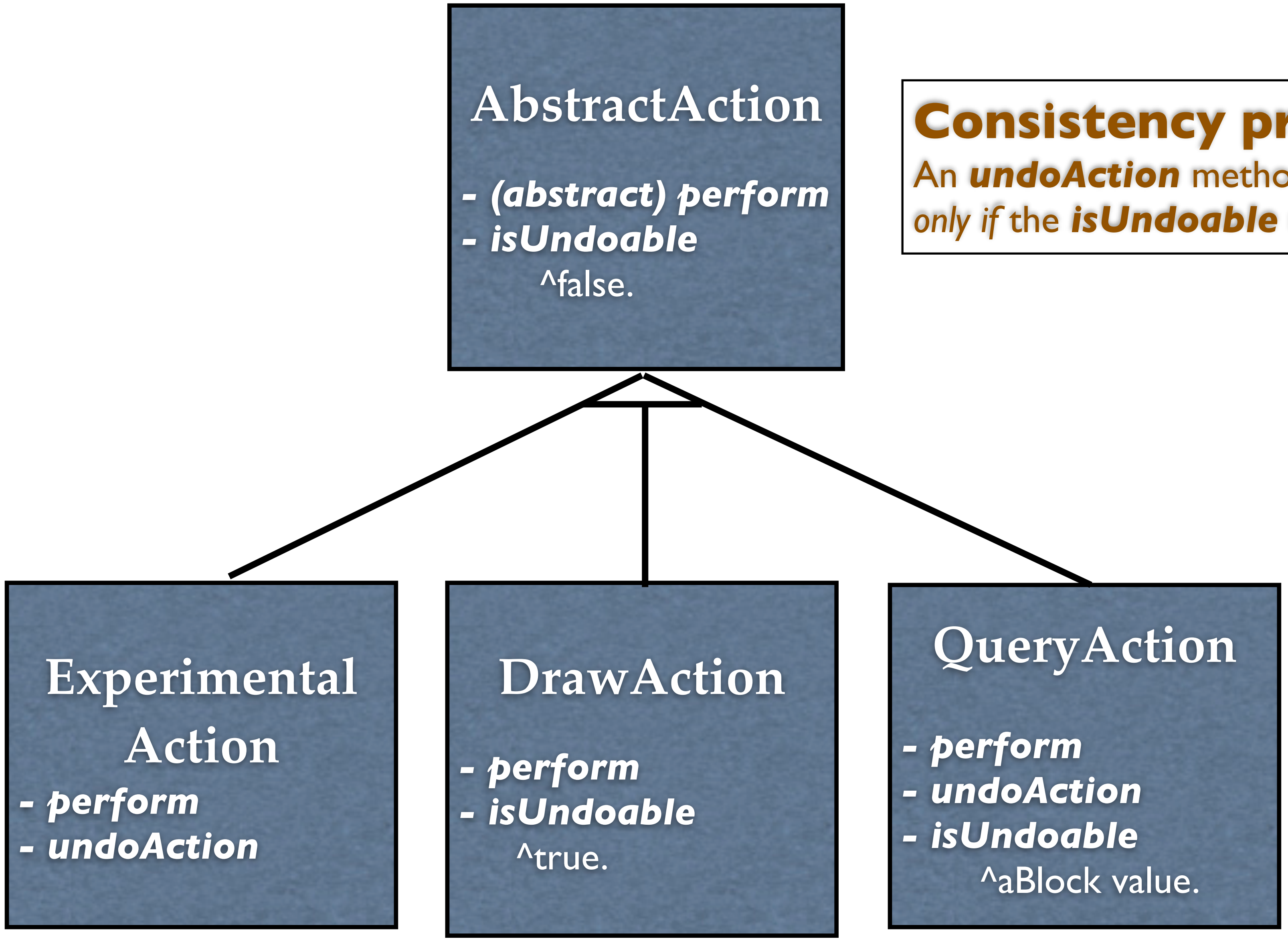
Inconsistency management in source code with abductive logic programming

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.

Experimental Action

- ***perform***
- ***undoAction***

DrawAction

- ***perform***
- ***isUndoable***

^true.

QueryAction

- ***perform***
- ***undoAction***
- ***isUndoable***

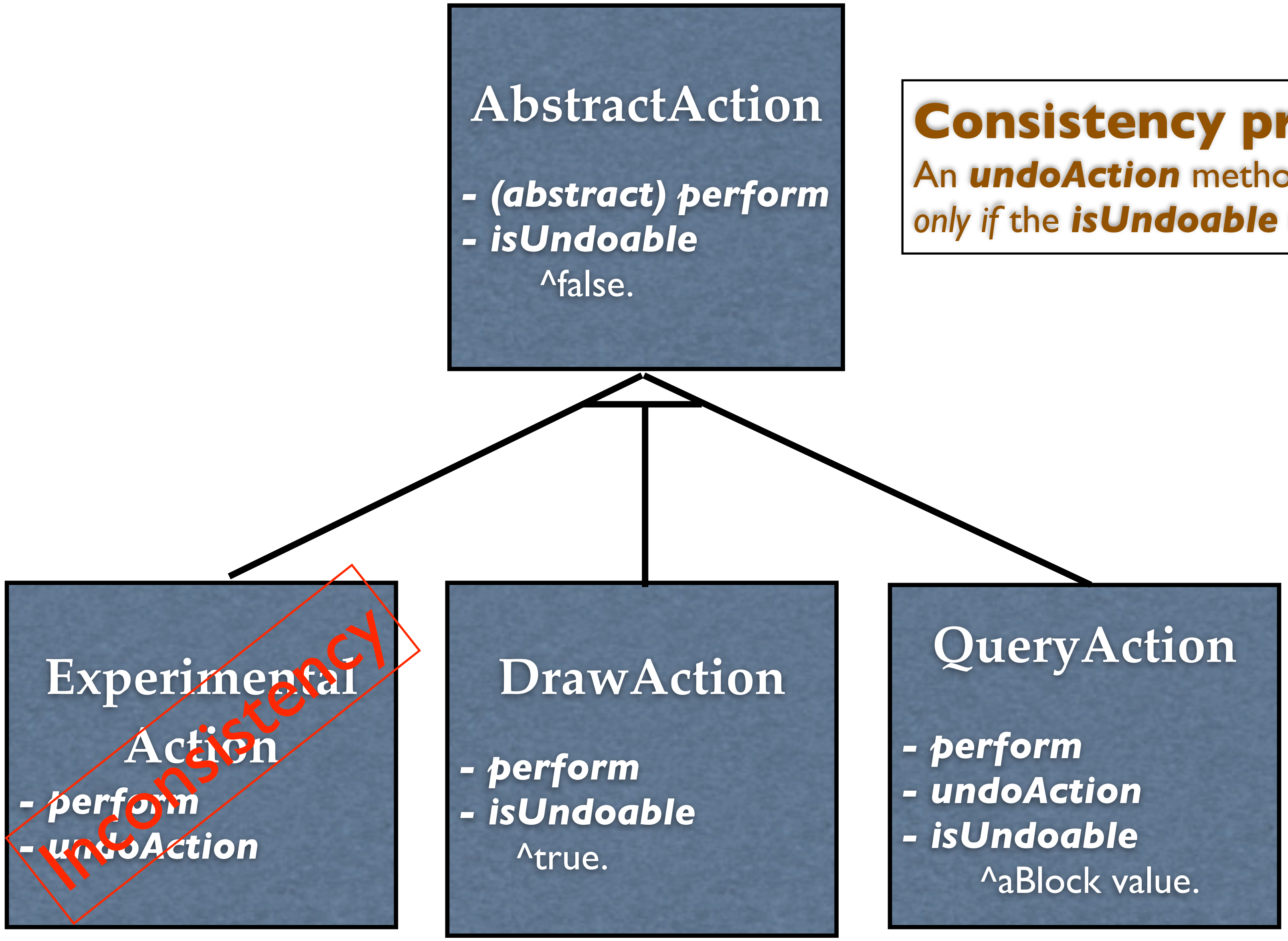
^aBlock value.

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



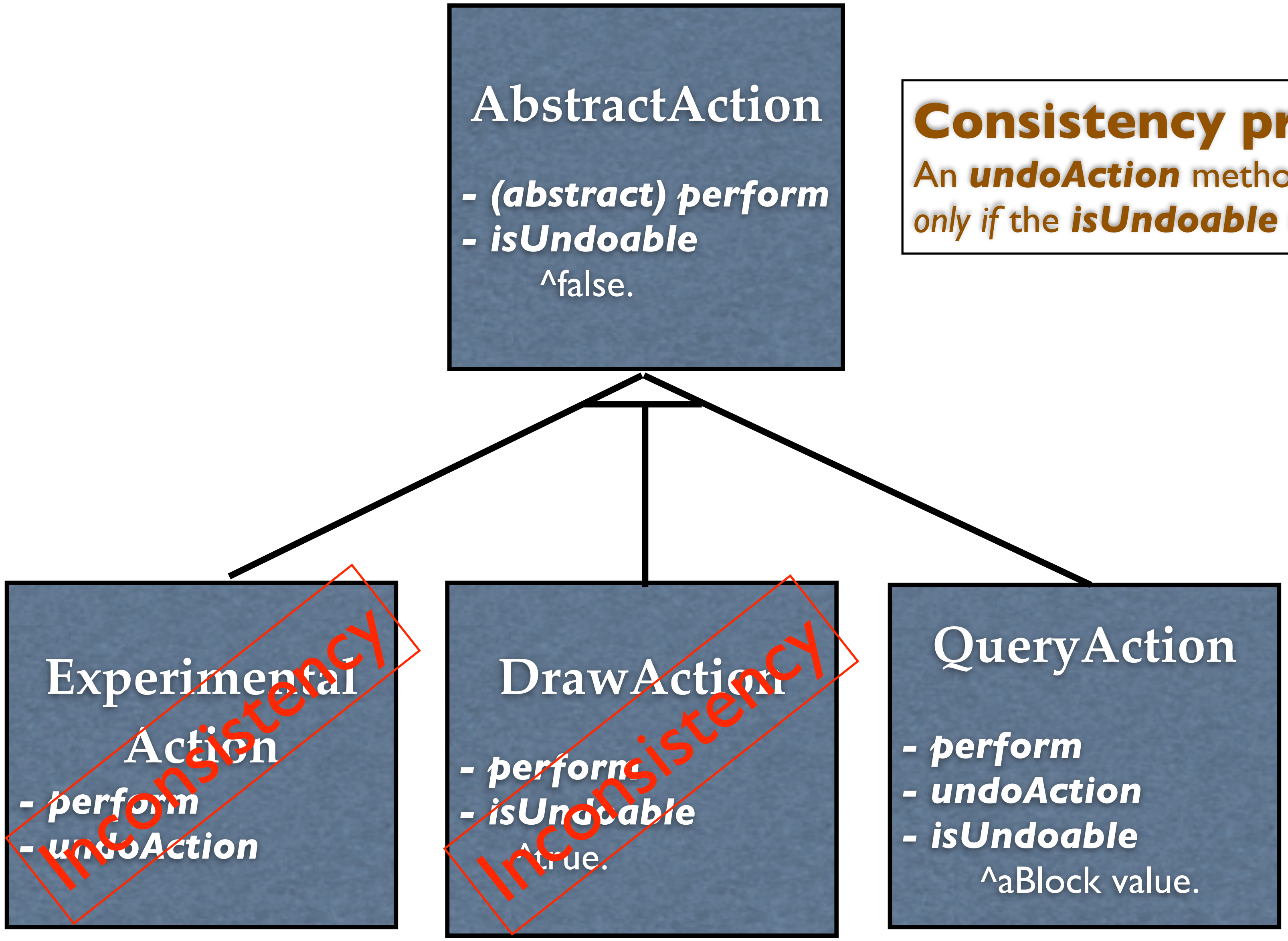
Consistency property:

An ***undoAction*** method should be provided *if and only if* the ***isUndoable*** method returns *true*

An example

An inconsistency in the command design pattern

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



AbstractAction

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

ExperimentalAction

- ***perform***
- ***undoAction***

Inconsistency

DrawAction

- ***perform***
- ***isUndoable***
  ^true.

