

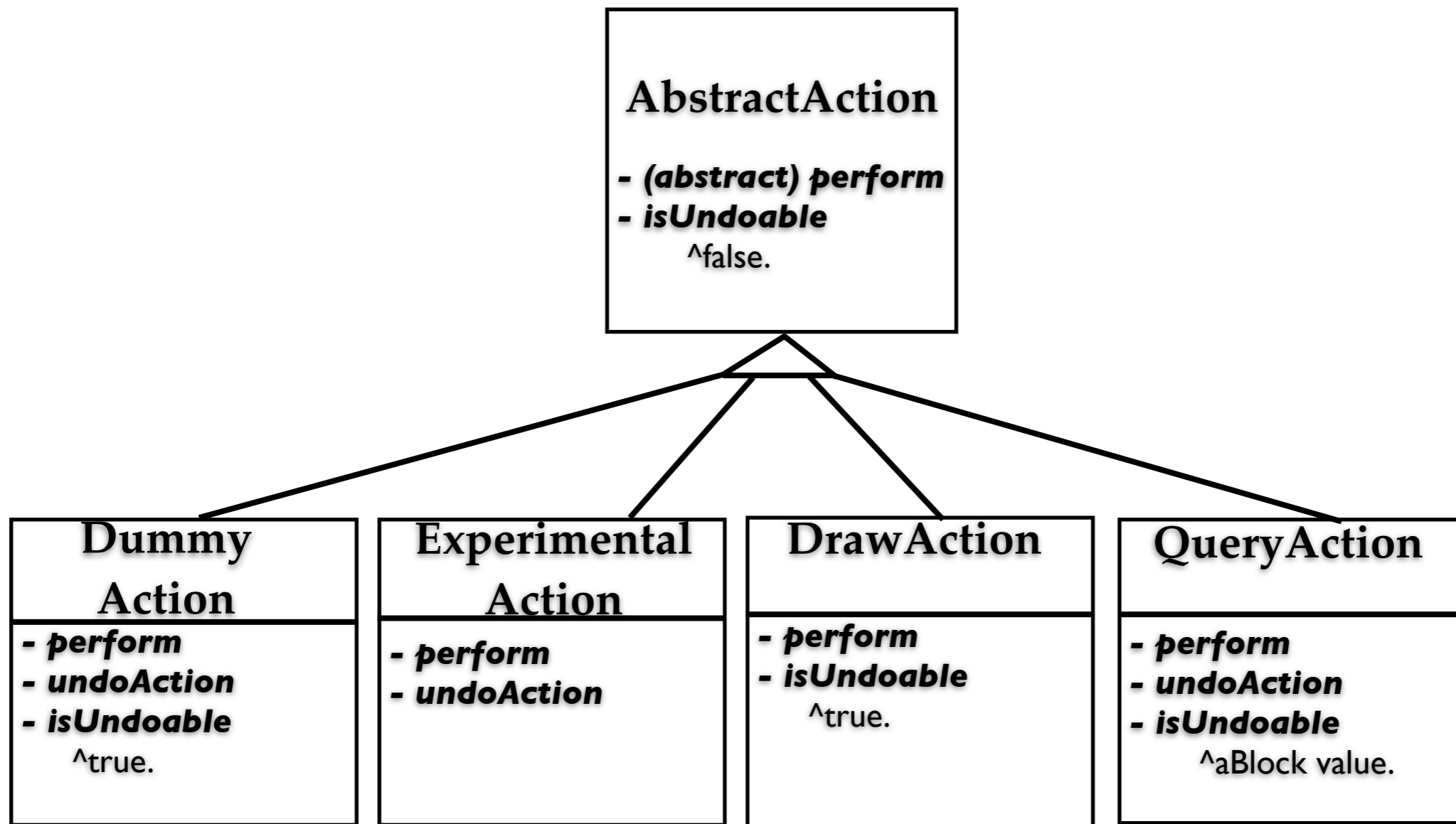


Diagnosis and semi-automatic correction of detected design inconsistencies in source code

Sergio Castro
***RELEASED* lab**

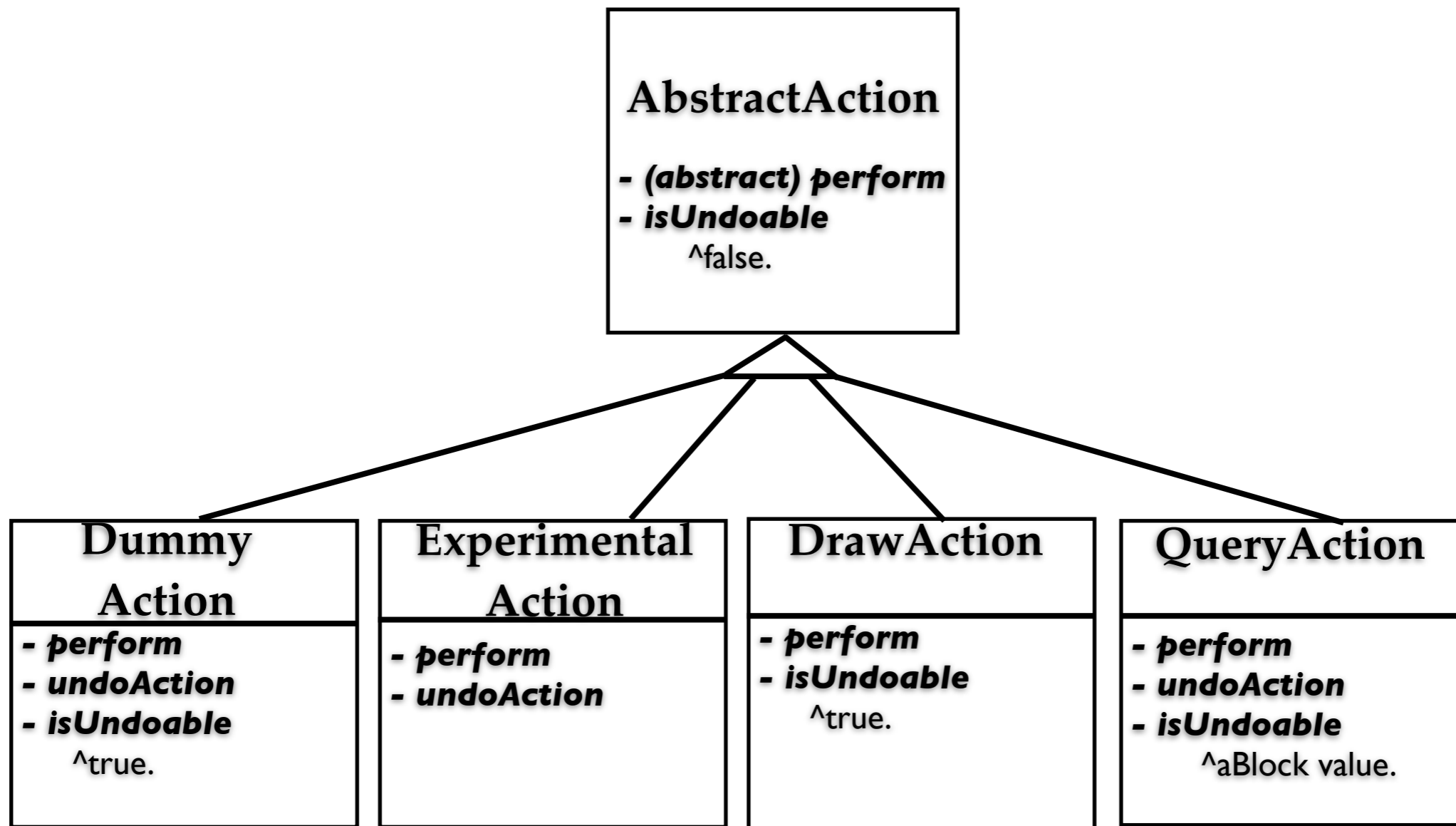
Université catholique de Louvain
sergio.castro@uclouvain.be
Advisor: Kim Mens

