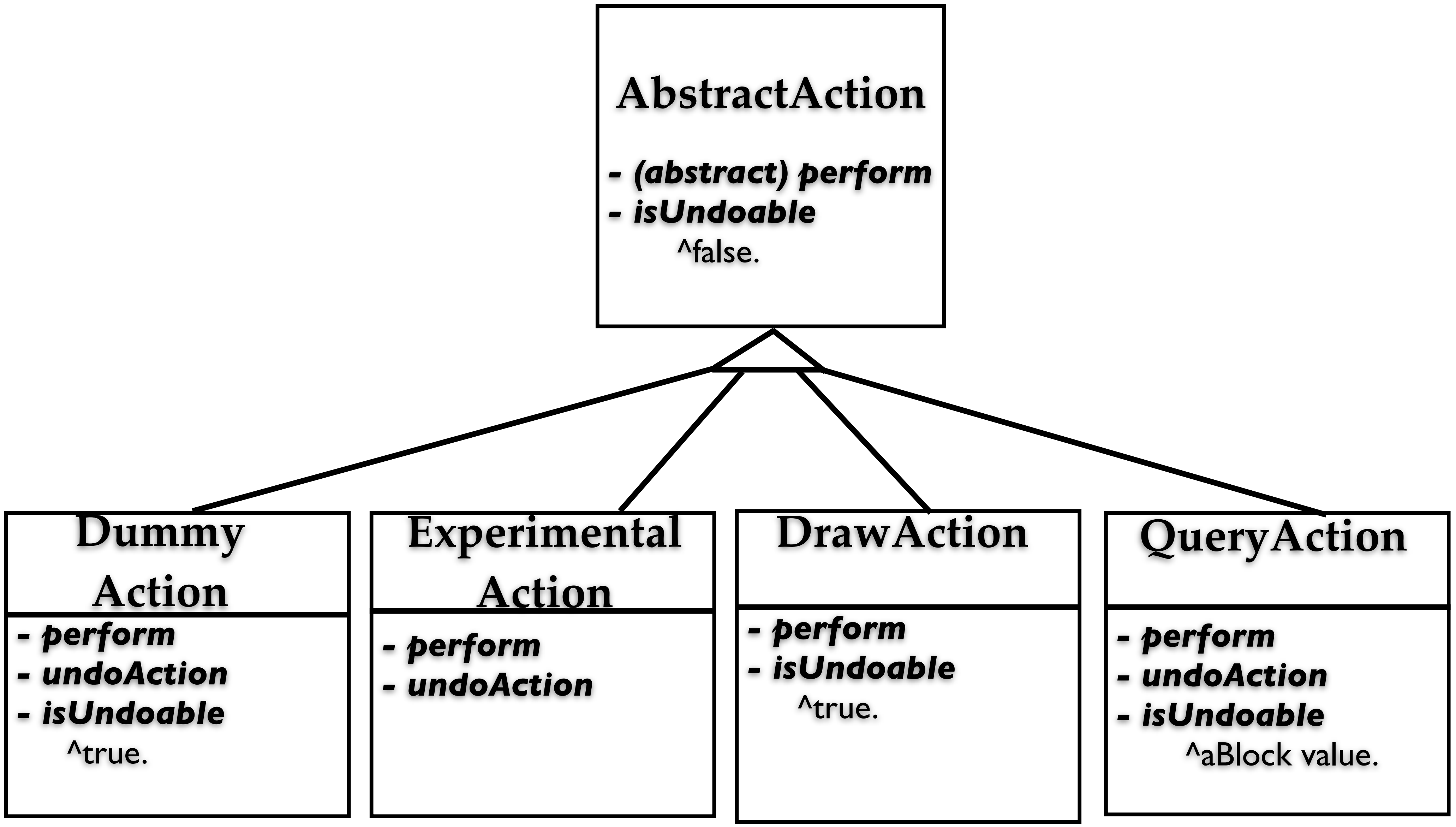
Diagnosis and semi-automatic correction of detected design inconsistencies in source code