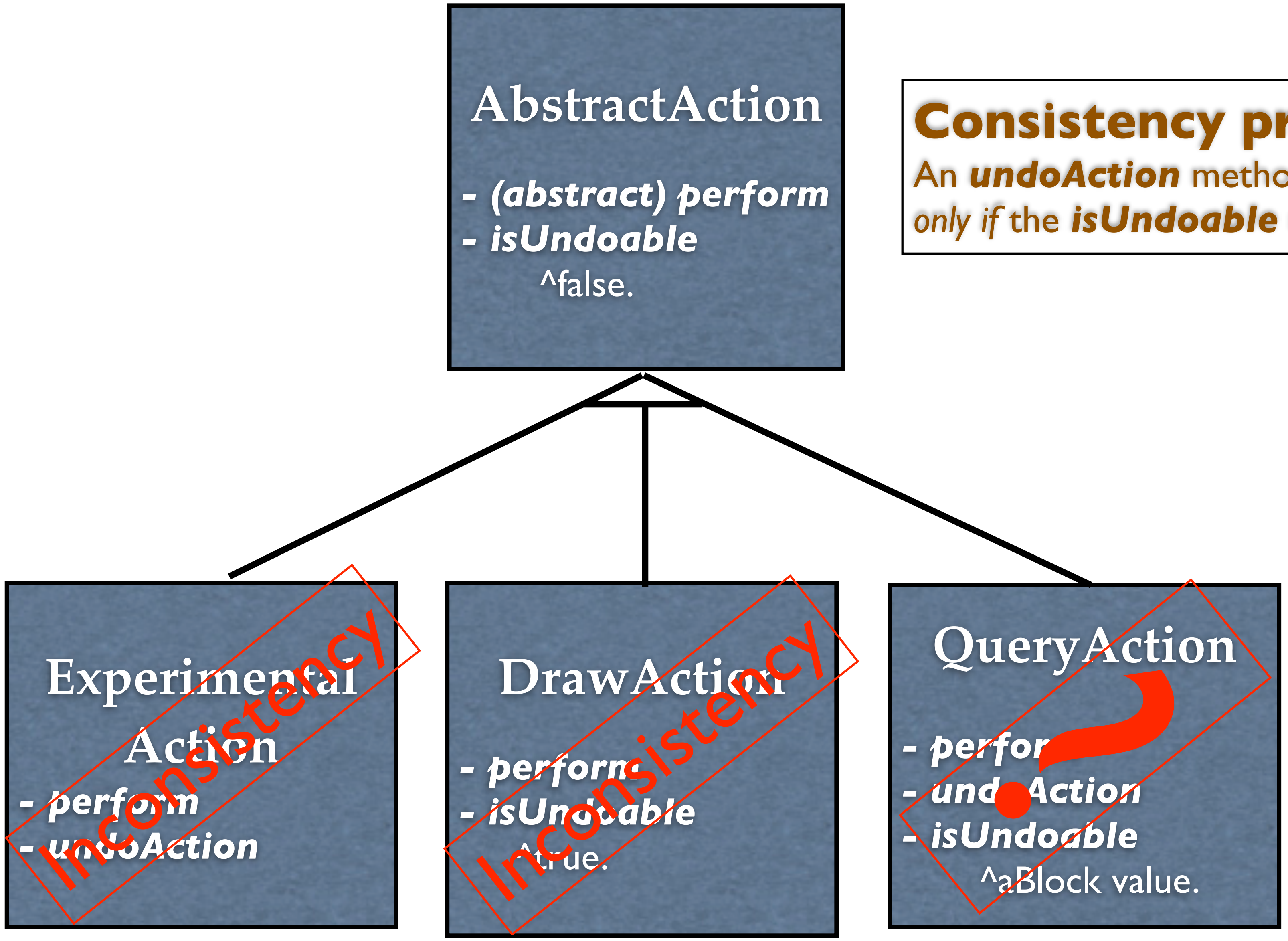
Inconsistency

QueryAction

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

An example

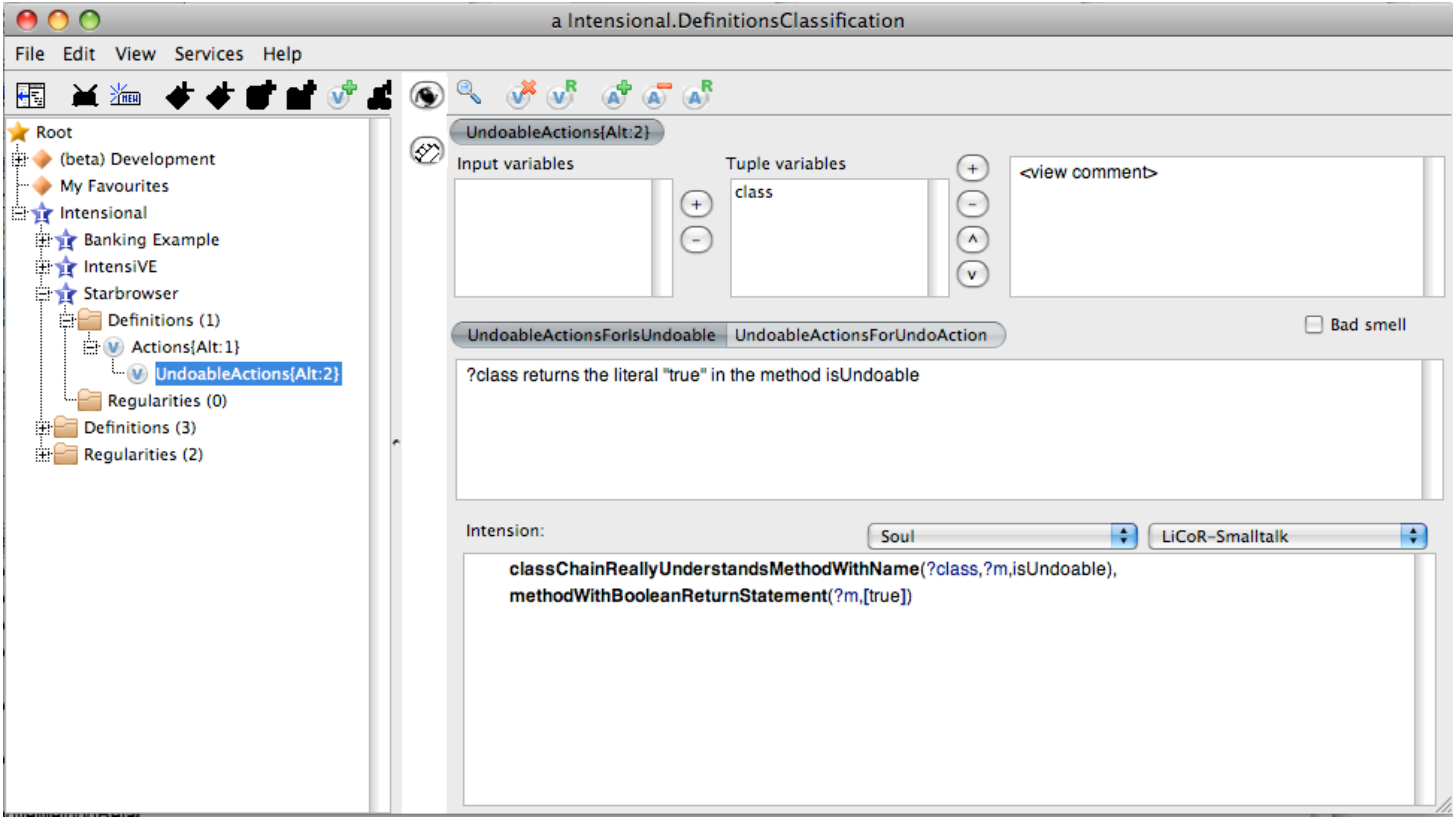
An inconsistency in the command design pattern



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

IntensiVE

(consistency definition)

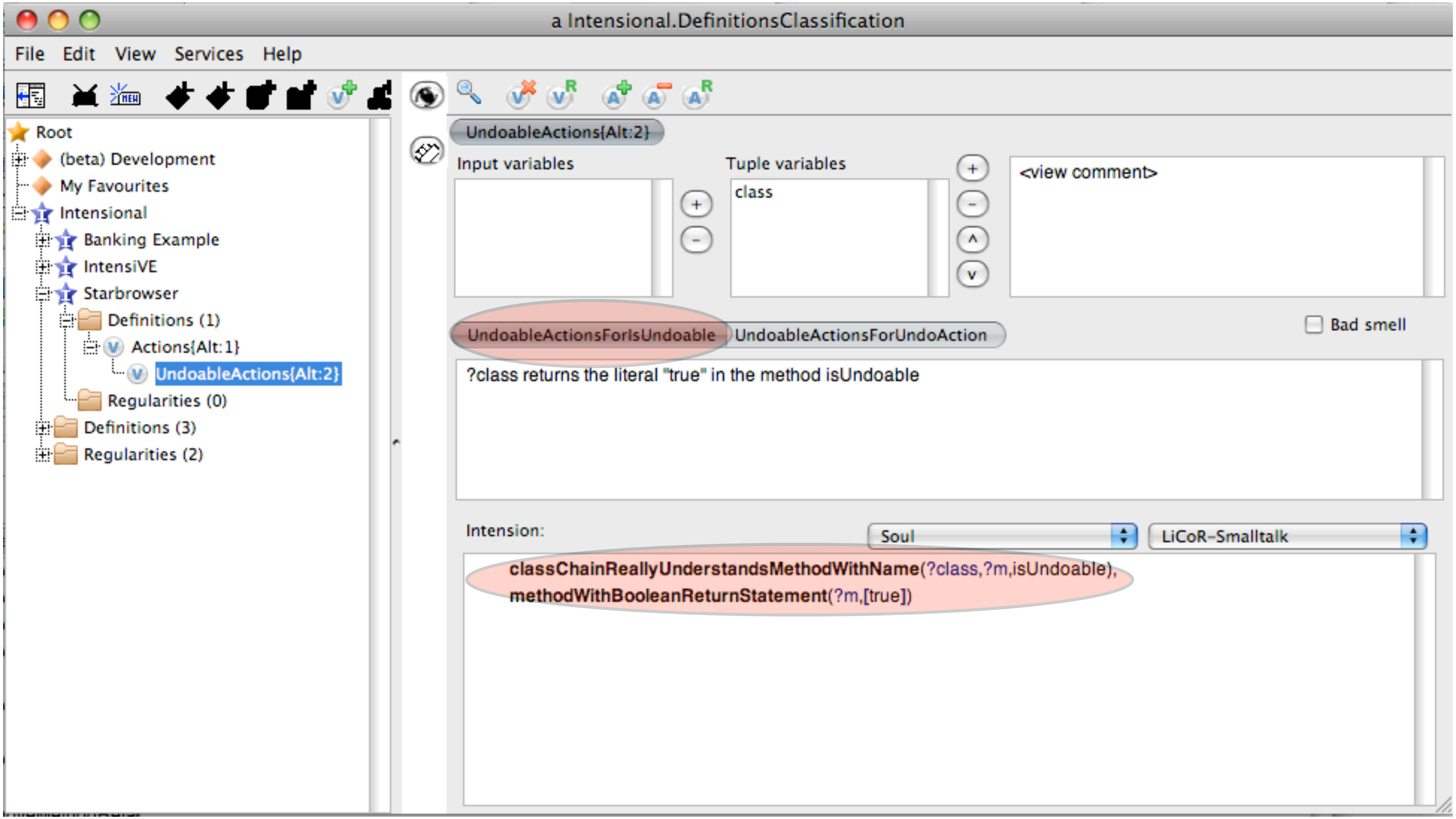


?class returns the literal "true" in the method isUndoable

```
classChainReallyUnderstandsMethodWithName(?class,?m,isUndoable),
methodWithBooleanReturnStatement(?m,[true])
```

IntensiVE

(consistency definition)



Tuple variables: class

<view comment>

□ Bad smell

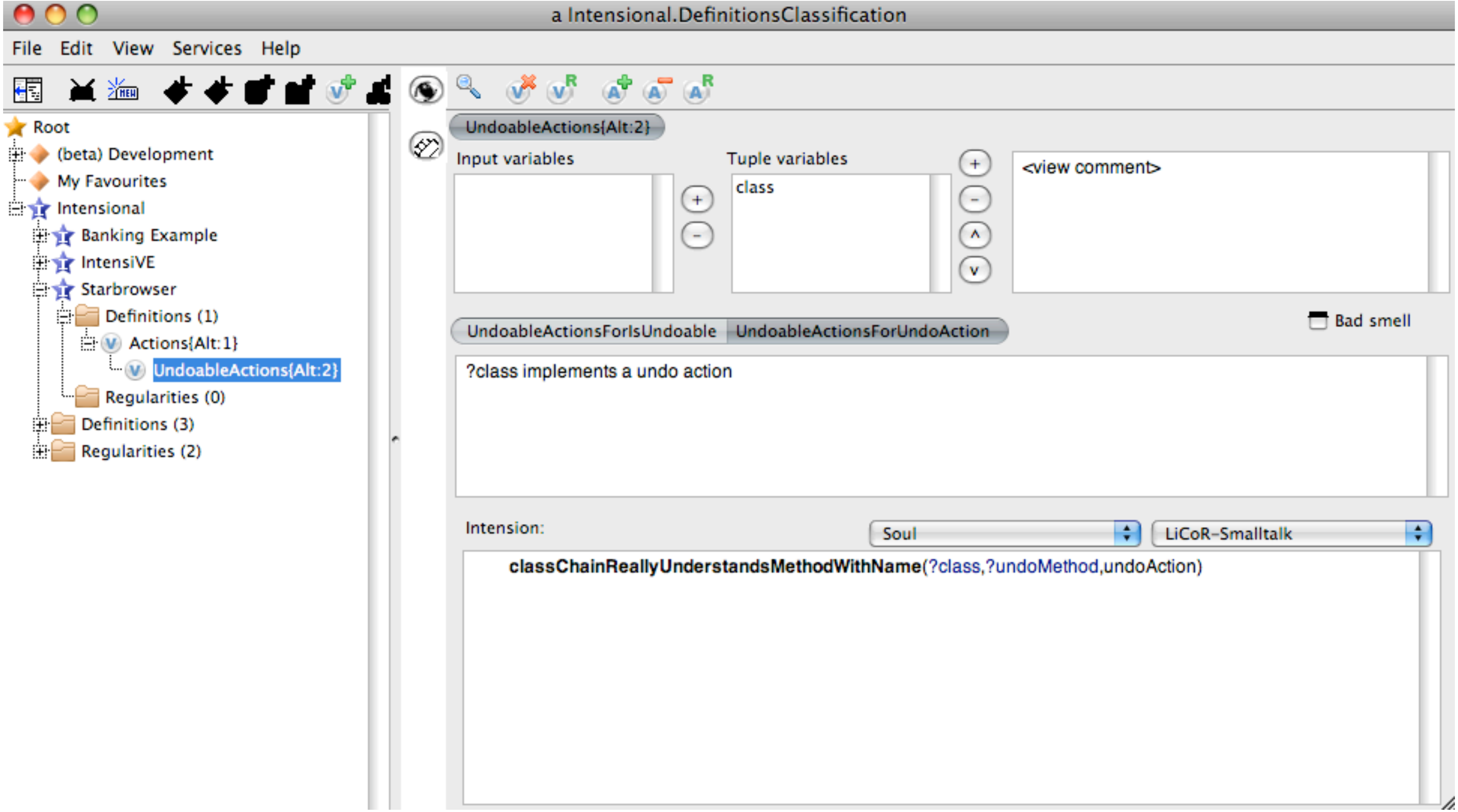
?class returns the literal "true" in the method isUndoable