An inconsistency example



An inconsistency example

The command design pattern



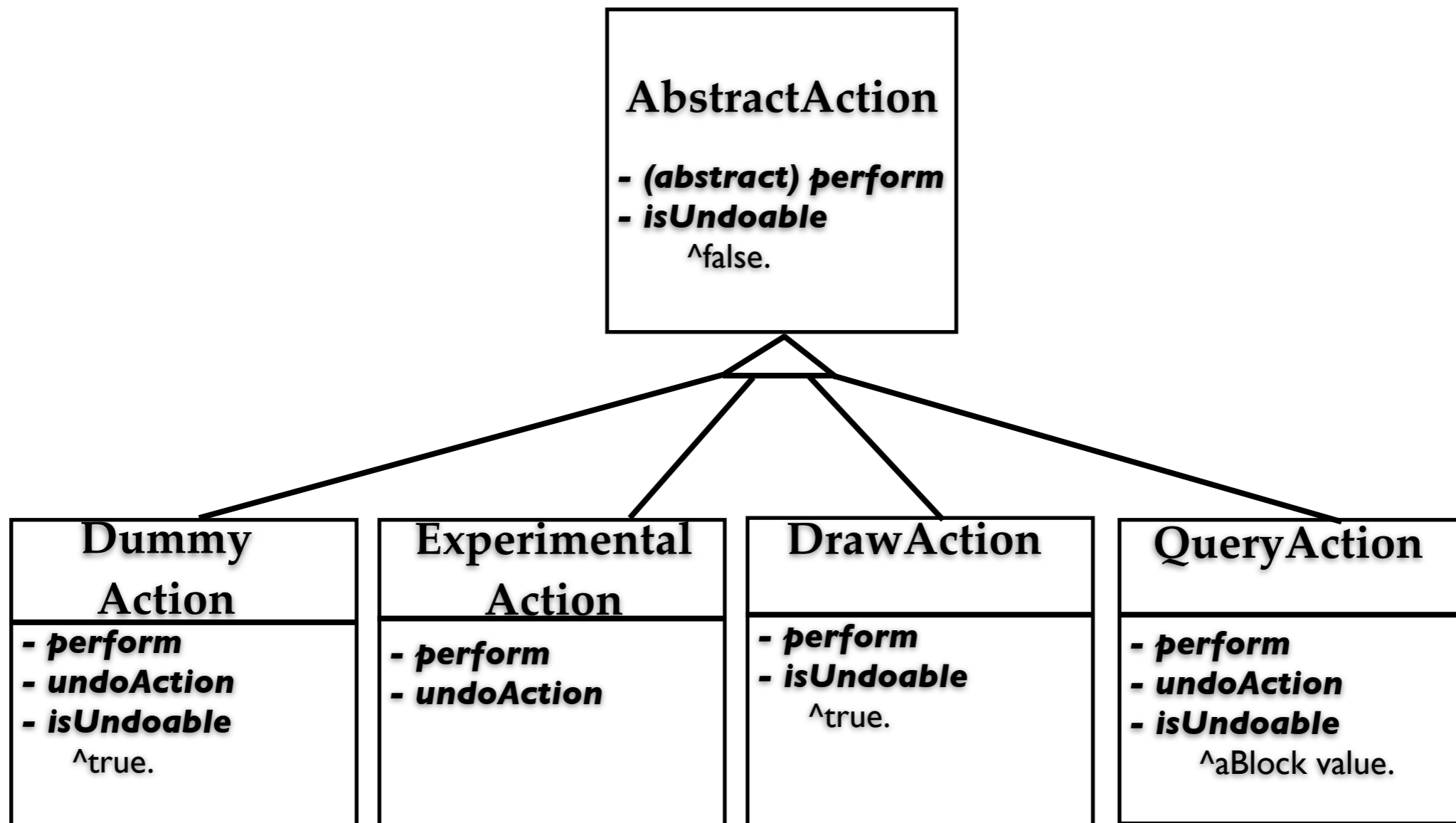
An inconsistency example

The command design pattern

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*



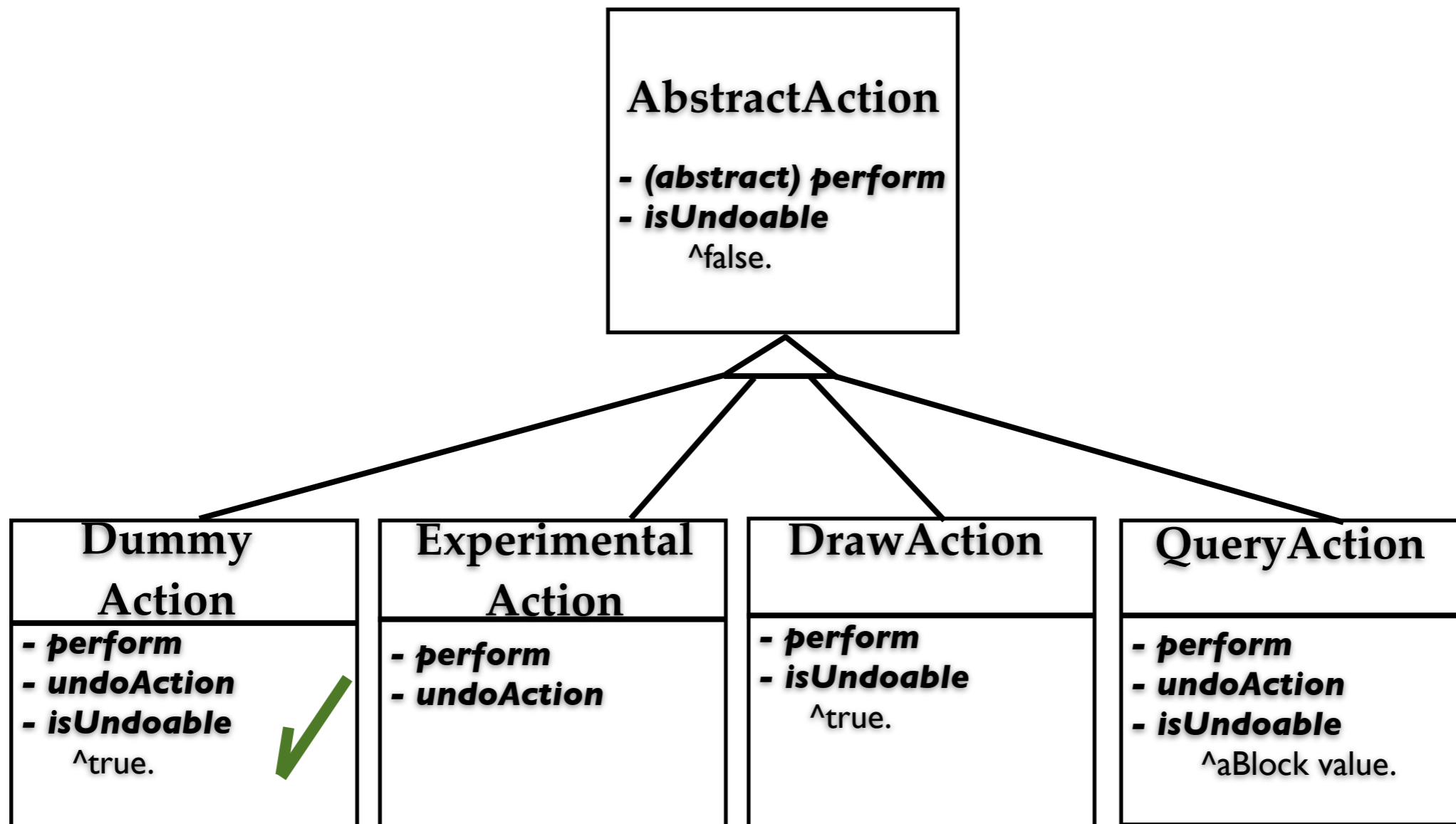
An inconsistency example

The command design pattern

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*



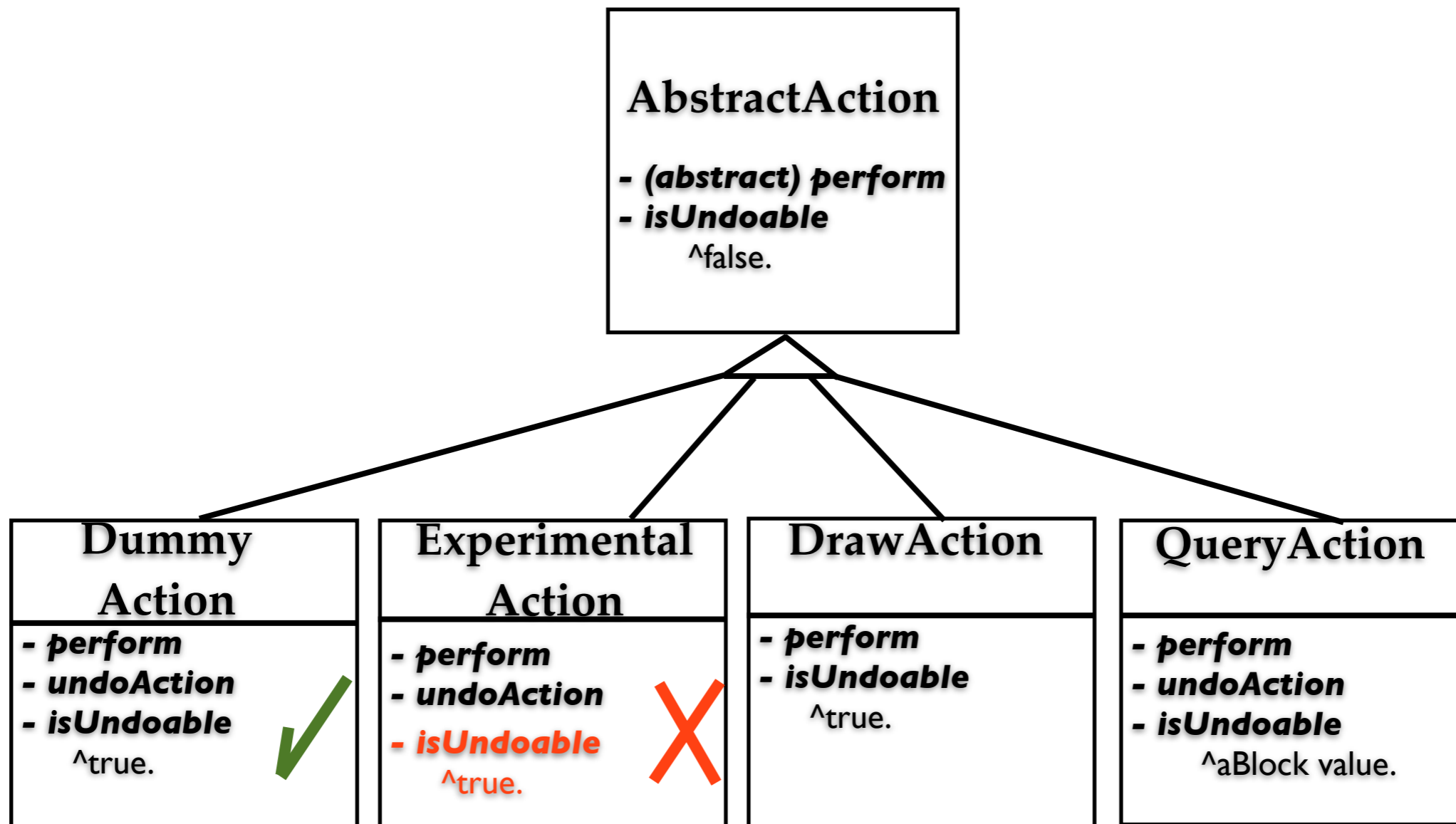
An inconsistency example

The command design pattern

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*



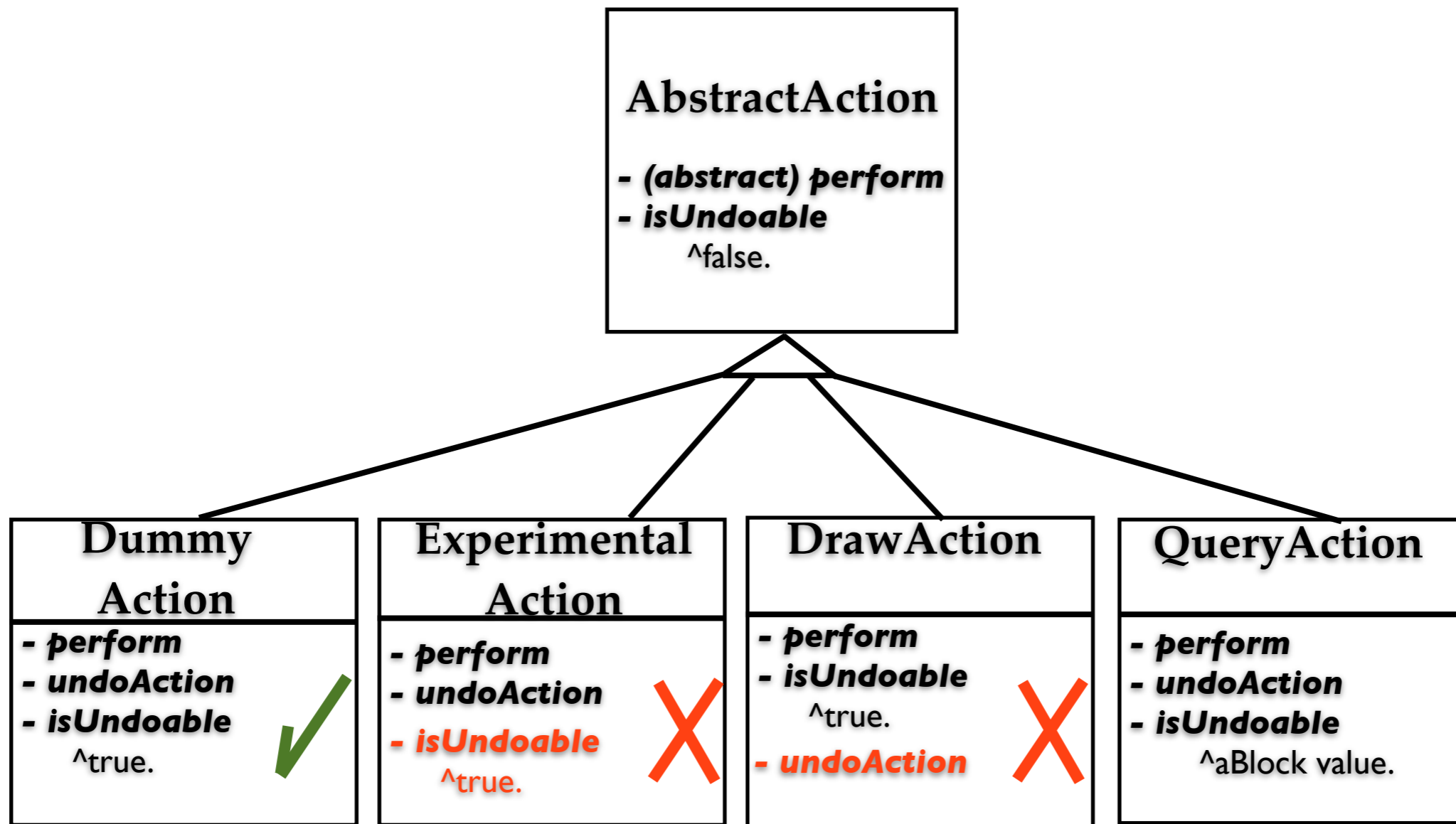
An inconsistency example

The command design pattern

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*



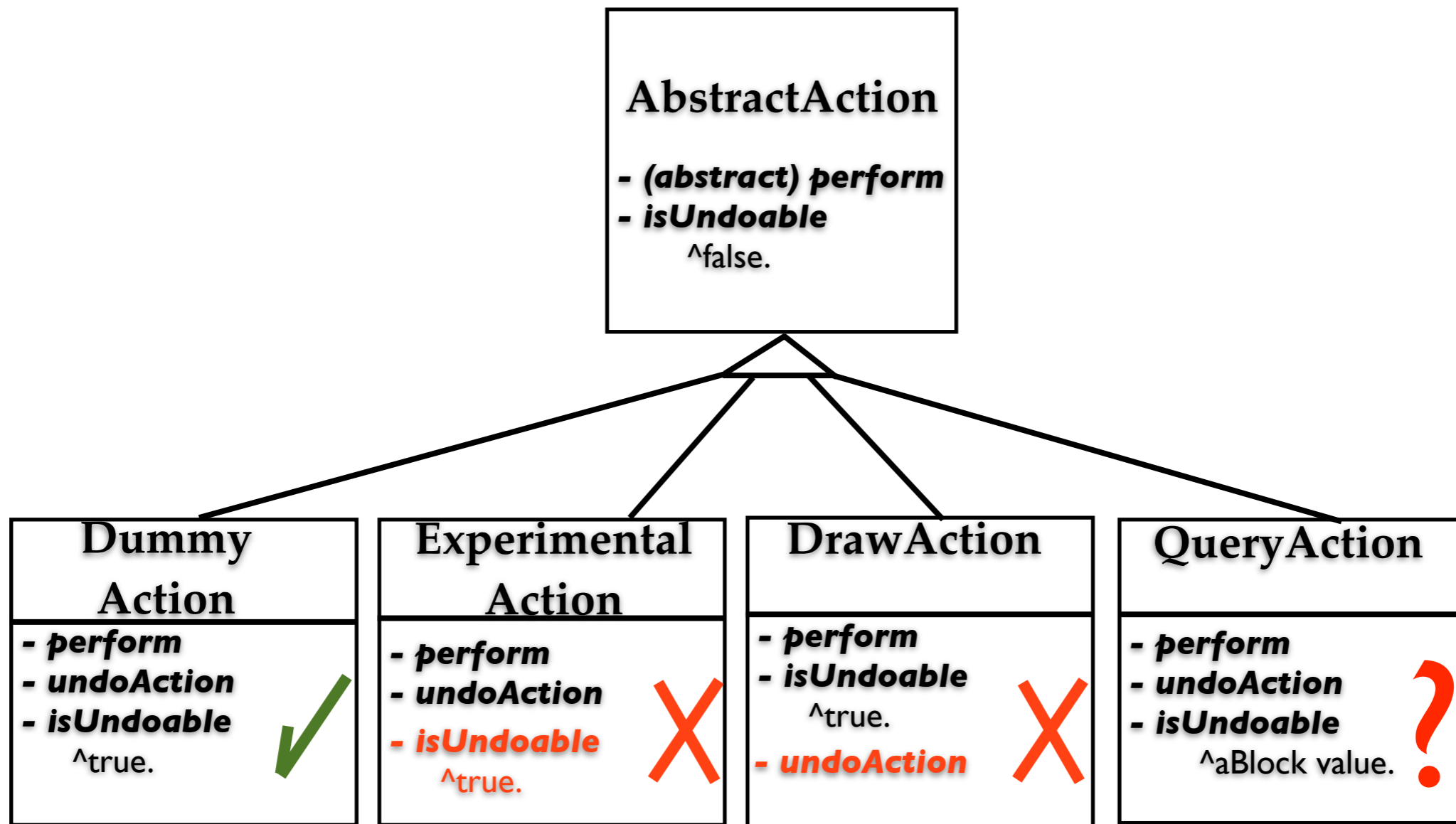
An inconsistency example

The command design pattern

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*



An inconsistency example

The command design pattern

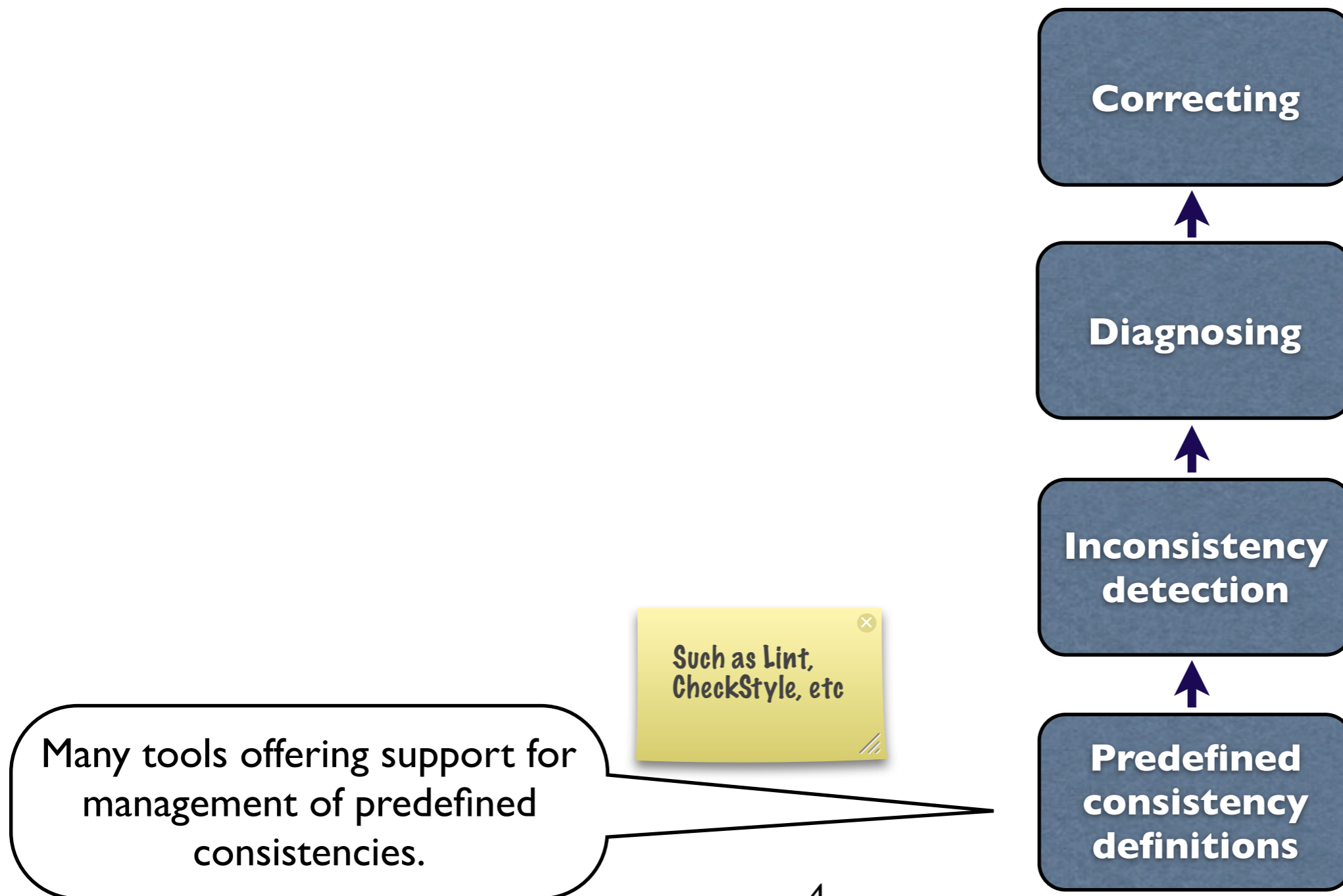
Consistency property:

∀ Action, an **undoAction** method should be provided

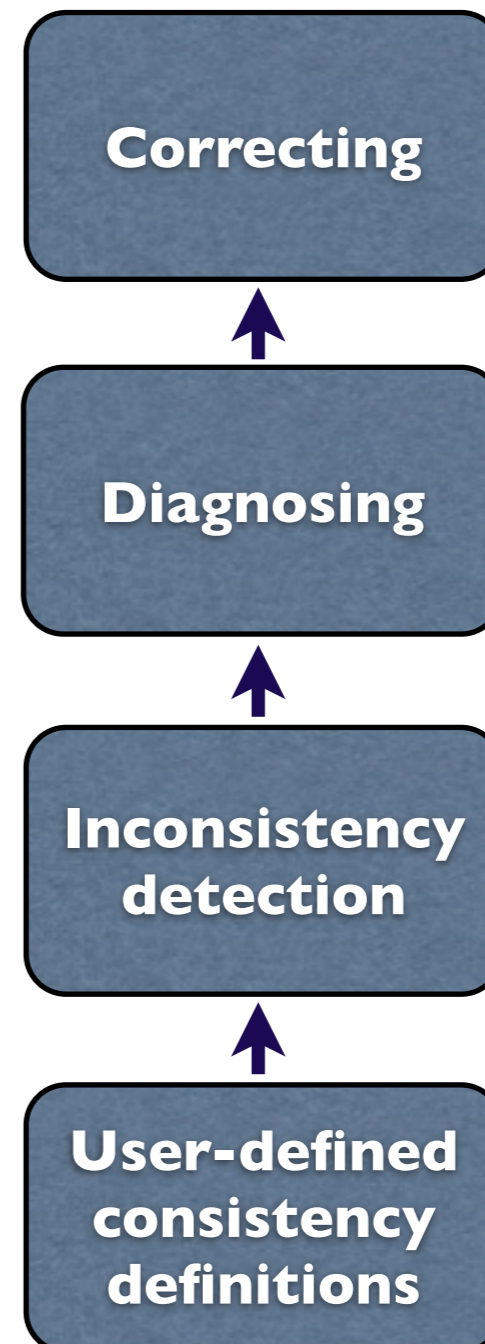
⇔ the **isUndoable** method returns *true*

User-defined
consistency

Inconsistency management activities

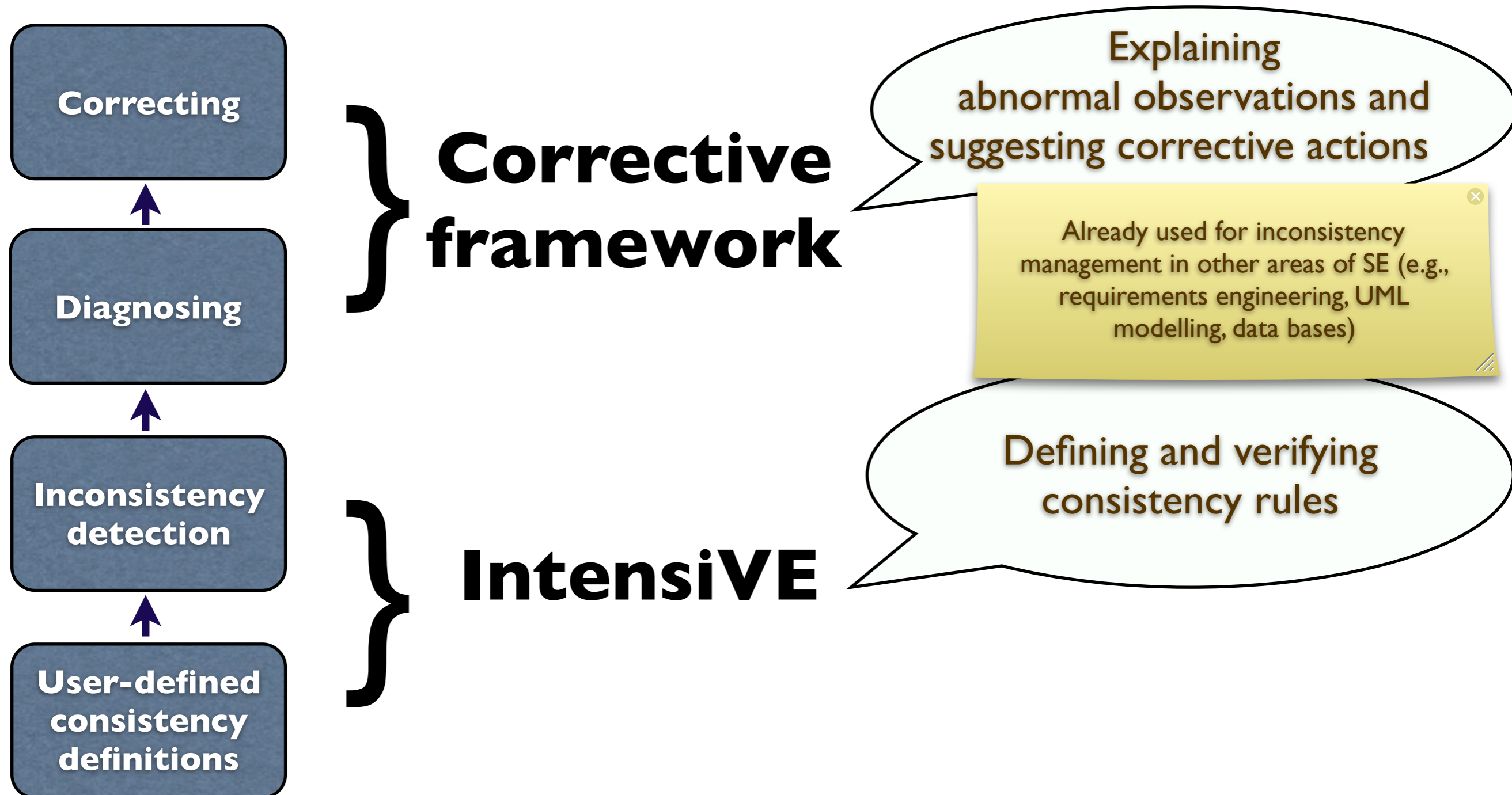


Inconsistency management activities

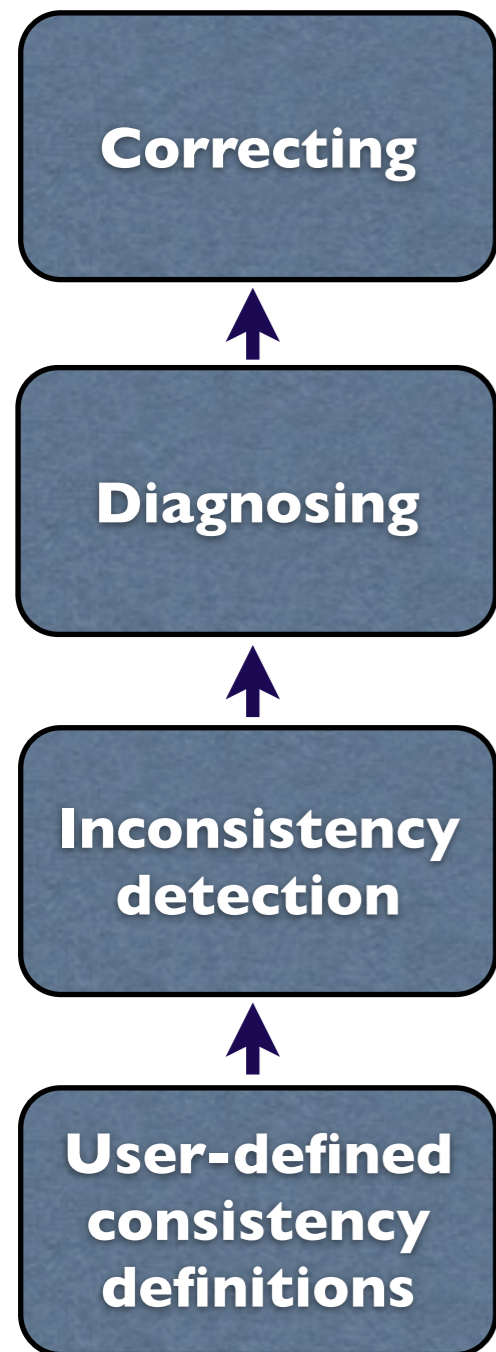


We propose a general approach for the management of user-defined consistencies.

Inconsistency management activities



Inconsistency management activities

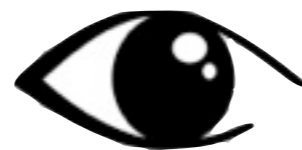


Corrective framework

Explaining abnormal observations and suggesting corrective actions

Already used for inconsistency management in other areas of SE (e.g., requirements engineering, UML modelling, data bases)

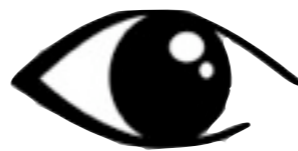
IntensiVE



(<http://www.intensive.be>)

Defining and verifying consistency rules

IntensiVE



(<http://www.intensive.be>)

**Inconsistency
detection**



**User-defined
consistency
definitions**

Defining and verifying constraints using IntensiVE

- Many kind of checks.
- Defined as **relationships among sets** of source code elements.
- Sets are **intensionally defined** and referred as **Intensional Views**.



with queries
over source code

Verification of the *Undoable Actions* constraint

Consistency property:

∀ Action, an *undoAction* method should be provided

↔ the *isUndoable* method returns *true*

**View with
classes returning
true in the
isUndoable
method**

**View with
classes
implementing the
undoAction
method**

Verification of the *Undoable Actions* constraint

QUERY:

classChainReallyUnderstandsMethodWithName(?class,?method,isUndoable),
methodReturnsBoolean(?method,[true])

Consistency property:

∀ Action, an *undoAction* method should be provided

↔ the *isUndoable* method returns *true*

**View with
classes returning
true in the
isUndoable
method**

**View with
classes
implementing the
undoAction
method**

Verification of the *Undoable Actions*

constraint

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?method,isUndoable),
methodReturnsBoolean(?method,[true])

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?,undoAction)

Consistency property:

∀ Action, an *undoAction* method should be provided

⇔ the *isUndoable* method returns *true*

**View with
classes returning
true in the
isUndoable
method**

**View with
classes
implementing the
undoAction
method**

Verification of the *Undoable Actions* constraint

QUERY:

classChainReallyUnderstandsMethodWithName(?class,?method,isUndoable),
methodReturnsBoolean(?method,[true])

QUERY:

classChainReallyUnderstandsMethodWithName(?class,?,undoAction)

Consistency property:

\forall Action, an ***undoAction*** method should be provided
 \Leftrightarrow the ***isUndoable*** method returns *true*

View with classes returning *true* in the *isUndoable* method

View with classes implementing the *undoAction* method

If consistent, any class in the first view should be also in the second, and vice versa

Verification of alternative views

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?method,isUndoable),
methodReturnsBoolean(?method,[true])

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?,undoAction)

Consistency property:

∀ Action, an **undoAction** method should be provided

⇔ the **isUndoable** method returns *true*

**View with classes
returning *true* in the
isUndoable method**

AbstractAddAction

AddClassificationAction

ExperimentalAction

**View with classes
implementing the
undoAction method**

AbstractAddAction

AddClassificationAction

Verification of alternative views

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?method,isUndoable),
methodReturnsBoolean(?method,[true])

QUERY:

classChainReallyUnderstandsMethodWith
Name(?class,?,undoAction)

Consistency property:

\forall Action, an **undoAction** method should be provided

\Leftrightarrow the **isUndoable** method returns *true*

**View with classes
returning *true* in the
isUndoable method**

AbstractAddAction

AddClassificationAction

ExperimentalAction

**View with classes
implementing the
undoAction method**

AbstractAddAction

AddClassificationAction

?