Sergio Castro
RELEASeD lab
Université catholique de Louvain
sergio.castro@uclouvain.be
Advisor: Kim Mens

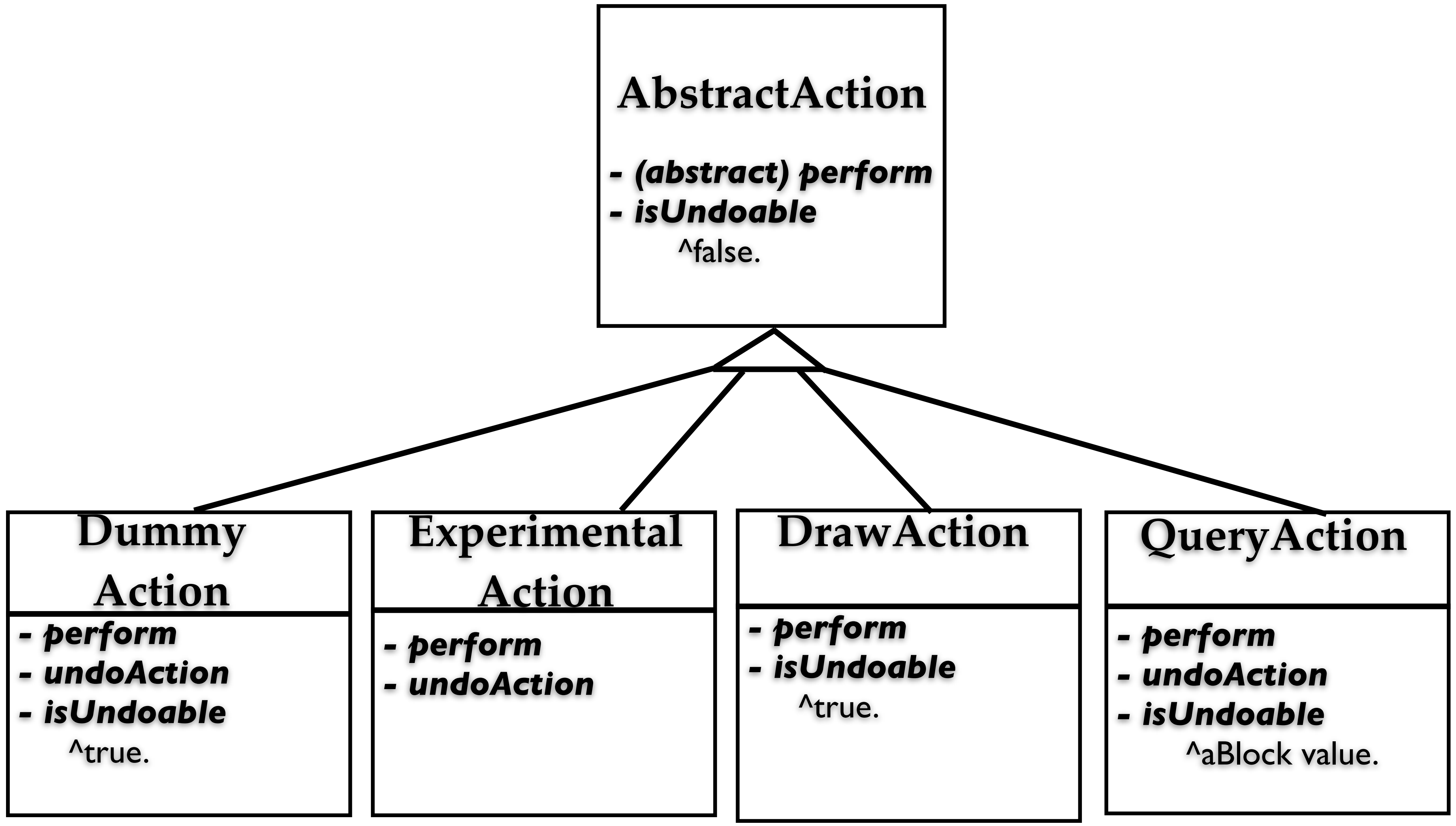
An inconsistency example