Intension: Soul | LiCoR-Smalltalk

```
classChainReallyUnderstandsMethodWithName(?class,?m,isUndoable),
methodWithBooleanReturnStatement(?m,[true])
```

IntensiVE

(consistency definition)



Tuple variables: class

<view comment>

?class implements a undo action

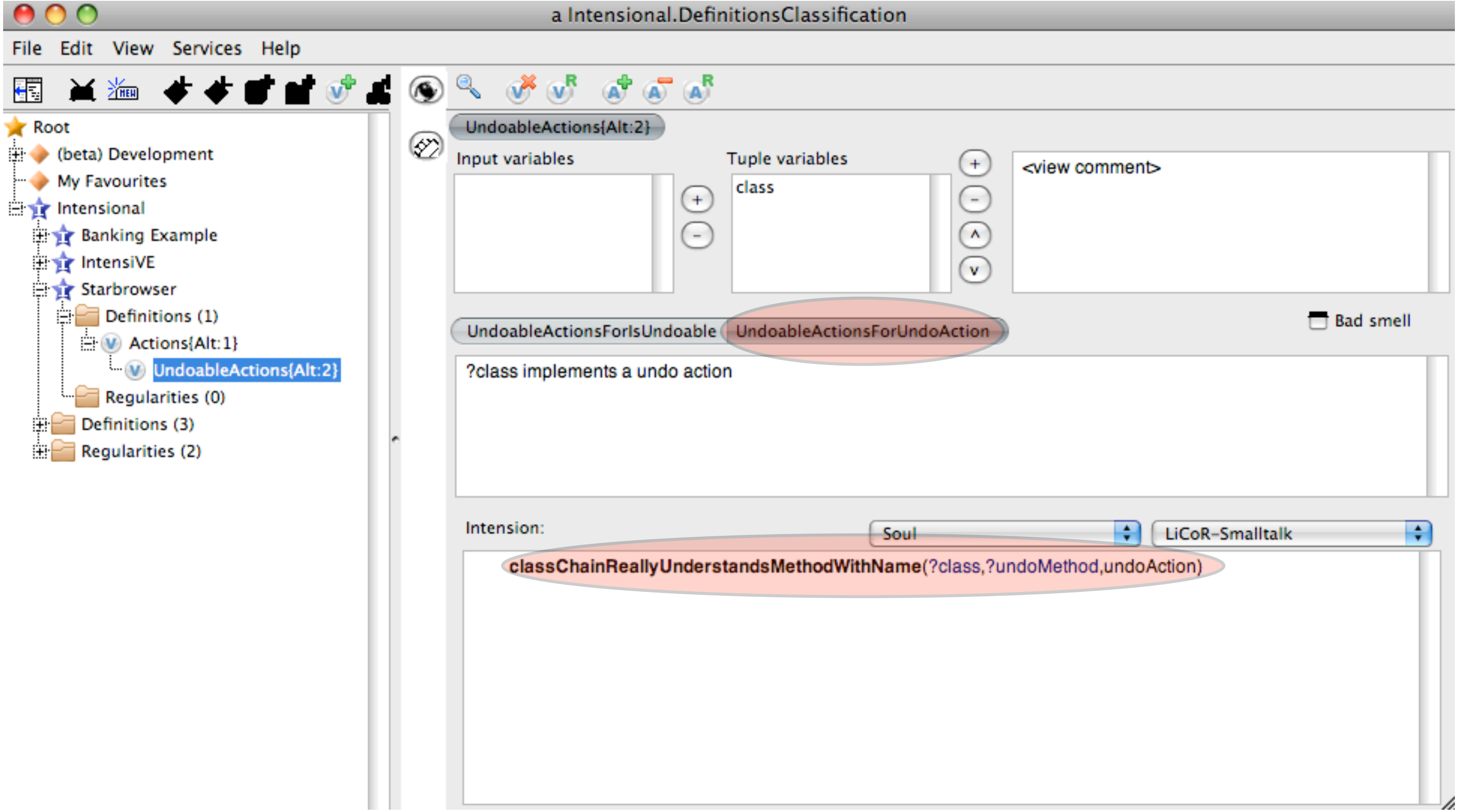
Soul

LiCoR-Smalltalk

```
classChainReallyUnderstandsMethodWithName(?class,?undoMethod,undoAction)
```

IntensiVE

(consistency definition)



class

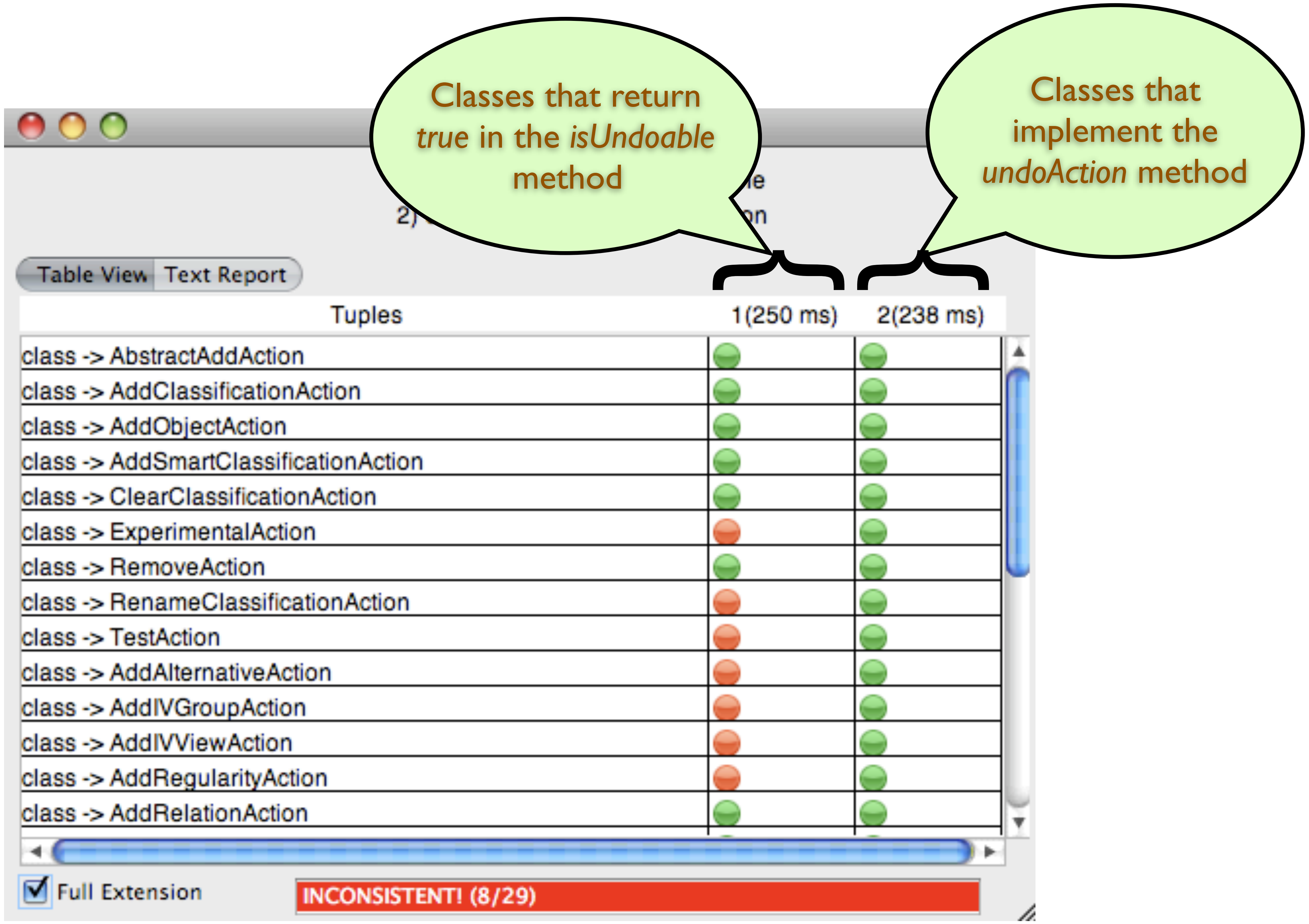
<view comment>

?class implements a undo action

```
classChainReallyUnderstandsMethodWithName(?class,?undoMethod,undoAction)
```

IntensiVE

(consistency checking)



Classes that return *true* in the *isUndoable* method

Classes that implement the *undoAction* method

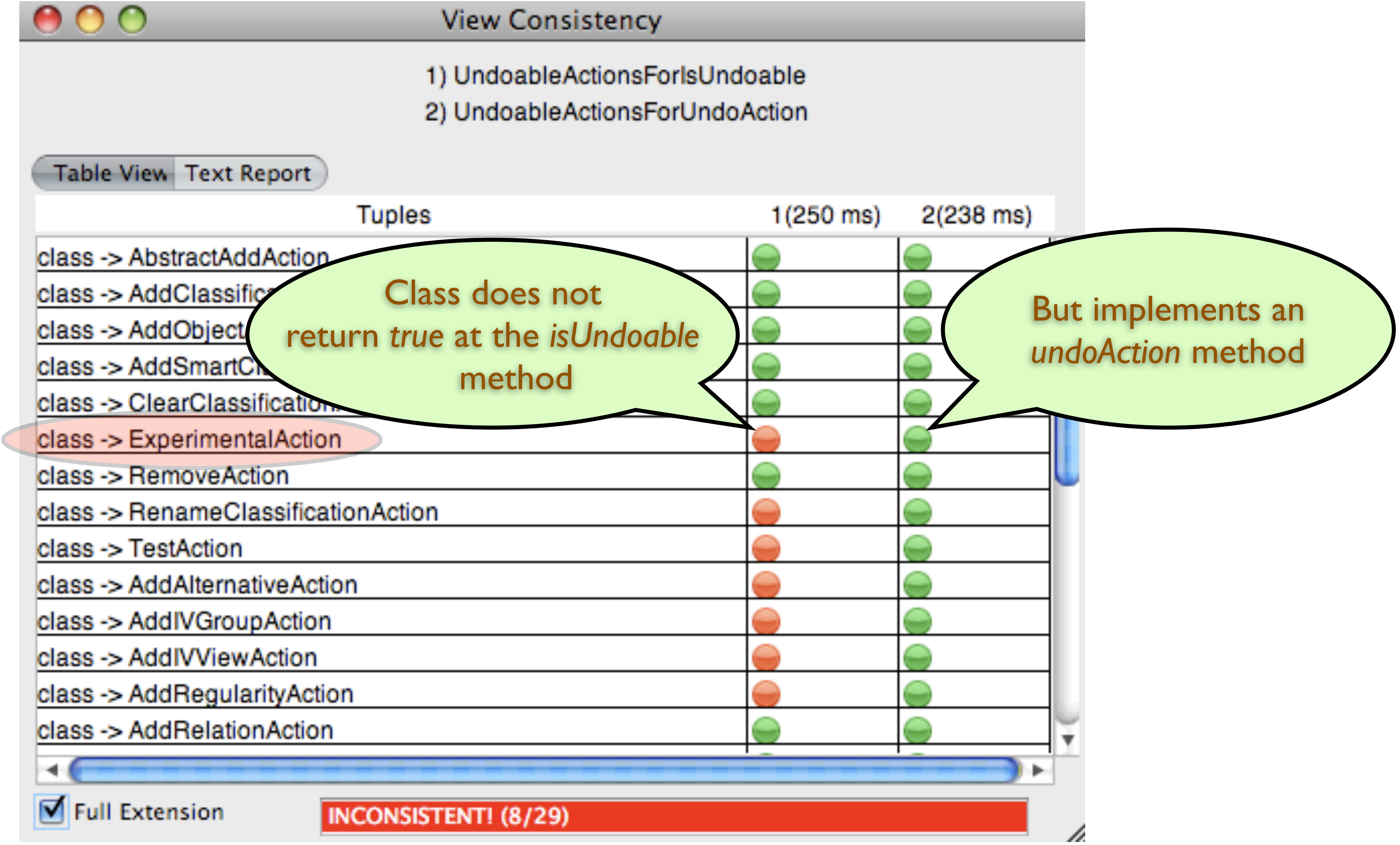
Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassificationAction	🟢	🟢
class -> AddObjectAction	🟢	🟢
class -> AddSmartClassificationAction	🟢	🟢
class -> ClearClassificationAction	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴	🟢
class -> TestAction	🔴	🟢
class -> AddAlternativeAction	🔴	🟢
class -> AddIVGroupAction	🔴	🟢
class -> AddIVViewAction	🔴	🟢
class -> AddRegularityAction	🔴	🟢
class -> AddRelationAction	🟢	🟢