IntensiVE

(consistency checking)

View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Table View Text Report

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)

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

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

Classes that implement the *undoAction* method

The screenshot shows a window with a table of classes. The table has three columns: 'Tuples', '1(250 ms)', and '2(238 ms)'. The 'Tuples' column lists 15 classes. The '1' and '2' columns contain green or red circles indicating compliance. A red bar at the bottom indicates 'INCONSISTENT! (8/29)'. A 'Full Extension' checkbox is checked.

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 -> AddClassAction	●	●
class -> AddObjectAction	●	●
class -> AddSmartAction	●	●
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)**

This class does not return *true* at the *isUndoable* method

But implements an *undoAction* method

IntensiVE

(consistency checking)

ExperimentalAction



This class does not return *true* at the *isUndoable* method

But implements an *undoAction* method

View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Table View Text Report

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	●	●
class -> AddClassAction	●	●
class -> AddObjectAction	●	●
class -> AddSmartAction	●	●
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

(diagnosing inconsistencies)

ExperimentalAction



View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Table View Text Report

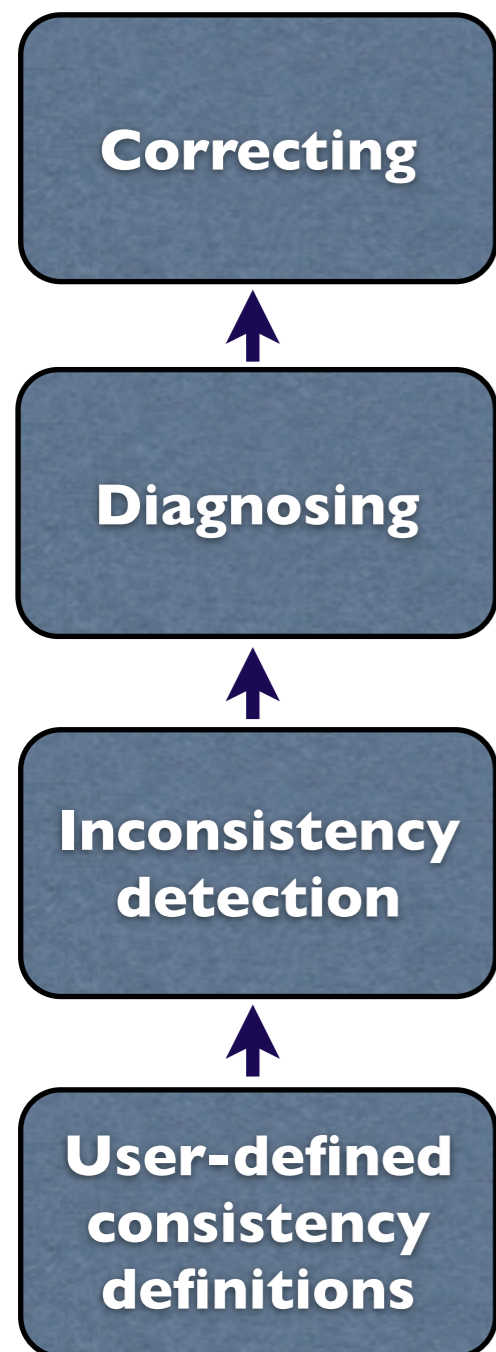
Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	●	●
class -> AddClassificationAction	●	●
class -> AddObjectAction	●	●
class -> AddSmartClassificationAction	●	●
class -> ClearClassificationAction	●	●
class -> ExperimentalAction	●	●
class -> RemoveAction	●	●
class -> Rename	●	●
class -> Test	●	●
class -> Add	●	●
class -> Add	●	●
class -> AddIVV	●	●
class -> AddRegularity	●	●
class -> AddRelationAction	●	●

Full Extension **INCONSISTENT! (8/29)**

Why **ExperimentalAction** is not a result of the query defining this set?

OR: Why **ExperimentalAction** is a result of the query defining this set?

Inconsistency management activities



Corrective framework

Explaining abnormal observations and suggesting corrective actions

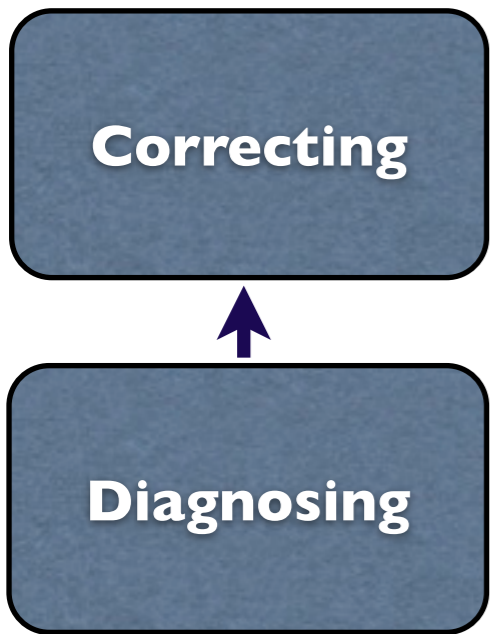
Already used for inconsistency management in other areas of SE (e.g., requirements engineering, UML modelling, data bases)

IntensiVE



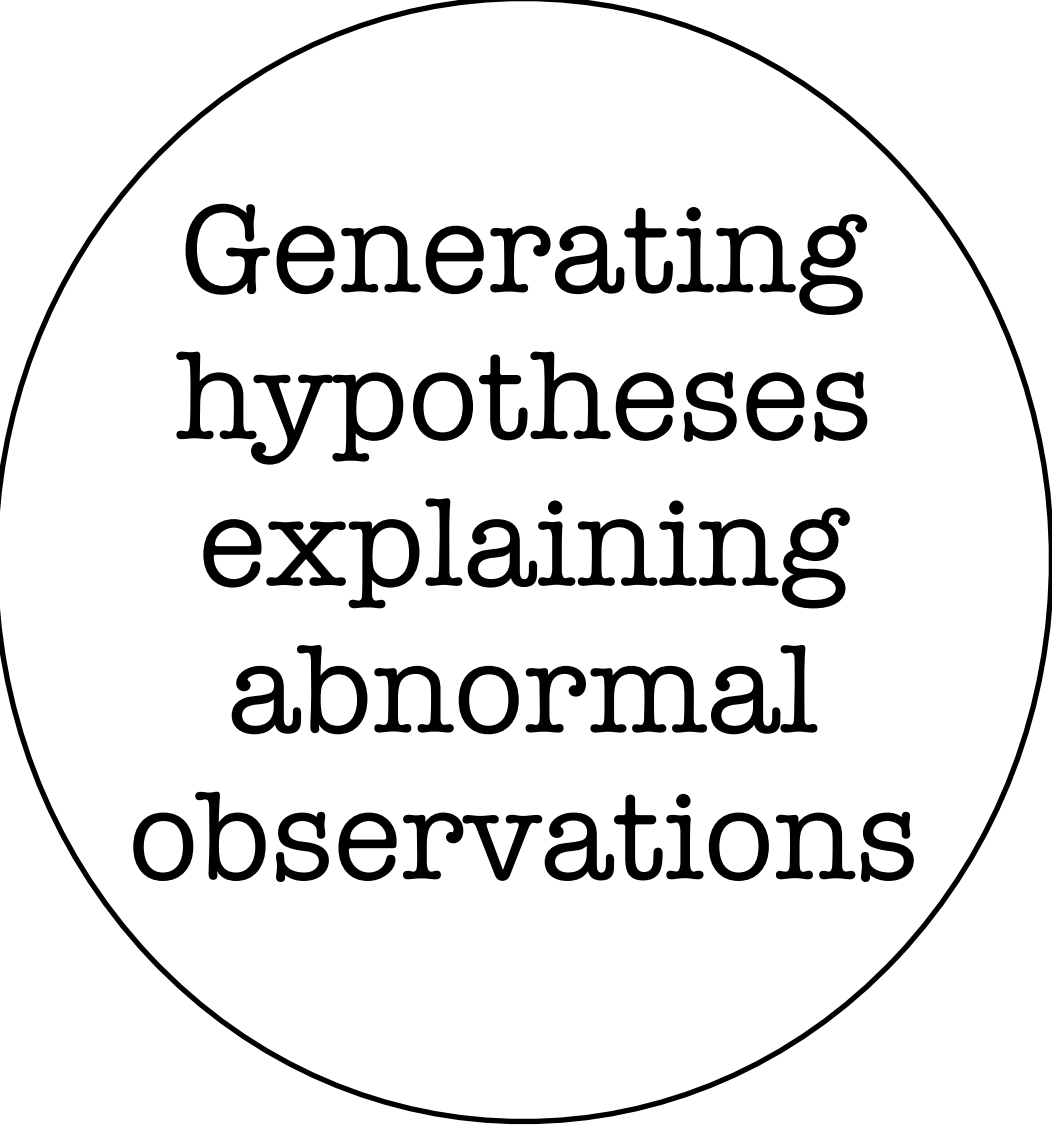
(<http://www.intensive.be>)

Defining and verifying consistency rules



Corrective framework

What is diagnosis?



Generating
hypotheses
explaining
abnormal
observations

**Abductive
reasoning**

What is diagnosis?

Generating
hypotheses
explaining
abnormal
observations



**Abductive
reasoning**

What is diagnosis?

Generating hypotheses explaining abnormal observations



Abductive reasoning

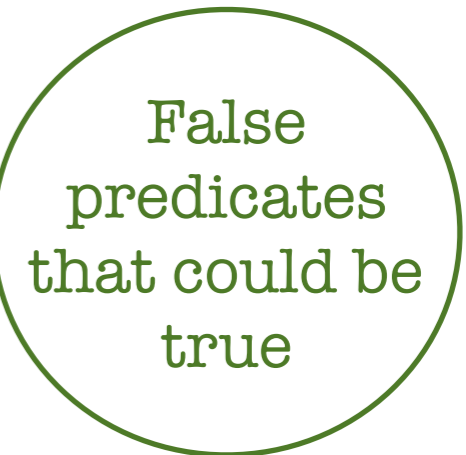
one of the fundamental forms of human reasoning according to *Pierce*

Abduction is suitable for

- Generating hypotheses that would **explain** an evidence.
- Explanations expressed in terms of certain predicates, declared before hand as ***abducibles***.

Abduction is suitable for

- Generating hypotheses that would **explain** an evidence.
- Explanations expressed in terms of certain predicates, declared before hand as **abducibles**.



Abduction is suitable for

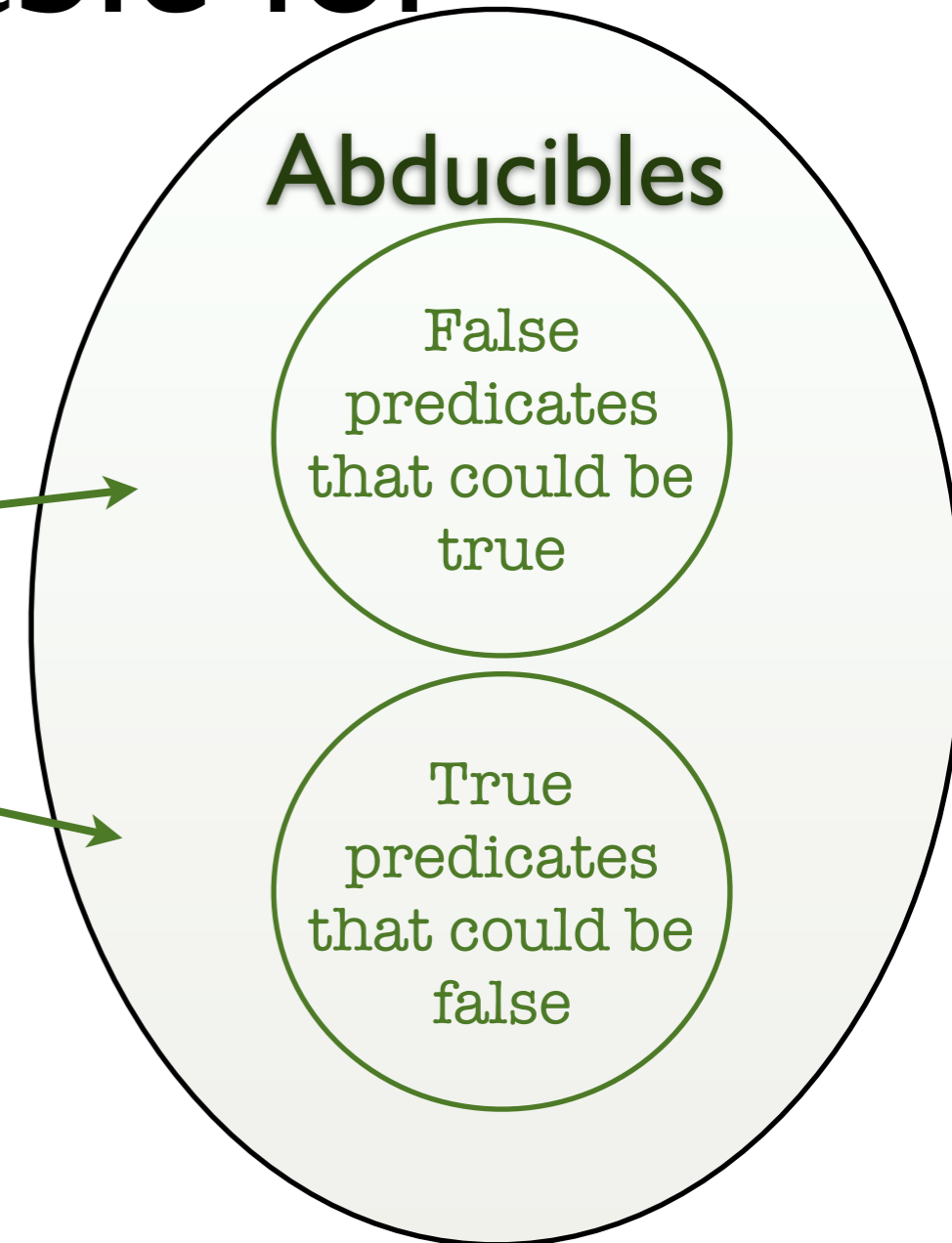
- Generating hypotheses that would **explain** an evidence.
- Explanations expressed in terms of certain predicates, declared before hand as **abducibles**.