☑ Full Extension

INCONSISTENT! (8/29)

IntensiVE

(consistency checking)



View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Table View | Text Report

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassific	🟢	🟢
class -> AddObject	🟢	🟢
class -> AddSmartCl	🟢	🟢
class -> ClearClassification	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴	🟢
class -> TestAction	🔴	🟢
class -> AddAlternativeAction	🔴	🟢
class -> AddIVGroupAction	🔴	🟢
class -> AddIVViewAction	🔴	🟢
class -> AddRegularityAction	🔴	🟢
class -> AddRelationAction	🟢	🟢

☑ Full Extension

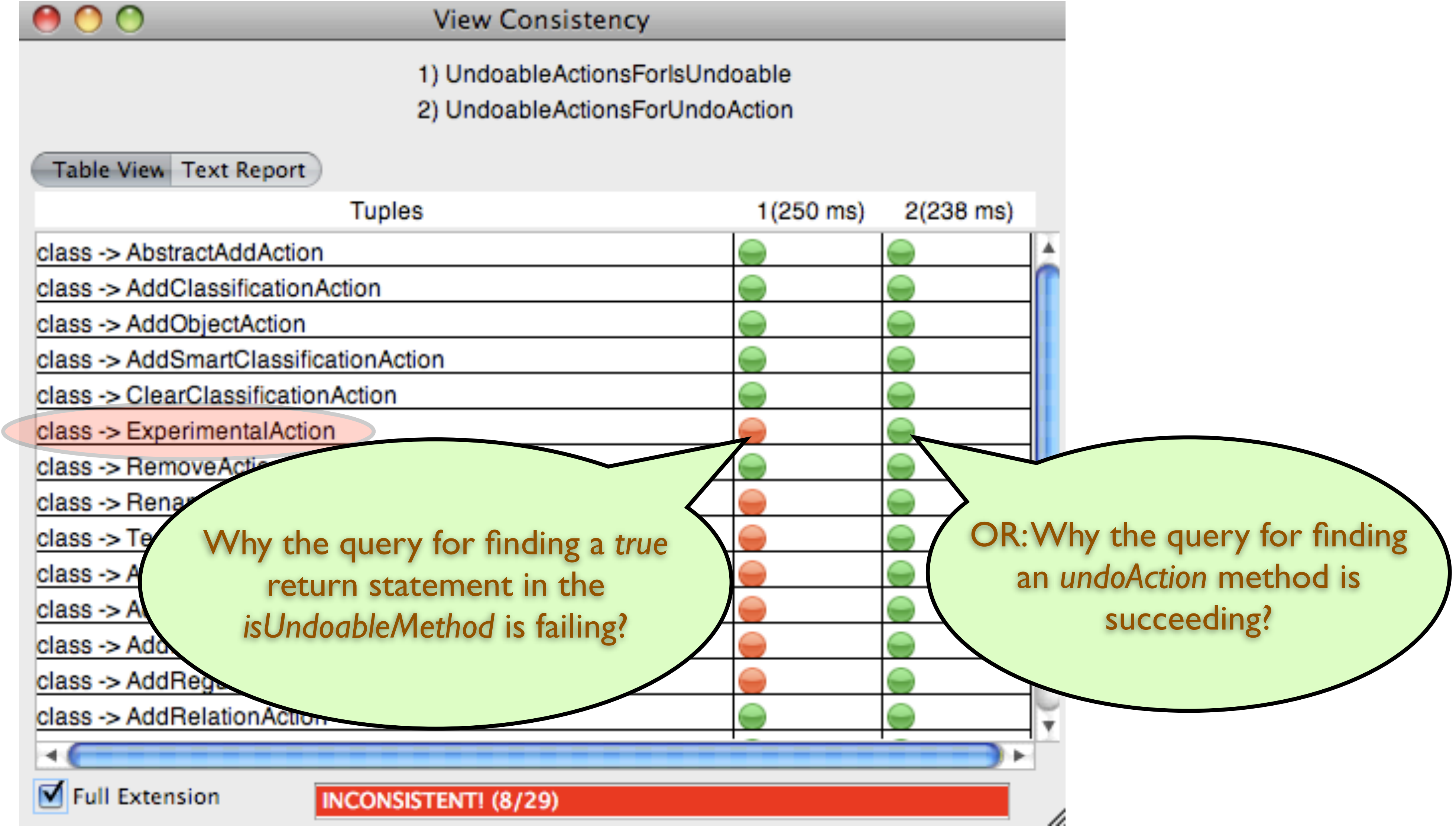
INCONSISTENT! (8/29)

Class does not return *true* at the *isUndoable* method

But implements an *undoAction* method

IntensiVE

(diagnosing inconsistencies)



View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassificationAction	🟢	🟢
class -> AddObjectAction	🟢	🟢
class -> AddSmartClassificationAction	🟢	🟢
class -> ClearClassificationAction	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveActi	🟢	🟢
class -> Rena	🔴	🟢
class -> Te	🔴	🟢
class -> A	🔴	🟢
class -> A	🔴	🟢
class -> Add	🔴	🟢
class -> AddReg	🔴	🟢
class -> AddRelationAction	🟢	🟢

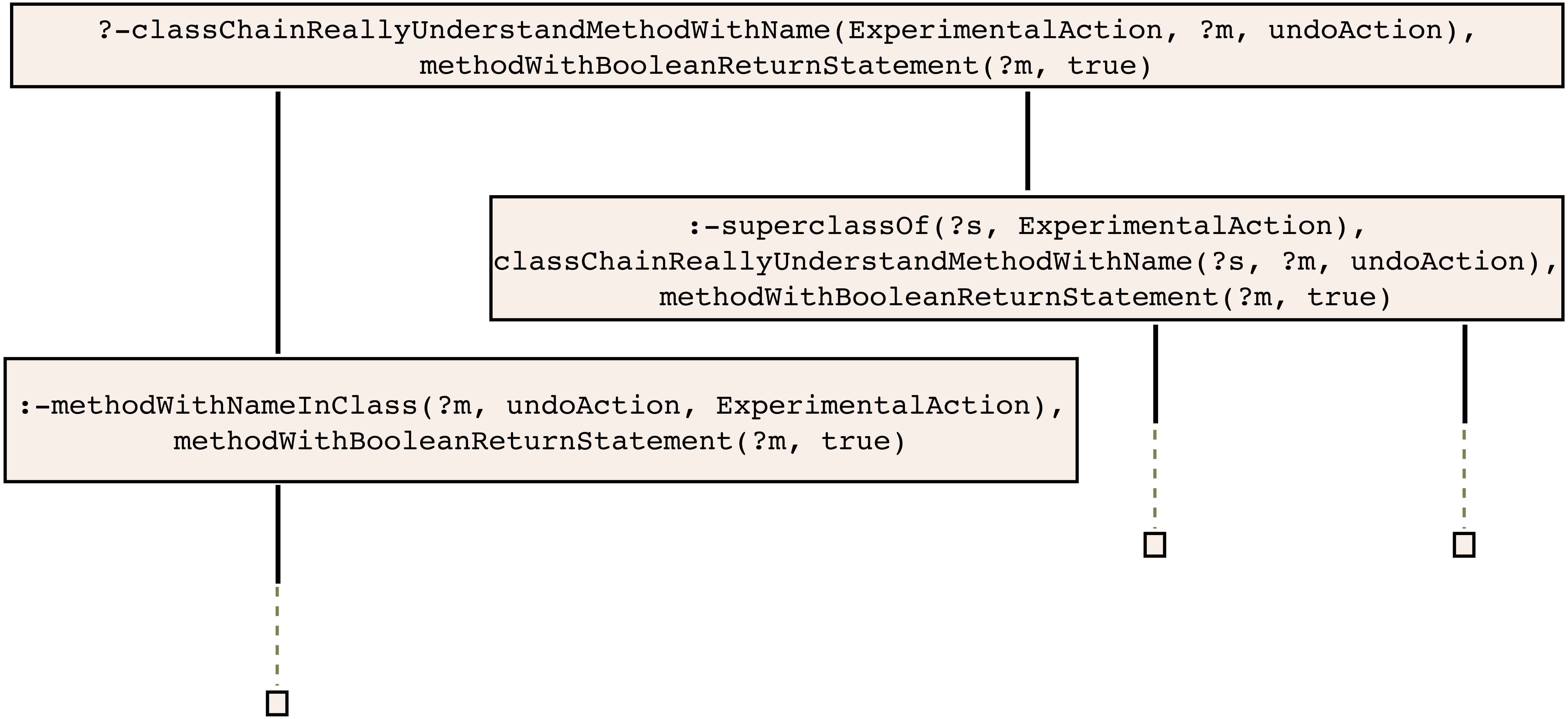
☑ Full Extension

INCONSISTENT! (8/29)

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



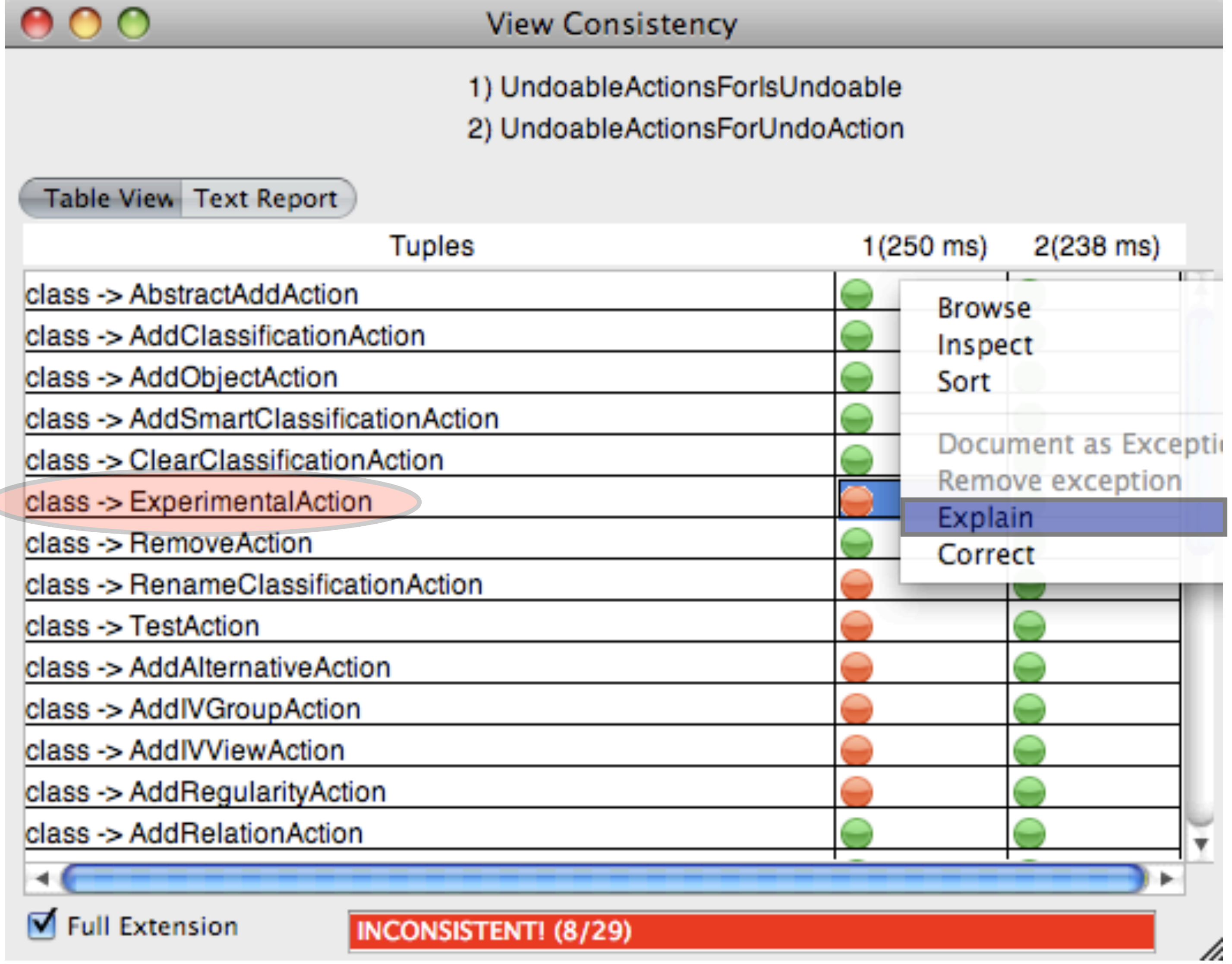
?-classChainReallyUnderstandMethodWithName(ExperimentalAction, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)

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

:-methodWithNameInClass(?m, undoAction, ExperimentalAction),
methodWithBooleanReturnStatement(?m, true)

IntensiVE

(diagnosing inconsistencies)



View Consistency

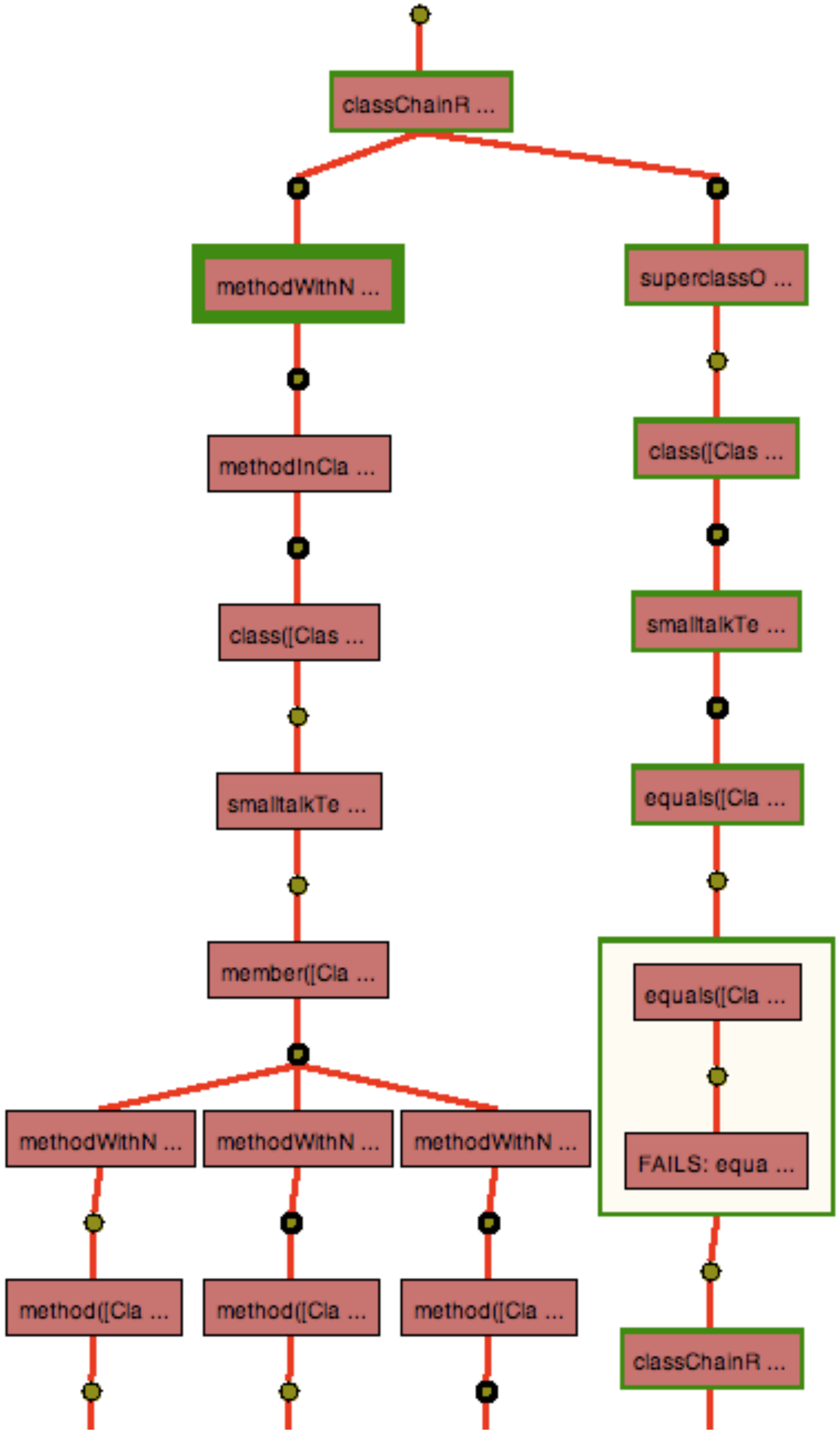
1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	
class -> AddClassificationAction	🟢	
class -> AddObjectAction	🟢	
class -> AddSmartClassificationAction	🟢	
class -> ClearClassificationAction	🟢	
class -> ExperimentalAction	🔴	
class -> RemoveAction	🟢	
class -> RenameClassificationAction	🔴	🟢
class -> TestAction	🔴	🟢
class -> AddAlternativeAction	🔴	🟢
class -> AddIVGroupAction	🔴	🟢
class -> AddIVViewAction	🔴	🟢
class -> AddRegularityAction	🔴	🟢
class -> AddRelationAction	🟢	🟢

☑ Full Extension

INCONSISTENT! (8/29)

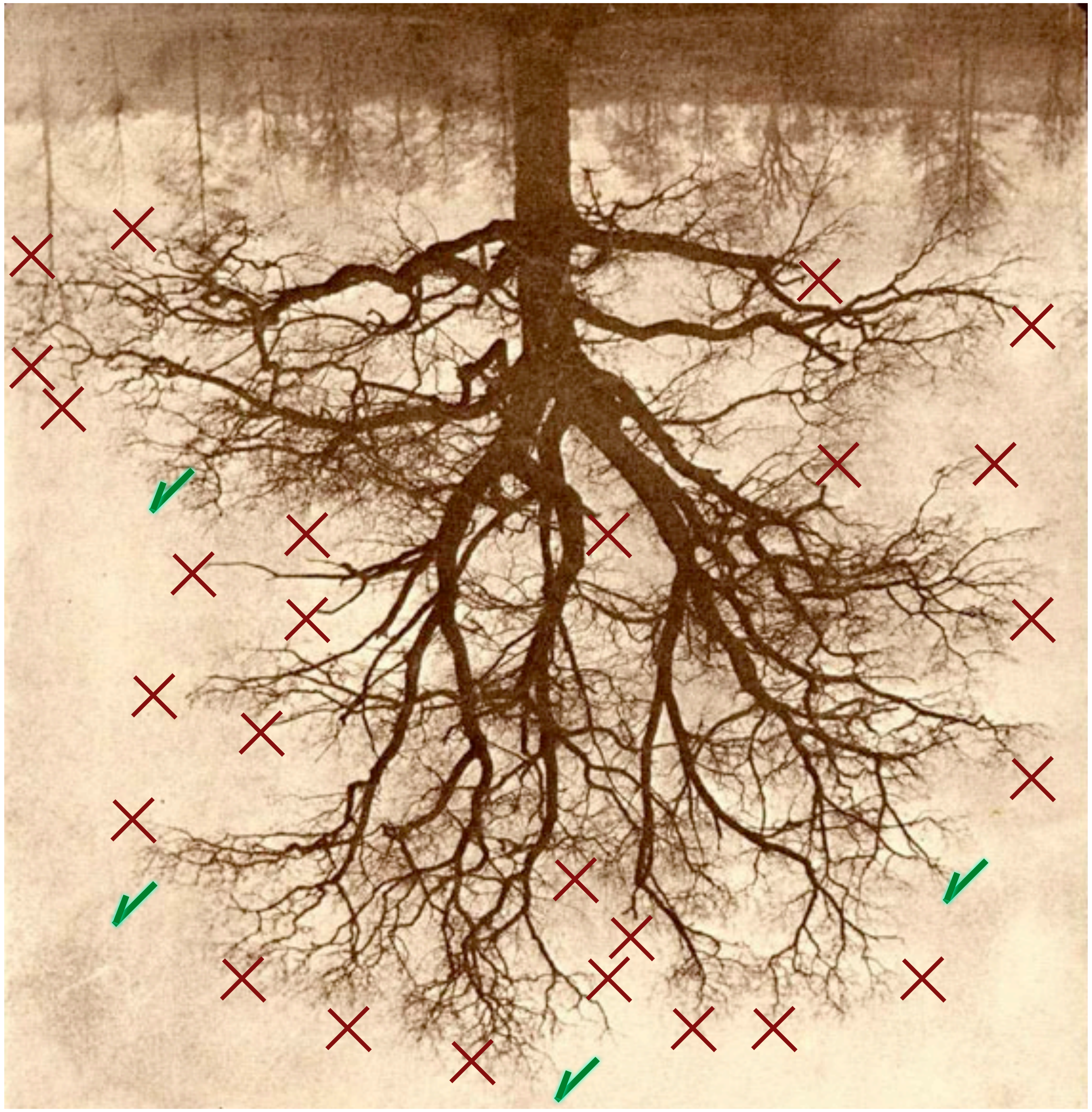
A proof tree with our tool



But …

What does a real SLD tree look like?

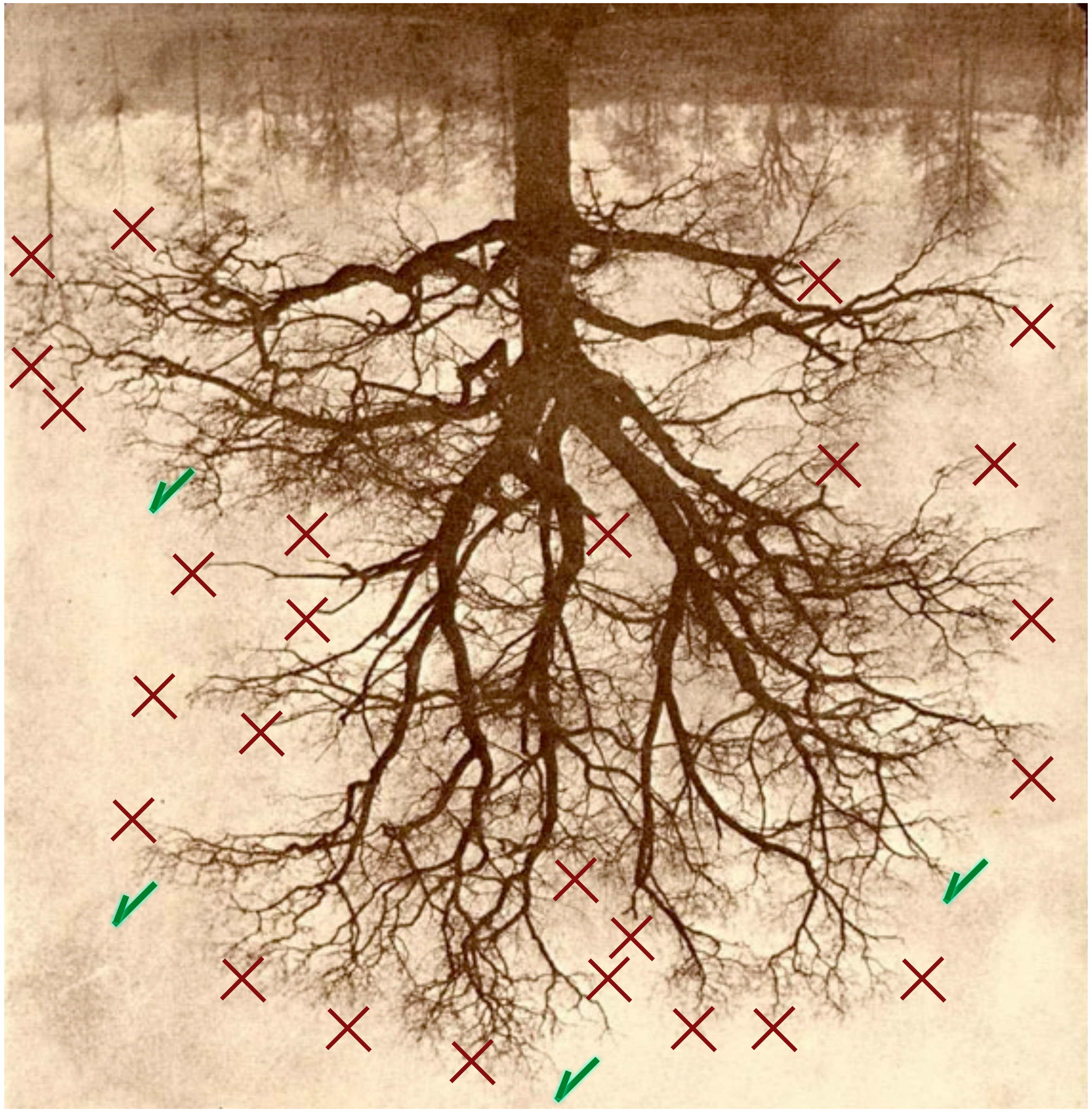
What does a real SLD tree look like?