False predicates that could be true

True predicates that could be false

Abduction is suitable for

- Generating hypotheses that would **explain** an evidence.
- Explanations expressed in terms of certain predicates, declared before hand as **abducibles**.



We can think in
“**Abducibles**” as
“**Correctables**”

An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)

Abducibles

bird(?x).

broken-wing(?x).

An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)

Abducibles

bird(?x).

broken-wing(?x).

Observation:

flies(opus)

An example of abduction

Theory:

flies(?x) if **bird(?x)**, not (ab(?x))

ab(?x) if broken-wing(?x)

failing

Abducibles

bird(?x).

broken-wing(?x).

Observation:

flies(opus)

An example of abduction

Theory:

flies(?x) if **bird(?x)**, not (ab(?x))

ab(?x) if broken-wing(?x)

bird(opus)

failing

added to
our theory

Abducibles

bird(?x).

broken-wing(?x).

Observation:

flies(opus)

An example of abduction

Theory:

flies(?x) if **bird(?x)**, not (ab(?x))

ab(?x) if broken-wing(?x)

bird(opus)

Observation:

flies(opus)

succeeding

Abducibles

bird(?x).

broken-wing(?x).

added to
our theory

An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)



bird(tweety)

broken-wing(tweety)

Abducibles

bird(?x).

broken-wing(?x).

An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)

bird(tweety)

broken-wing(tweety)



Observation:

flies(tweety)



Abducibles

bird(?x).

broken-wing(?x).

An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)

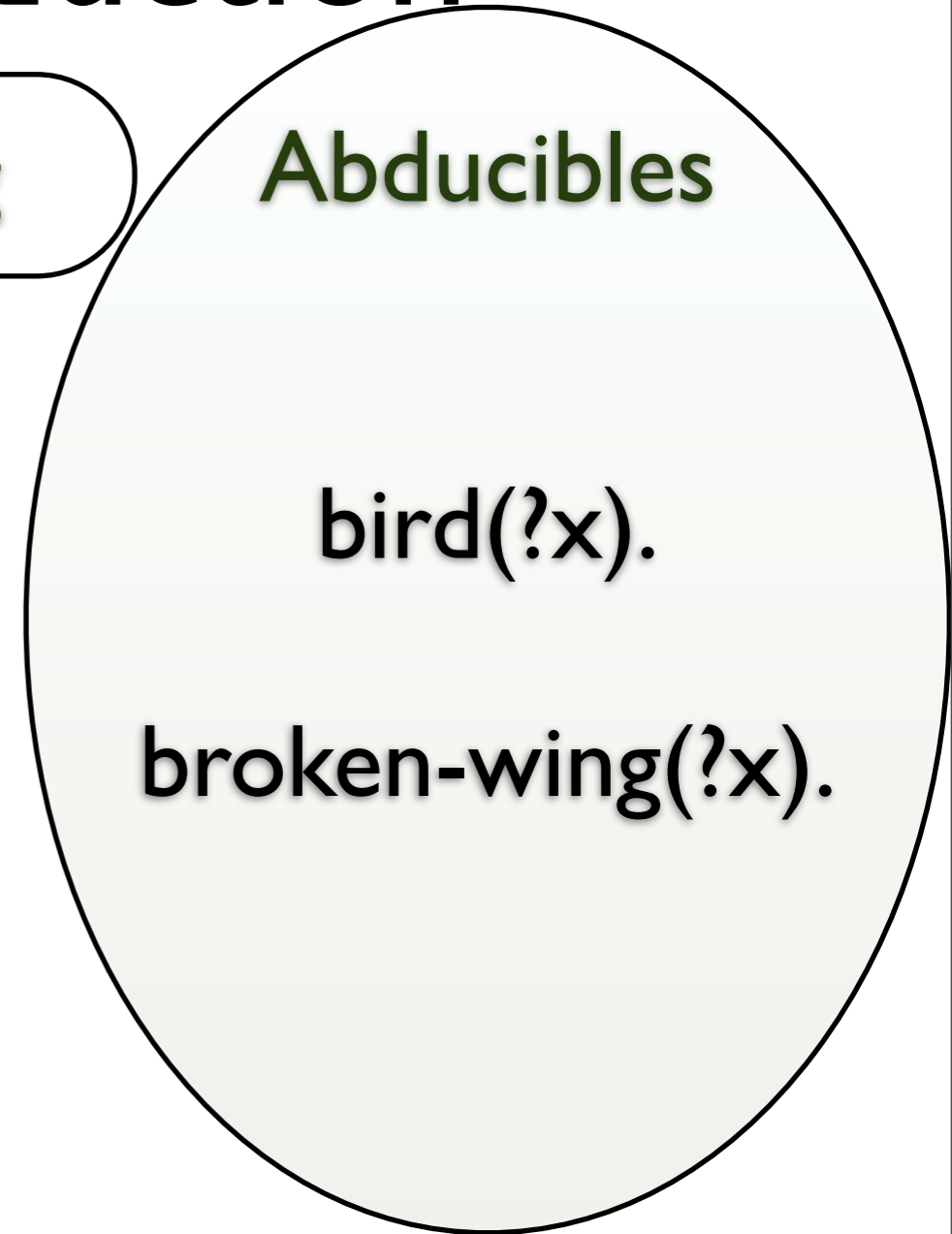
bird(tweety)

broken-wing(tweety)

Observation:

flies(tweety)

failing



An example of abduction

Theory:

flies(?x) if bird(?x), **not (ab(?x))**

ab(?x) if broken-wing(?x)

bird(tweety)

~~broken-wing(tweety)~~

Observation:

flies(tweety)

failing

Abducibles

bird(?x).

broken-wing(?x).

retracted from
our theory



An example of abduction

Theory:

flies(?x) if bird(?x), not (ab(?x))

ab(?x) if broken-wing(?x)

bird(tweety)

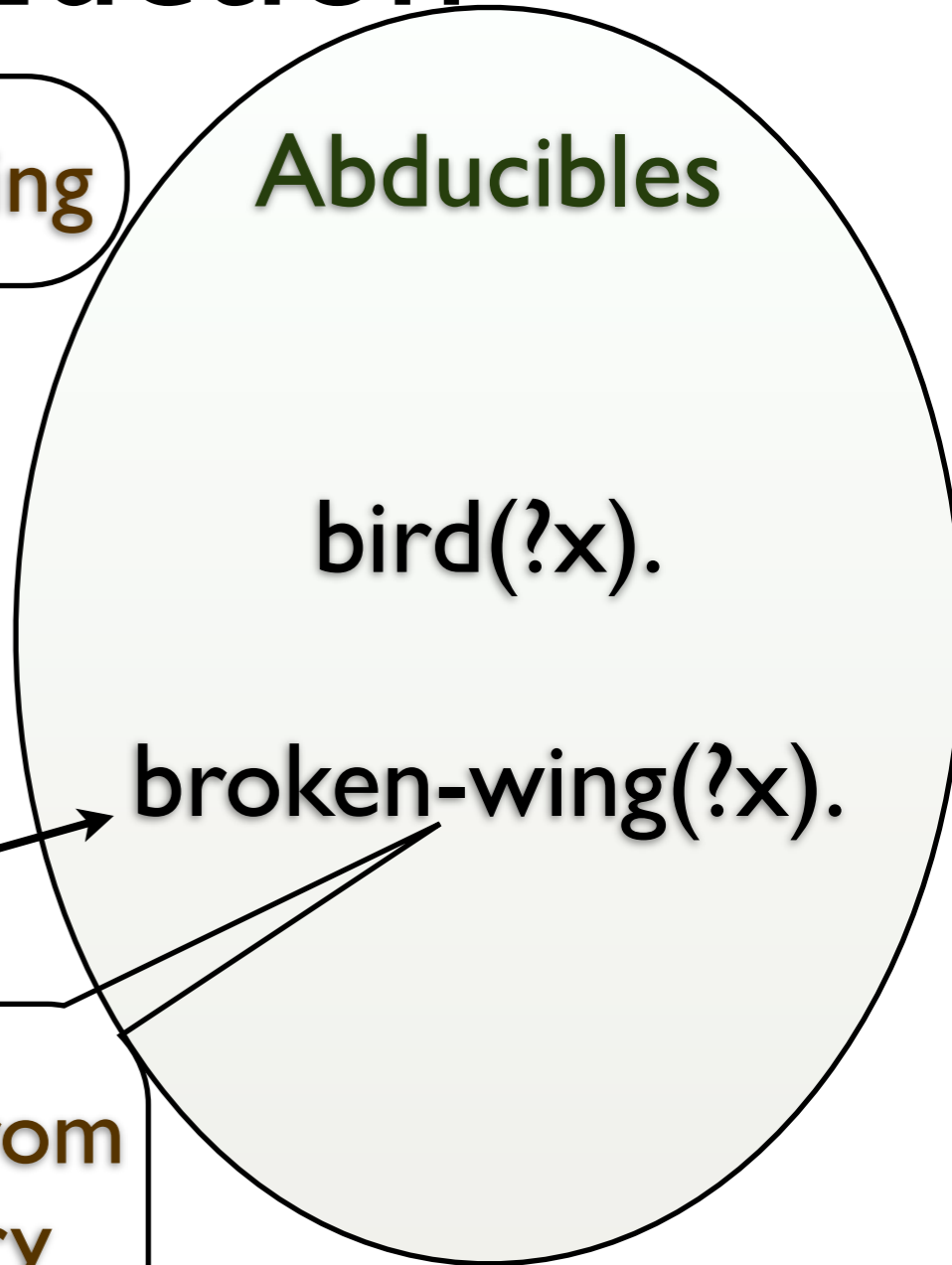
~~broken-wing(tweety)~~

Observation:

flies(tweety)

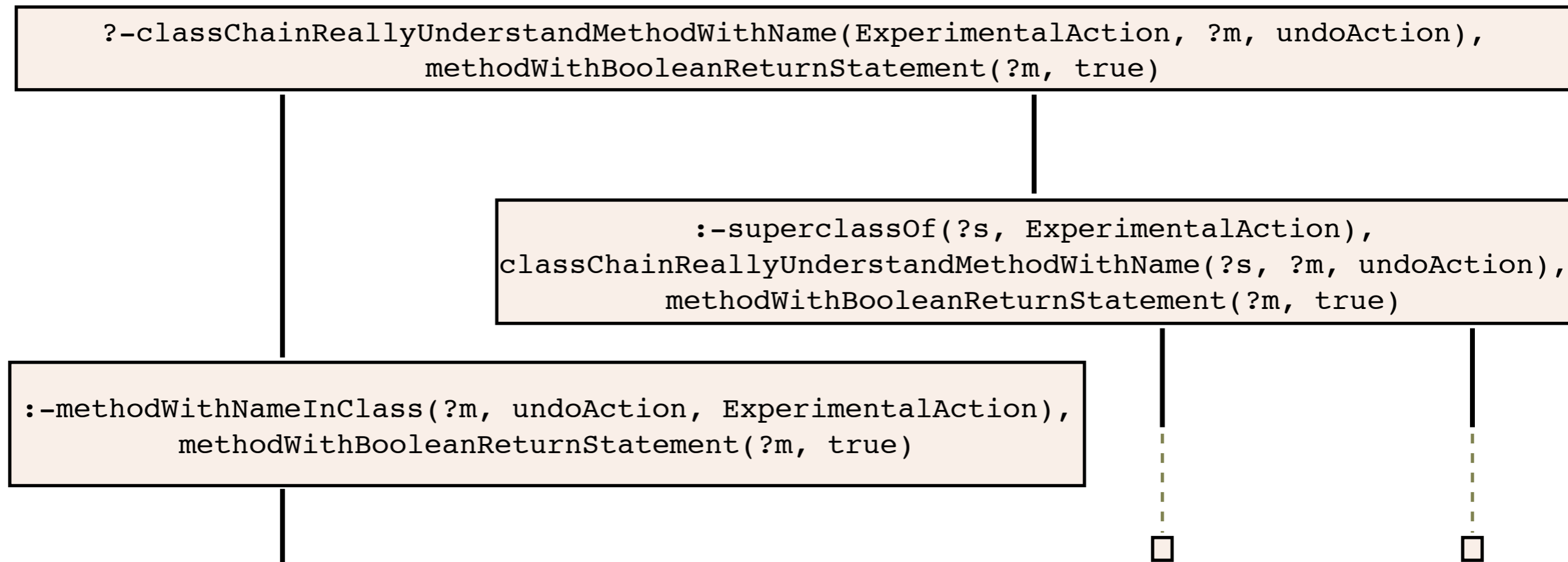
succeeding

not (ab(?x))