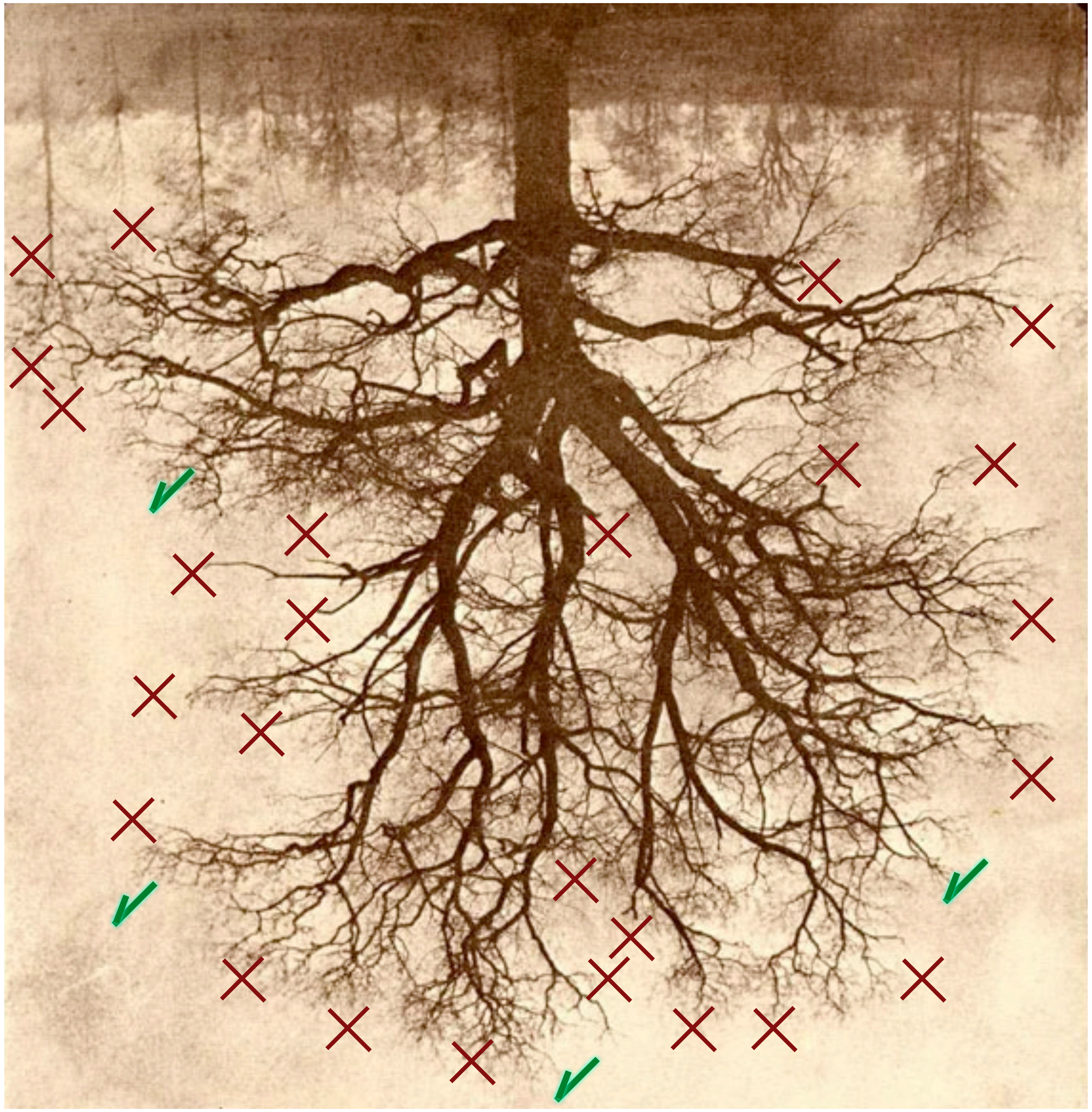
✓ Success

× Failure

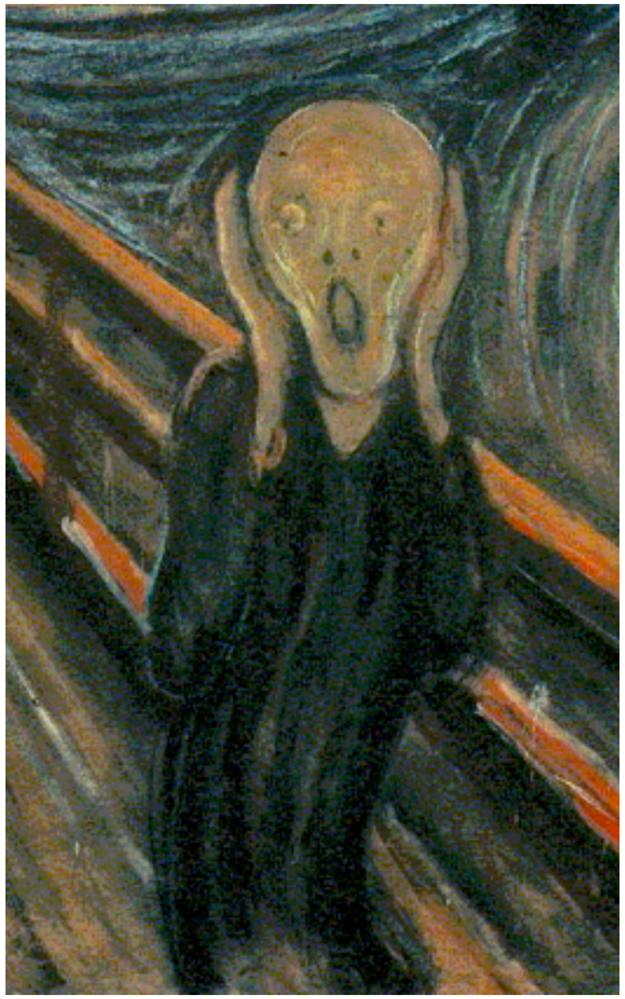
What does a real SLD tree look like?



What does a real SLD tree look like?

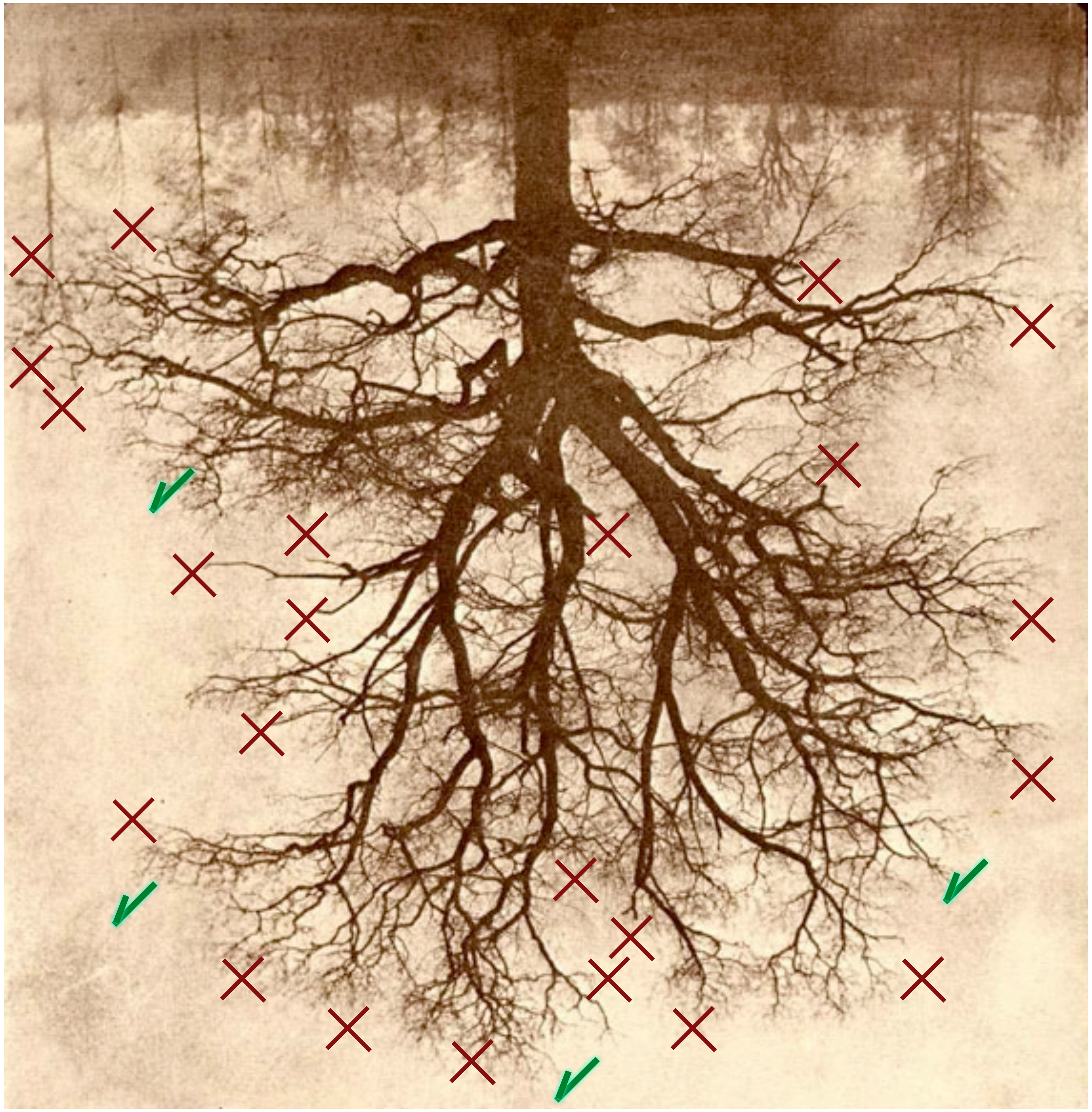


Where is the cause of the problem?



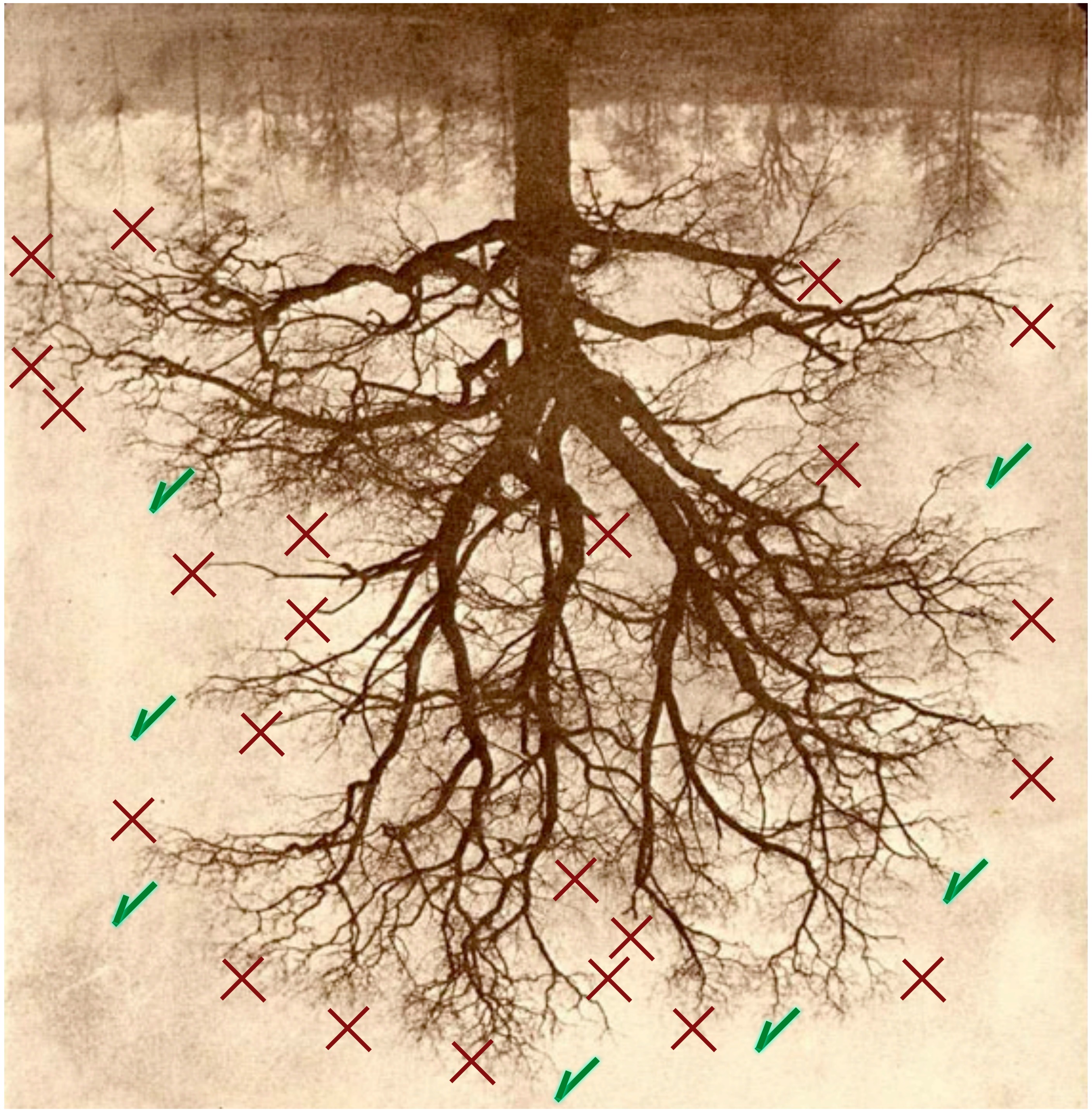
How to solve it?