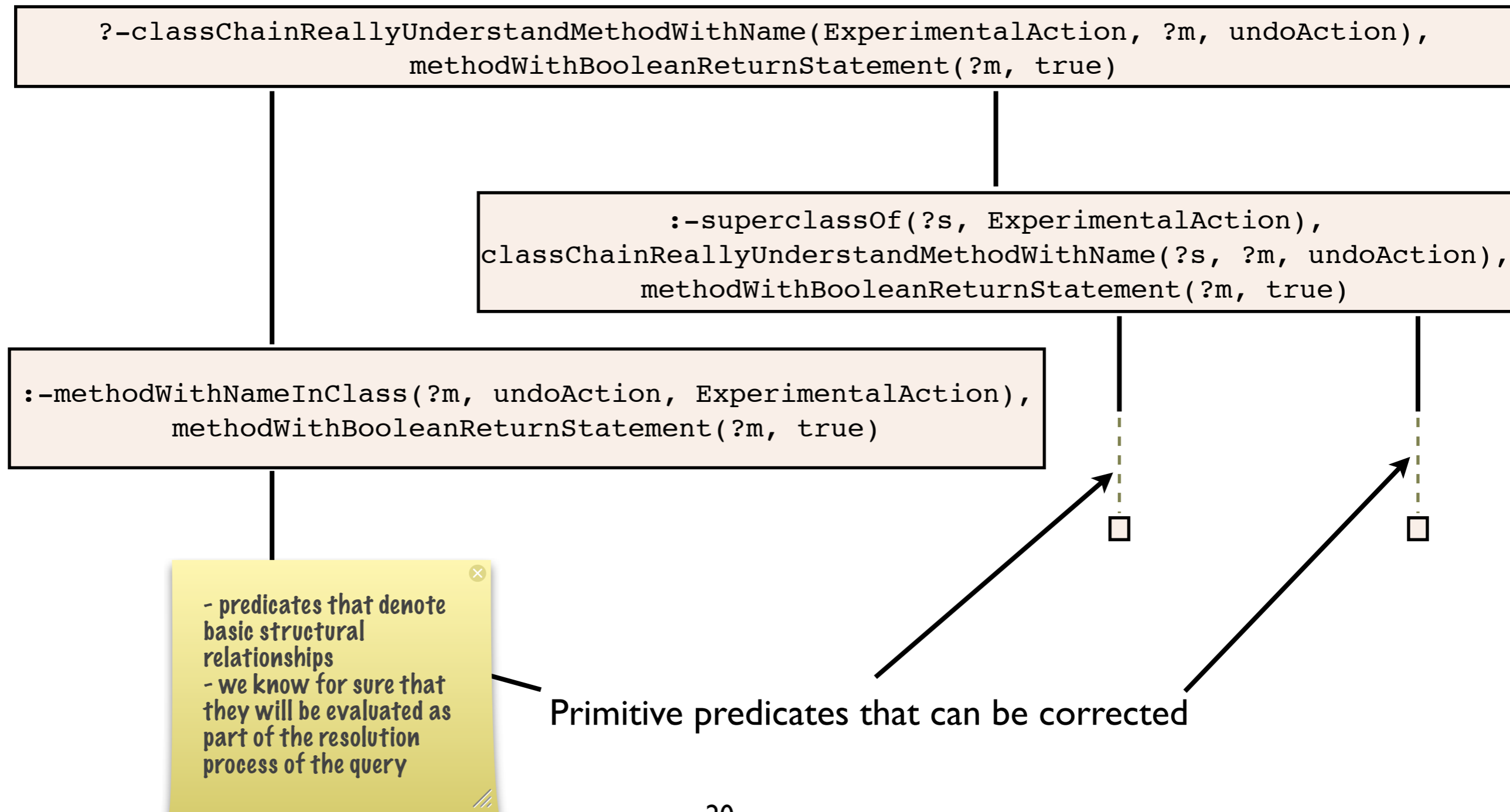
retracted from our theory

How to choose abducible predicates

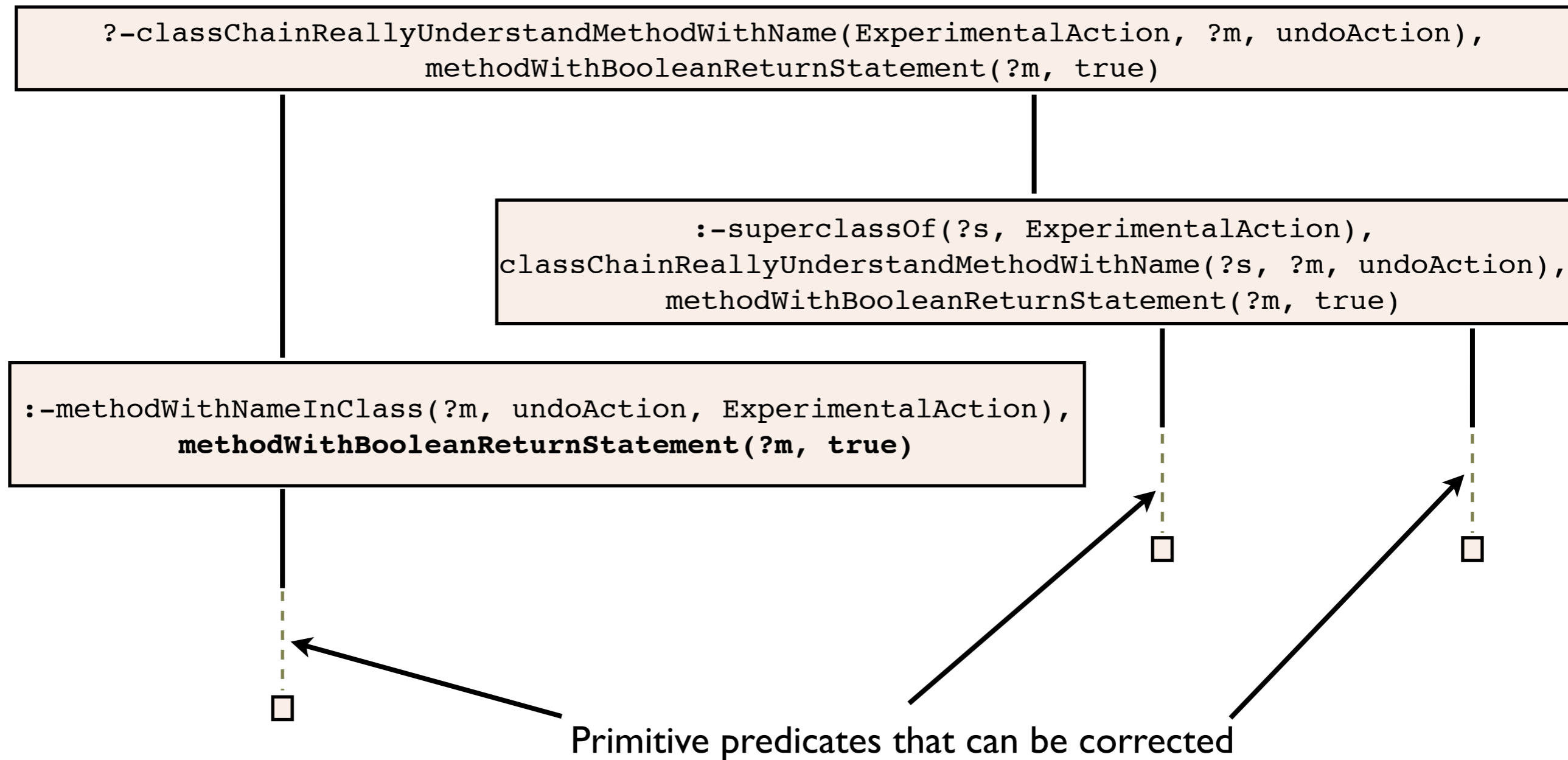


- predicates that denote basic structural relationships
- we know for sure that they will be evaluated as part of the resolution process of the query

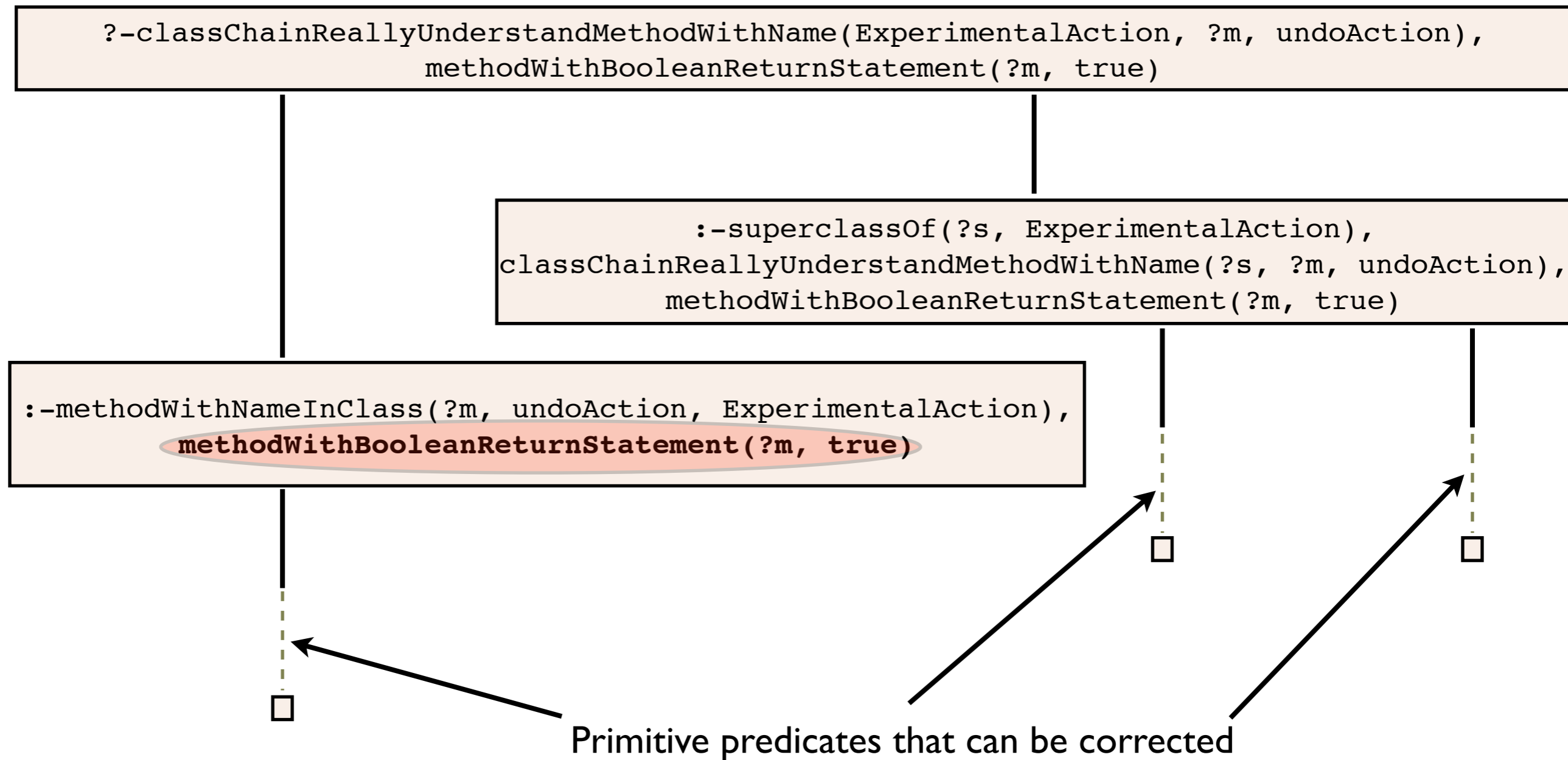
How to choose abducible predicates



How to choose abducible predicates



How to choose abducible predicates



Our framework allows ...

- The definition of abducible predicates.
- Declaration of multiple *corrective actions*.
- Generation of hypotheses explaining inconsistencies.
- Semi-automatic execution of corrective actions.

Defining *positive* explanations

(new facts will be added to the theory)

The screenshot shows the SOUL Clause Browser interface. At the top, there are window controls and the title "SOUL Clause Browser". Below that is a menu bar with "Tools", "Special", and "Help". A "Lookup:" dropdown menu is set to "default". On the left, a tree view shows a hierarchy of libraries: "CodeCorrectionLibrary" (selected), "SmalltalkReasoning", "JavaEclipseReasoning", and "IntensionalViewsLayer". The main area is divided into three panes: "methodManipulation", "correct/4", and a list of clauses. The "correct/4" pane is active, showing a list of clauses. The bottom pane displays a detailed view of a clause:

```
correct(in(methodWithBooleanReturnStatement(+?method,+?boolean)),  
[ 'this solution changes the code of the method ', ?method asString, '. The new code just has a simple ', ?boolean asString, ' return statement'],  
[[?class compile: (?selector asString, '^', ?boolean asString).]]) if  
  methodWithNameInClass(?method,?selector,?class)
```

Defining *positive* explanations

(new facts will be added to the theory)

The screenshot shows the SOUL Clause Browser interface. The top bar contains the title "SOUL Clause Browser" and menu items "Tools", "Special", and "Help". Below the menu is a "Lookup:" field set to "default". A tree view on the left shows a hierarchy of libraries: "CodeCorrectionLibrary", "SmalltalkReasoning", "JavaEclipseReasoning", and "IntensionalViewsLayer". The main area is divided into three panes: "methodManipulation", "correct/4", and a list of clauses. The "correct/4" pane is selected, showing a list of clauses. The bottom pane shows a detailed view of a clause: `correct(in(methodWithBooleanReturnStatement(+?method,+?boolean)),` followed by a description: `['this solution changes the code of the method ', ?method asString, '. The code just has a simple ', ?boolean asString, ' return statement'],` and a condition: `[[?class compile: (?selector asString, '^', ?boolean asString).]] if` followed by `methodWithNameInClass(?method,?selector,?class)`. A magnifying glass icon is positioned over the `in` predicate in the clause. A speech bubble points to the `in` predicate with the text: **Abducible predicate:** `in(methodWithBooleanReturnStatement(+?method,+?boolean))`.

Defining *positive* explanations

(new facts will be added to the theory)

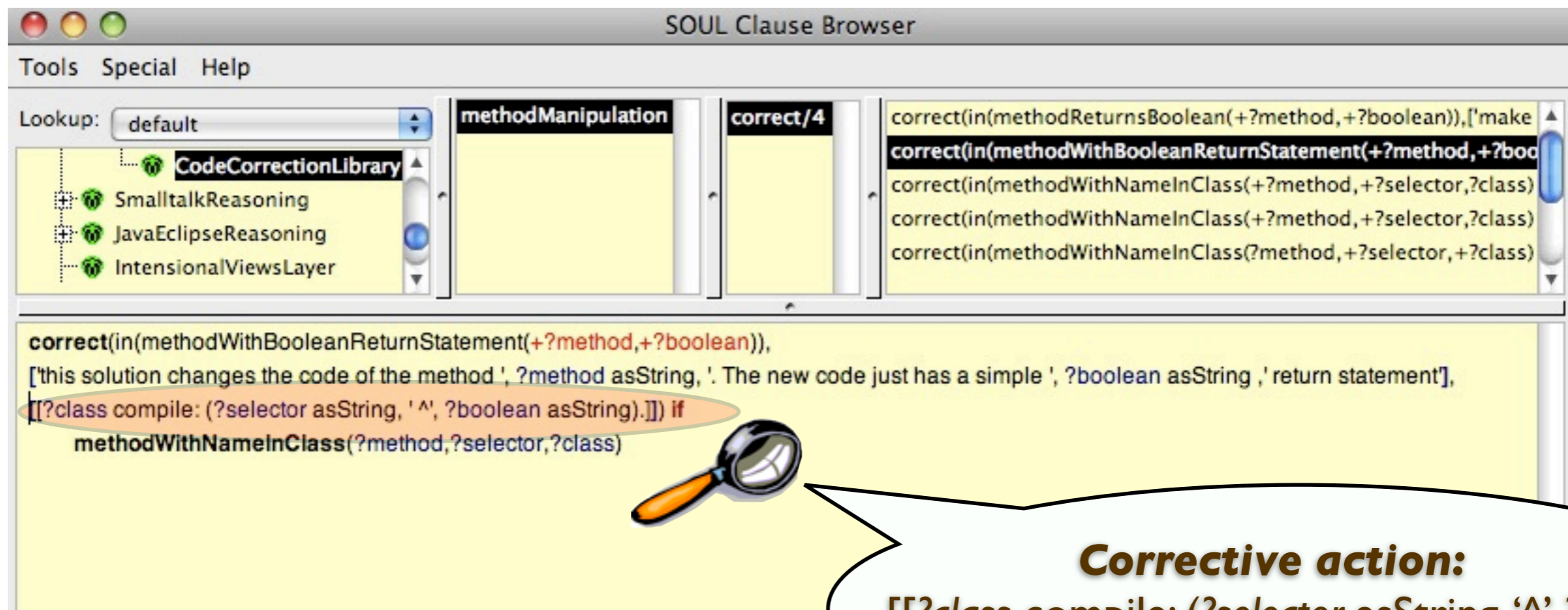
The screenshot shows the SOUL Clause Browser interface. At the top, there are window control buttons and the title "SOUL Clause Browser". Below the title bar is a menu bar with "Tools", "Special", and "Help".

The main interface is divided into several panes:

- Lookup:** A dropdown menu set to "default".
- Tree View:** A hierarchical view of the knowledge base. The root is "CodeCorrectionLibrary", which contains sub-nodes for "SmalltalkReasoning", "JavaEclipseReasoning", and "IntensionalViewsLayer".
- Method Manipulation:** A pane with a yellow background, currently empty.
- correct/4:** A pane with a yellow background, currently empty.
- Clause List:** A list of clauses. The first clause is highlighted: `correct(in(methodReturnsBoolean(+?method, +?boolean)), ['make`. Other clauses include `correct(in(methodWithBooleanReturnStatement(+?method, +?boolean),`, `correct(in(methodWithNameInClass(+?method, +?selector, ?class)`, `correct(in(methodWithNameInClass(+?method, +?selector, ?class)`, and `correct(in(methodWithNameInClass(?method, +?selector, +?class)`.
- Clause Detail:** A large pane with a yellow background showing the selected clause: `correct(in(methodWithBooleanReturnStatement(+?method, +?boolean)),`. Below the clause is a list of conditions: `['this solution changes the code of the method ', ?method asString, '. The new code just has a simple ', ?boolean asString, ' return statement'],` and `[[?class compile: (?selector asString, '^', ?boolean asString).]]`. The clause is followed by the keyword `if` and the clause `methodWithNameInClass(?method, ?selector, ?class)`.

Defining *positive* explanations

(new facts will be added to the theory)



The screenshot shows the SOUL Clause Browser interface. The top menu includes 'Tools', 'Special', and 'Help'. A 'Lookup:' dropdown is set to 'default'. The left sidebar shows a tree view with 'CodeCorrectionLibrary' selected, containing sub-items: 'SmalltalkReasoning', 'JavaEclipseReasoning', and 'IntensionalViewsLayer'. The main area is divided into three panes: 'methodManipulation', 'correct/4', and a code editor. The code editor displays a list of clauses, with the first one highlighted: `correct(in(methodReturnsBoolean(+?method,+?boolean)),['make`. Below this, a larger pane shows a detailed view of a clause: `correct(in(methodWithBooleanReturnStatement(+?method,+?boolean)),` followed by a comment: `['this solution changes the code of the method ', ?method asString, '. The new code just has a simple ', ?boolean asString, ' return statement'],` and a corrective action: `[[?class compile: (?selector asString, '^', ?boolean asString).]] if` and `methodWithNameInClass(?method,?selector,?class)`. A magnifying glass icon is positioned over the corrective action part of the code.

Corrective action:
[[?class compile: (?selector asString, '^', ?boolean asString)]]

Correcting inconsistencies

The screenshot shows a window titled "View Consistency" with two tabs: "Table View" and "Text Report". The window displays a table comparing two views, "1(4587 ms)" and "2(4670 ms)". The table lists various classes and their consistency status, indicated by red (inconsistent) or green (consistent) circles. A context menu is open over the "Correct" button for the first row.

Tuples	1(4587 ms)	2(4670 ms)
class -> AbstractQuantifier	●	●
class -> ExceptionElement	●	●
class -> Extension	●	●
class -> Intension	●	●
class -> IntensionalRootProject	●	●
class -> IVEntityDefinition	●	●
class -> IVGroup	●	●
class -> IVRelationDef	●	●
class -> ProjectChildClassification	●	●
class -> RegularityInstancesClassification	●	●
class -> ResultPair	●	●
class -> Template	●	●
class -> Tuple	●	●
class -> TupleVariable	●	●

Context menu options: Browse, Inspect, Sort, Document as Ex, Remove exceptic, Explain, Correct.

Full Extension **INCONSISTENT! (5/15)**

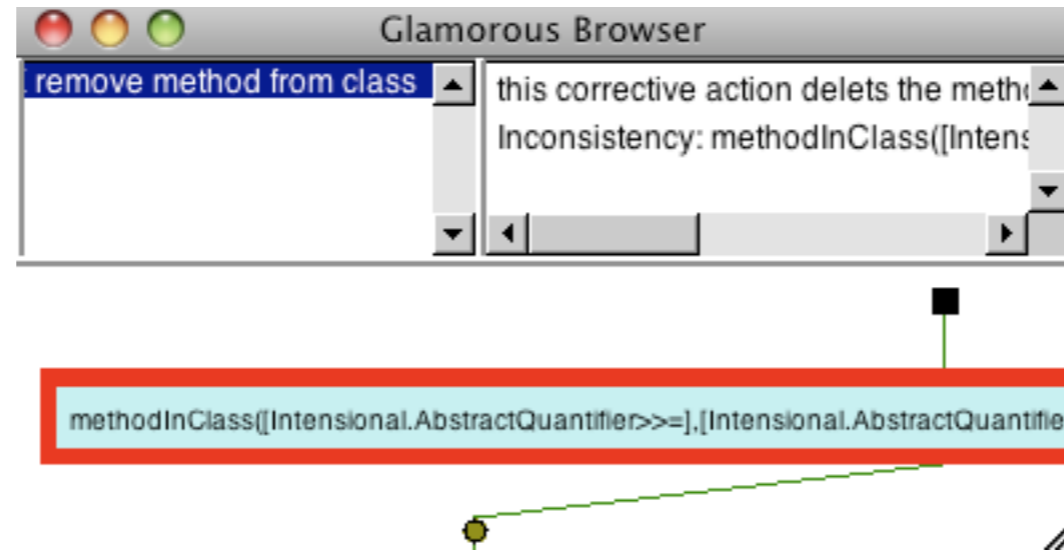
Correcting inconsistencies

The screenshot shows a window titled "View Consistency" with two tabs: "Table View" and "Text Report". The window displays a table with the following columns: "Tuples", "1(4587 ms)", and "2(4670 ms)". The table lists various class comparisons, each with a status indicator (red or green dot). A context menu is open over the table, with the "Correct" option highlighted. A magnifying glass icon is positioned over the "Correct" button, and a speech bubble points to it with the word "Correct".

Tuples	1(4587 ms)	2(4670 ms)
class -> AbstractQuantifier	●	●
class -> ExceptionElement	●	●
class -> Extension	●	●
class -> Intension	●	●
class -> IntensionalRootProject	●	●
class -> IVEntityDefinition	●	●
class -> IVGroup	●	●
class -> IVRelationDef	●	●
class -> ProjectChildClassification	●	●
class -> RegularityInstancesClassification	●	●
class -> ResultPair	●	●
class -> Template	●	●
class -> Tuple	●	●
class -> TupleVariable	●	●