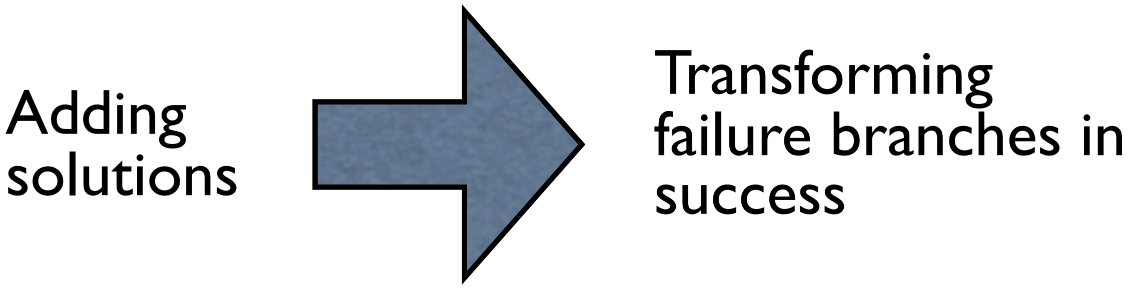
The result of a query can be changed ...



Adding solutions → Transforming failure branches in success

The result of a query can be changed ...



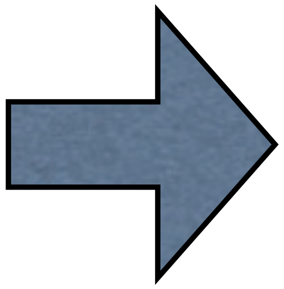


Adding solutions

Transforming failure branches in success

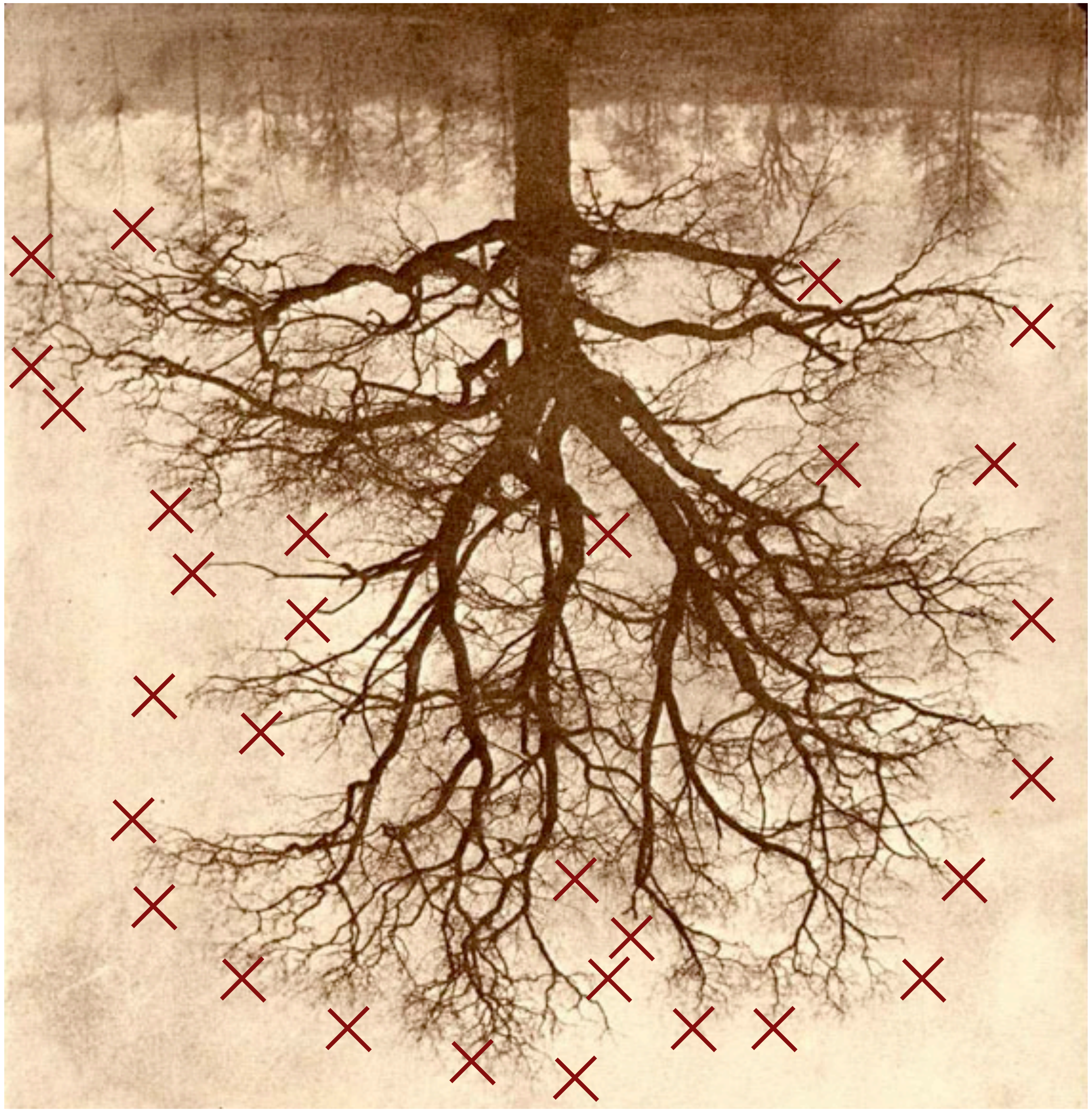
The result of a query can be changed ...

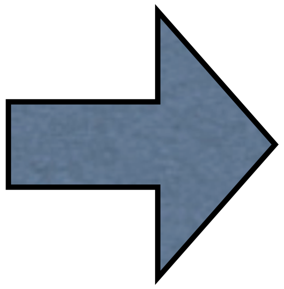
Suppressing Solutions



Transforming success branches in failures

The result of a query can be changed ...





Suppressing Solutions

Transforming success branches in failures

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

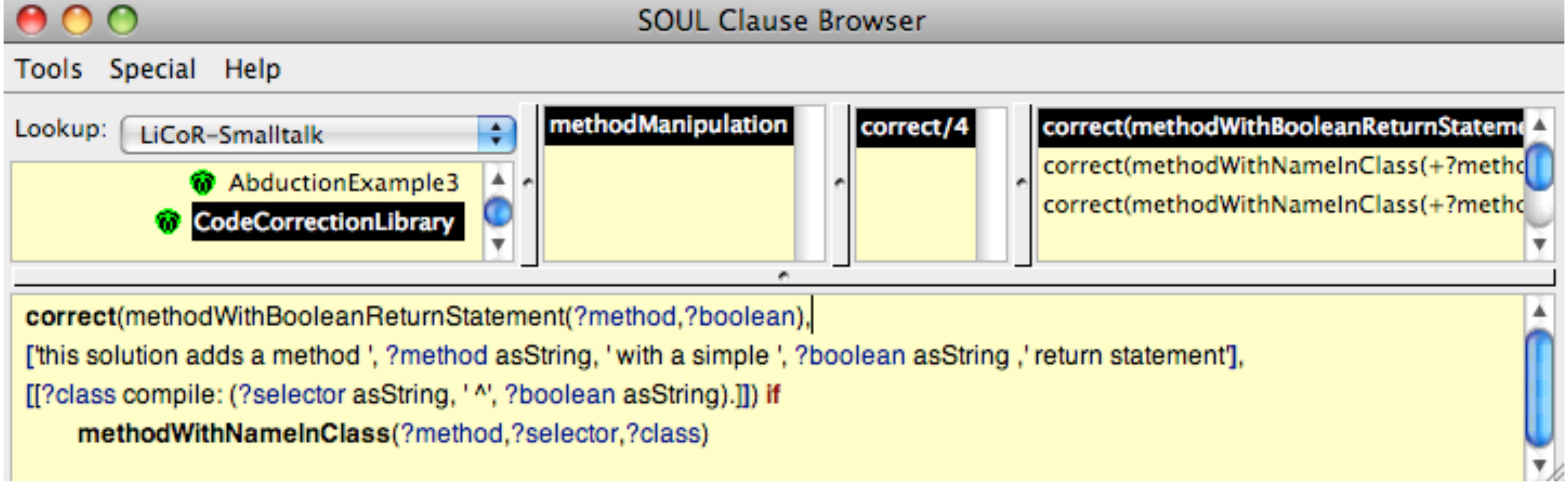
- Deduction
- Induction
- Abduction

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

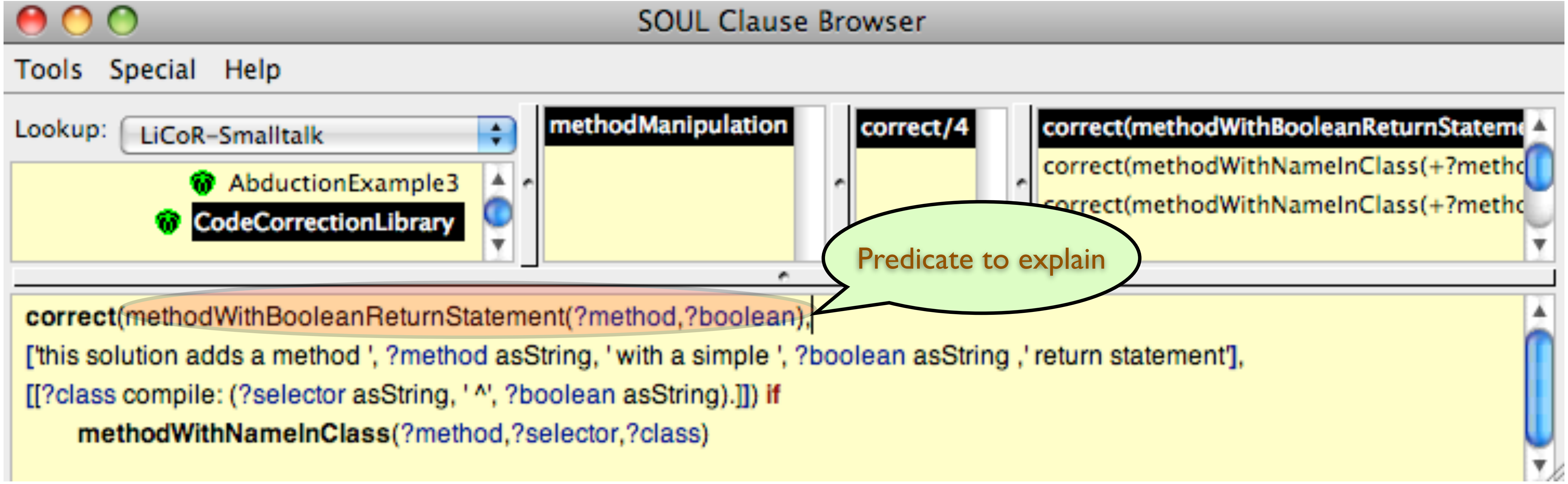
(*abducible* predicates)



```
correct(methodWithBooleanReturnStatement(?method,?boolean),
['this solution adds a method ', ?method asString, ' with a simple ', ?boolean asString ,' return statement'],
[[?class compile: (?selector asString, ' ^', ?boolean asString).]]) if
     methodWithNameInClass(?method,?selector,?class)
```

Defining positive explanations

(*abducible* predicates)

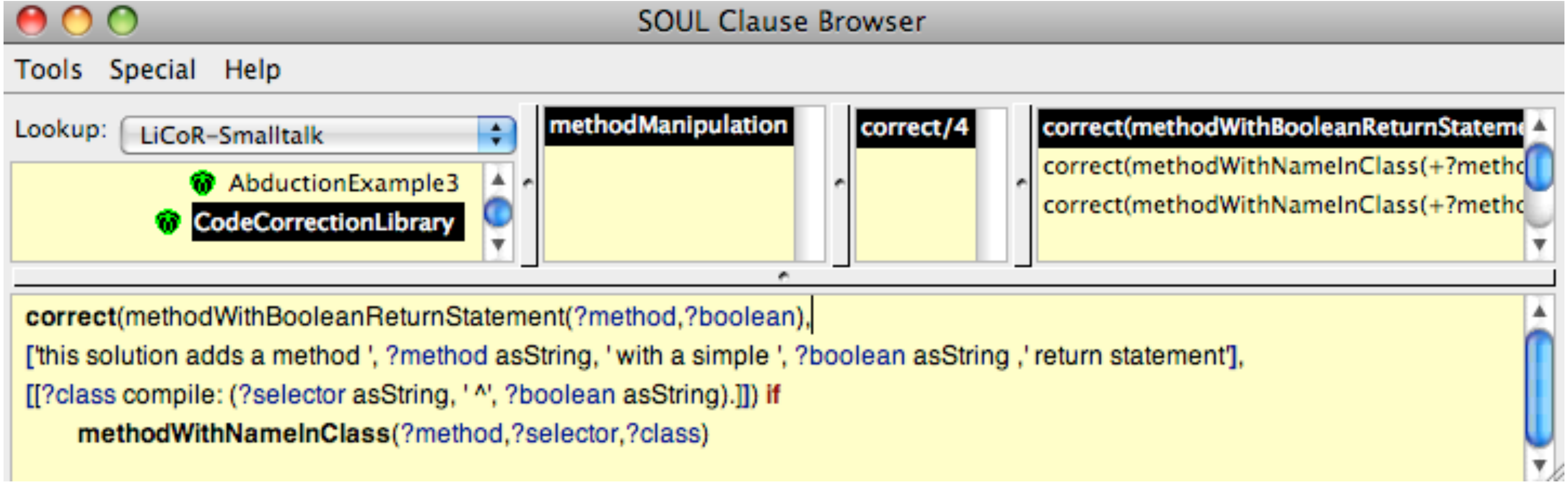


Predicate to explain

```
correct(methodWithBooleanReturnStatement(?method,?boolean),
['this solution adds a method ', ?method asString, ' with a simple ', ?boolean asString ,' return statement'],
[[?class compile: (?selector asString, ' ^', ?boolean asString).]]) if
    methodWithNameInClass(?method,?selector,?class)
```

Defining positive explanations

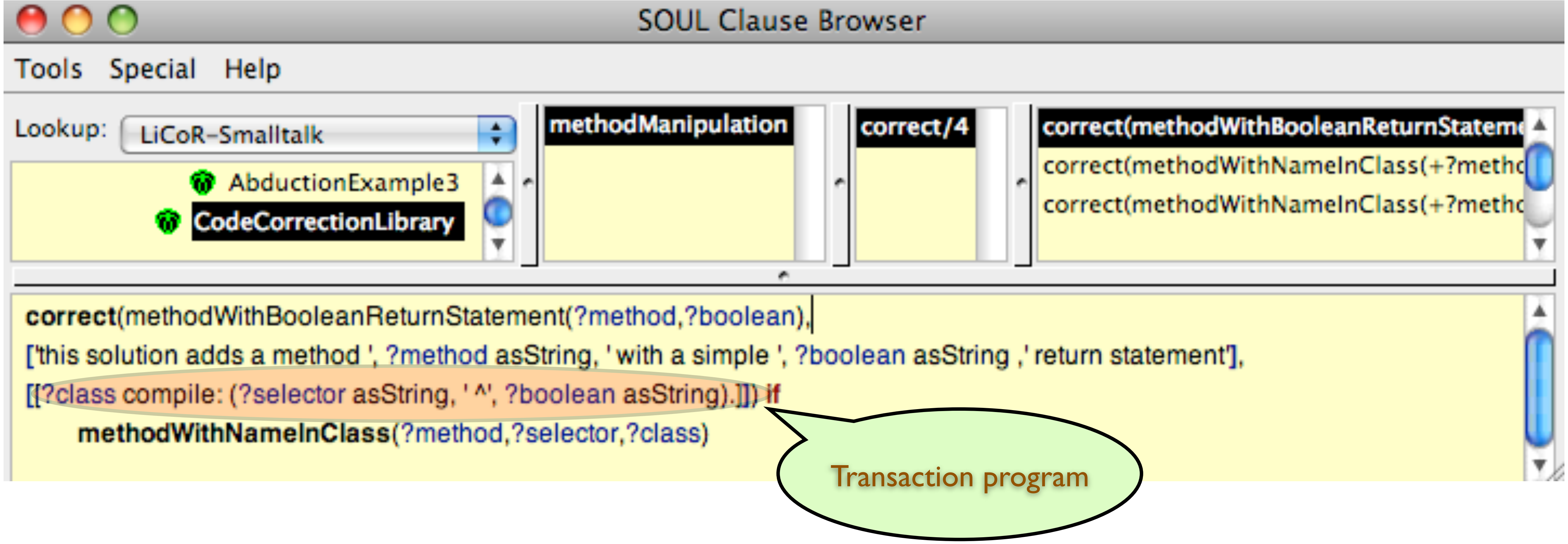
(*abducible* predicates)



```
correct(methodWithBooleanReturnStatement(?method,?boolean),
['this solution adds a method ', ?method asString, ' with a simple ', ?boolean asString ,' return statement'],
[[?class compile: (?selector asString, ' ^', ?boolean asString).]]) if
     methodWithNameInClass(?method,?selector,?class)
```

Defining positive explanations

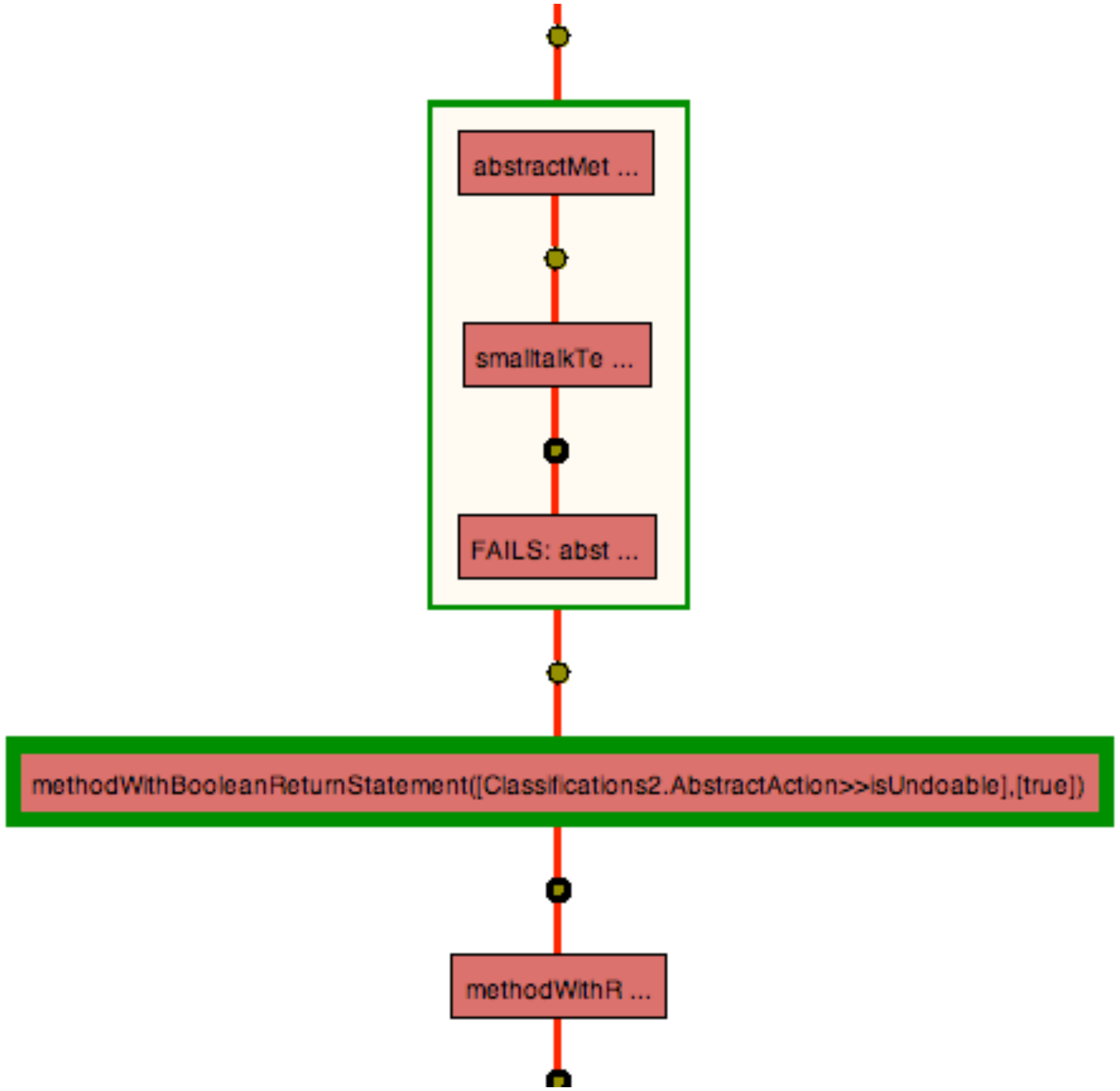
(*abducible* predicates)



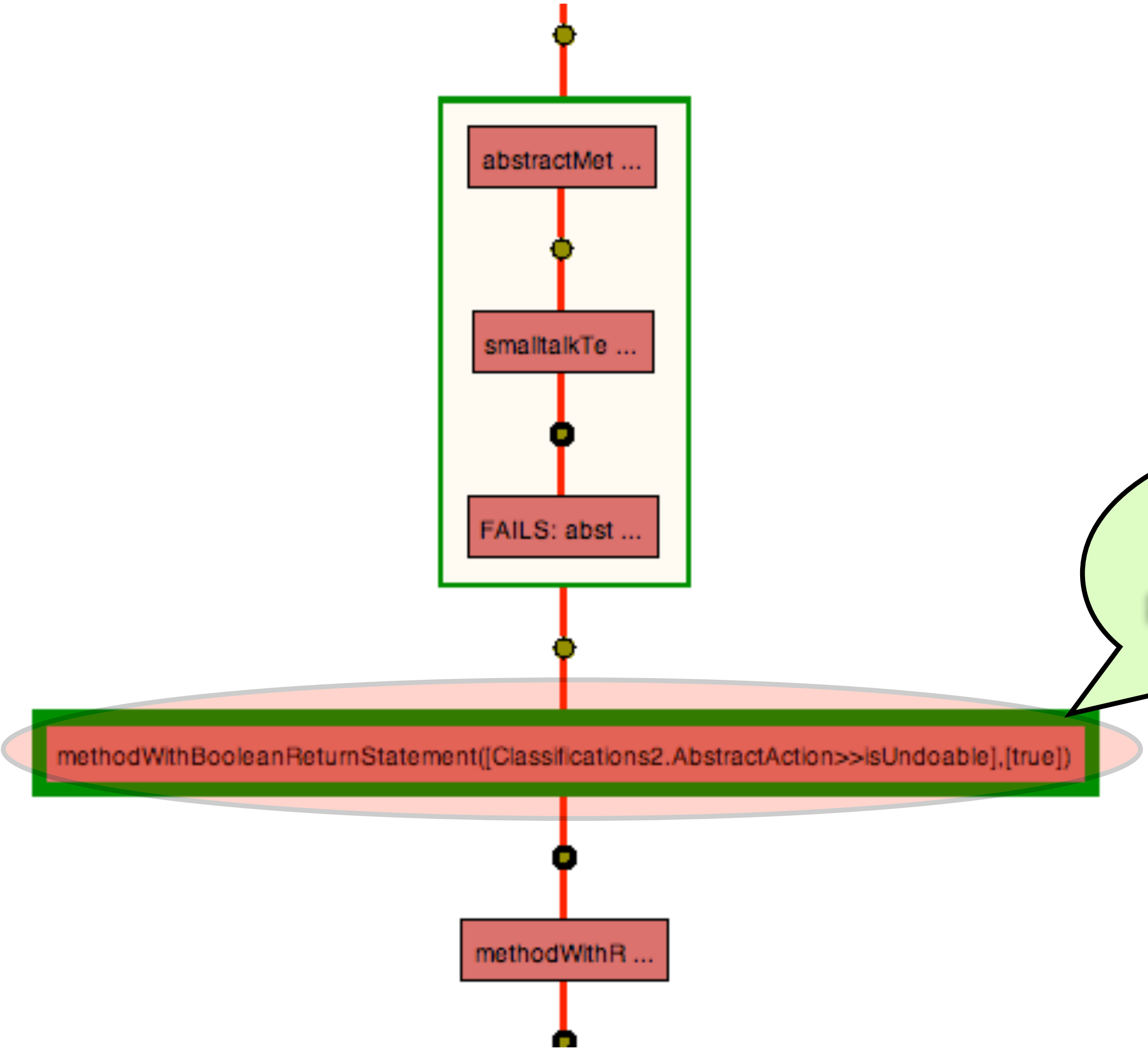
```
correct(methodWithBooleanReturnStatement(?method,?boolean),
['this solution adds a method ', ?method asString, ' with a simple ', ?boolean asString ,' return statement'],
[[?class compile: (?selector asString, ' ^', ?boolean asString).]]) if
    methodWithNameInClass(?method,?selector,?class)
```

Transaction program

Diagnosing & correcting inconsistenies


abstractMet ...
smalltalkTe ...
FAILS: abst ...
methodWithBooleanReturnStatement([Classifications2.AbstractAction>>isUndoable],[true])
methodWithR ...

Diagnosing & correcting inconsistenies


abstractMet ...
smalltalkTe ...
FAILS: abst ...
methodWithBooleanReturnStatement([Classifications2.AbstractAction>>isUndoable],[true])
methodWithR ...

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?