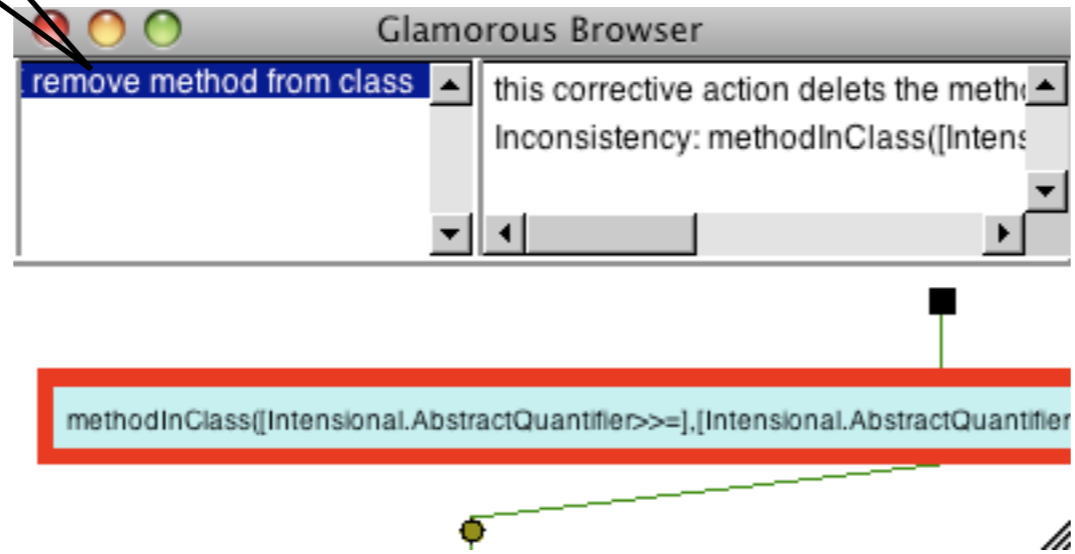
Full Extension **INCONSISTENT! (5/15)**

The corrective browser

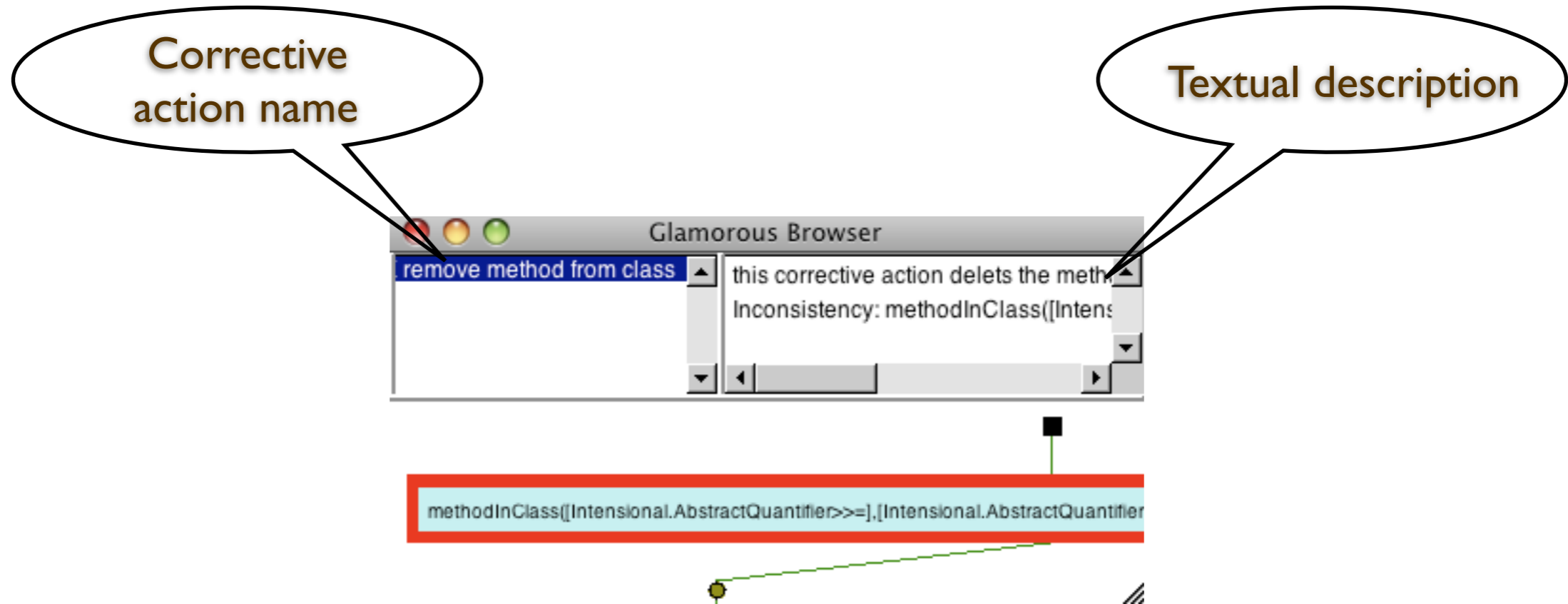


The corrective browser

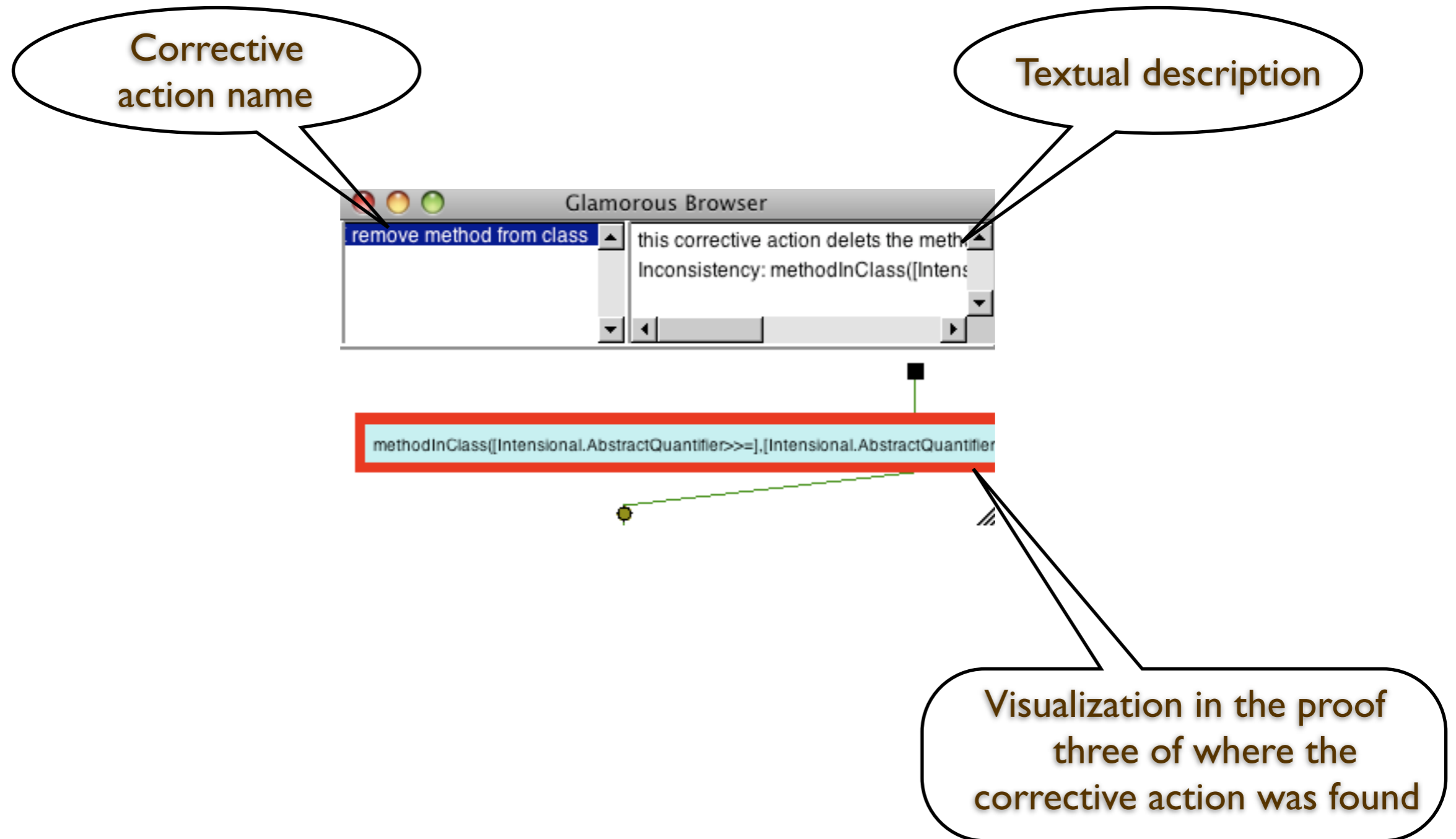
Corrective
action name



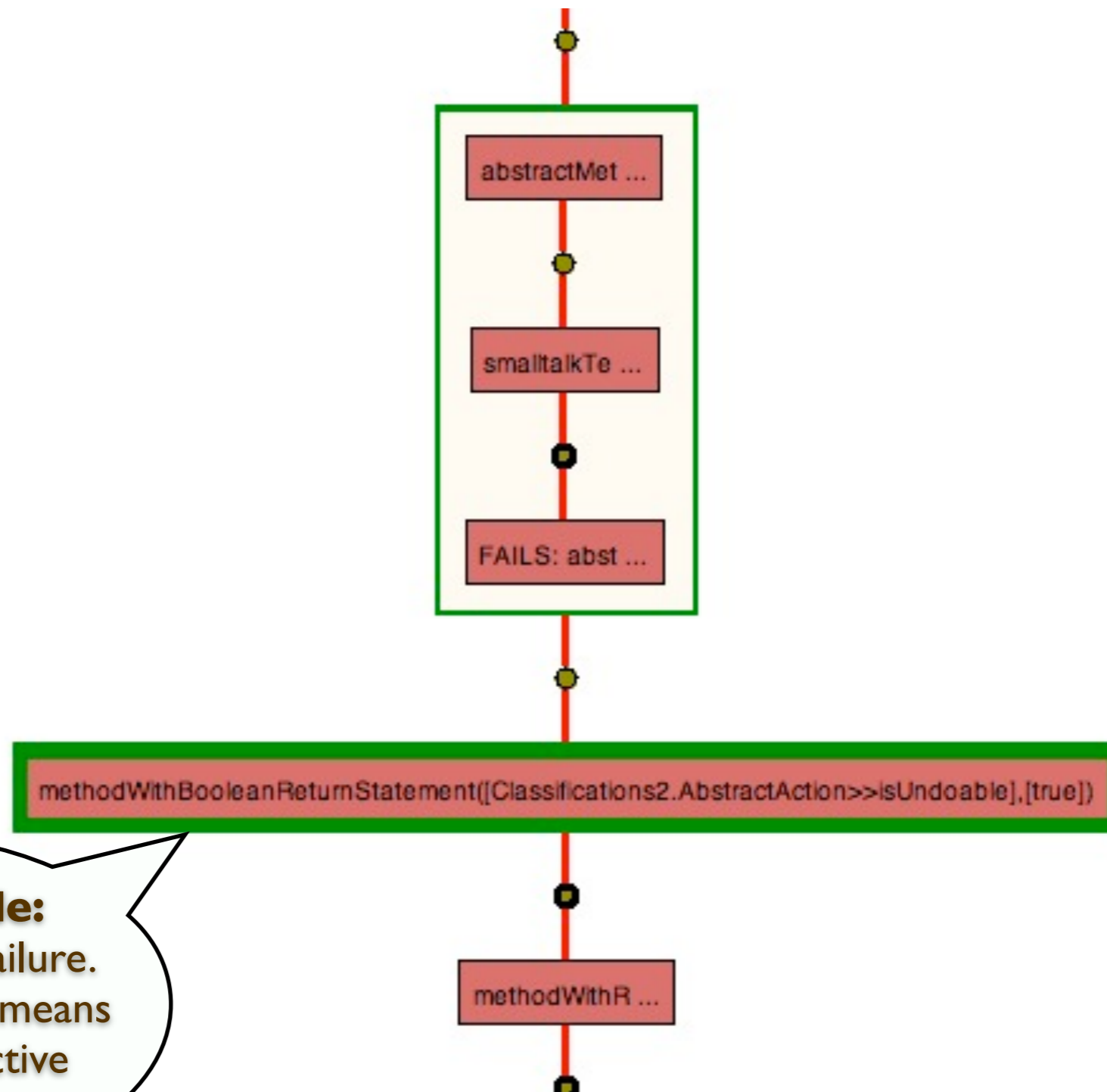
The corrective browser



The corrective browser



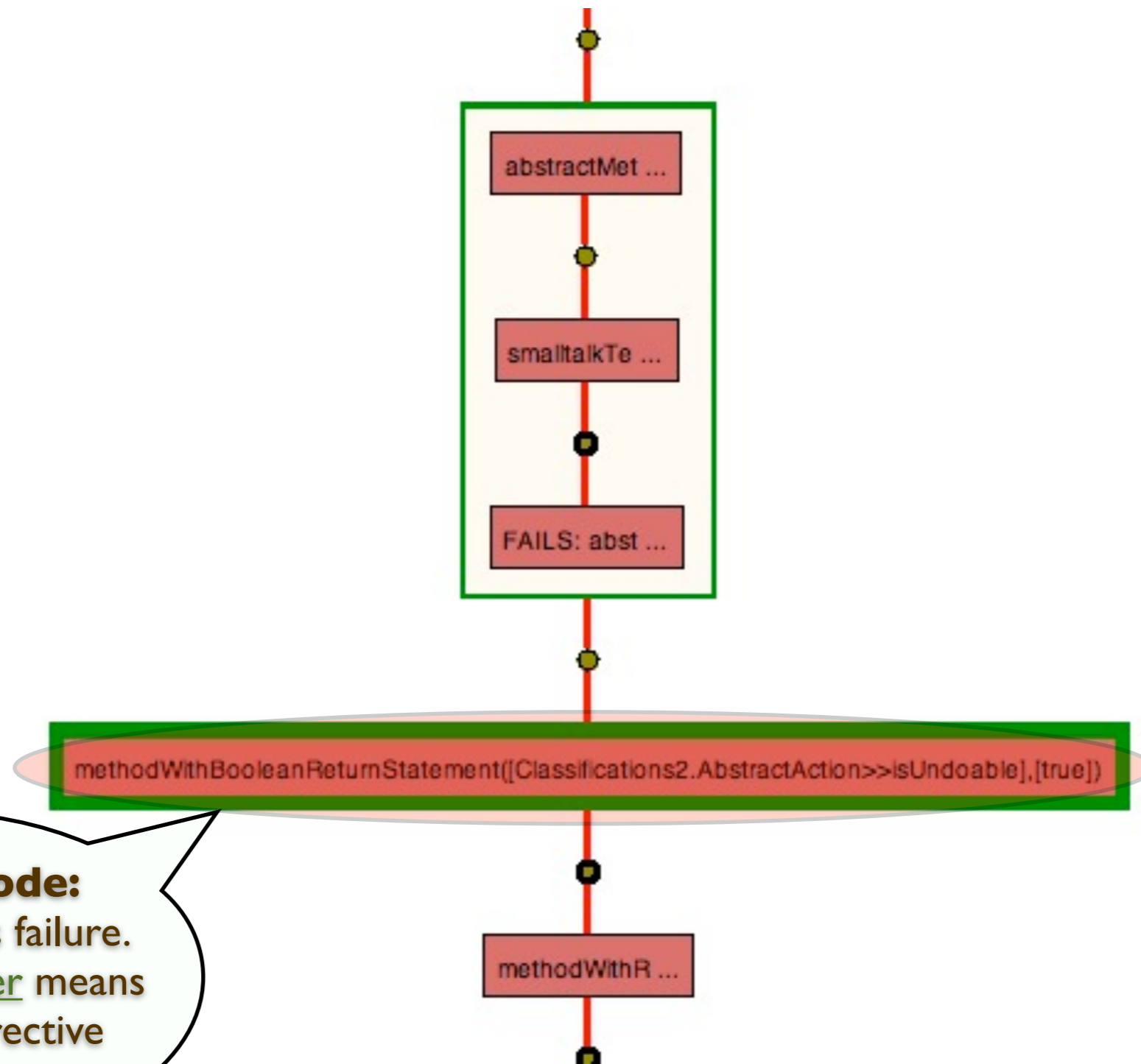
Visualisation of the nodes with corrective actions



Color code:

- Red filling means failure.
- Thick green border means availability of corrective actions.

Visualisation of the nodes with corrective actions



Color code:

- **Red filling** means failure.
- **Thick green border** means availability of corrective actions.

Future work

- How to choose among different solutions.
- Detect solutions that could cause new inconsistencies.

Conclusions

- We can infer solutions from the rules that define consistencies.
- A library of “primitive” solutions to small problems should be provided.
- These solutions can be composed and reused across distinct problems.

Many Thanks !

Questions and feedback are welcomed