2

An inconsistency example

The command design pattern



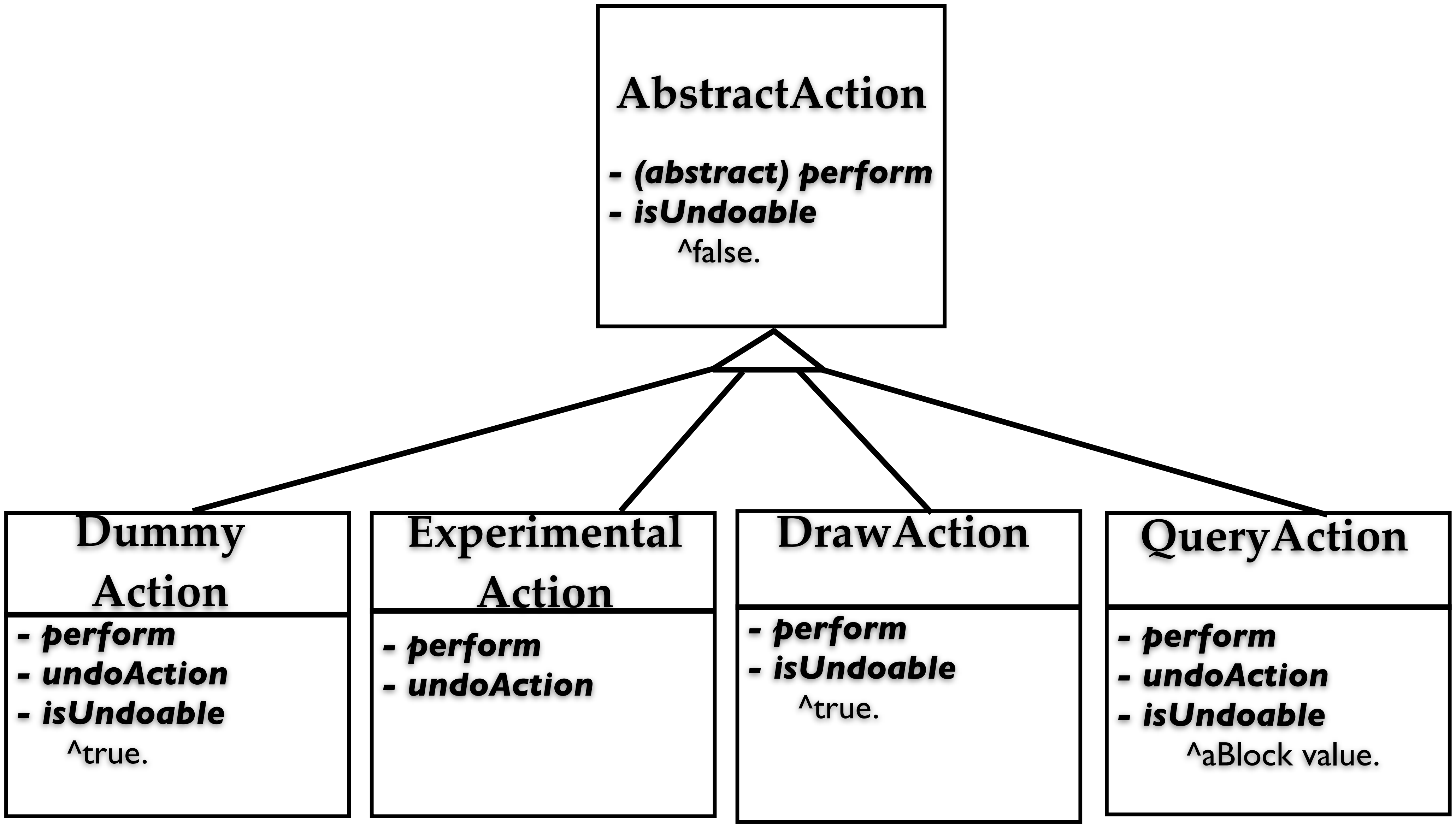
An inconsistency example

The command design pattern

Consistency property:

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

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



2

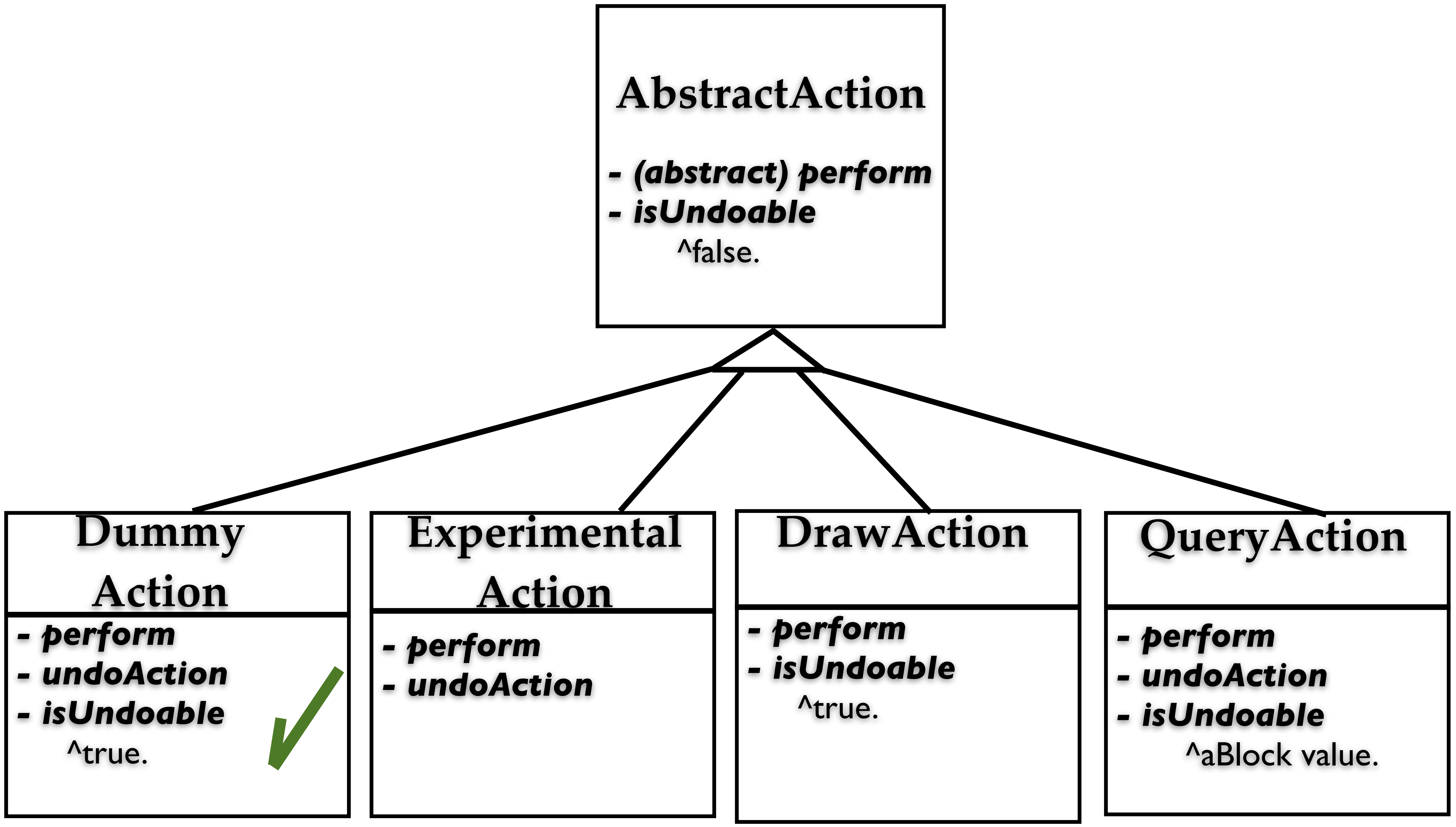
An inconsistency example

The command design pattern

Consistency property:

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

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



2

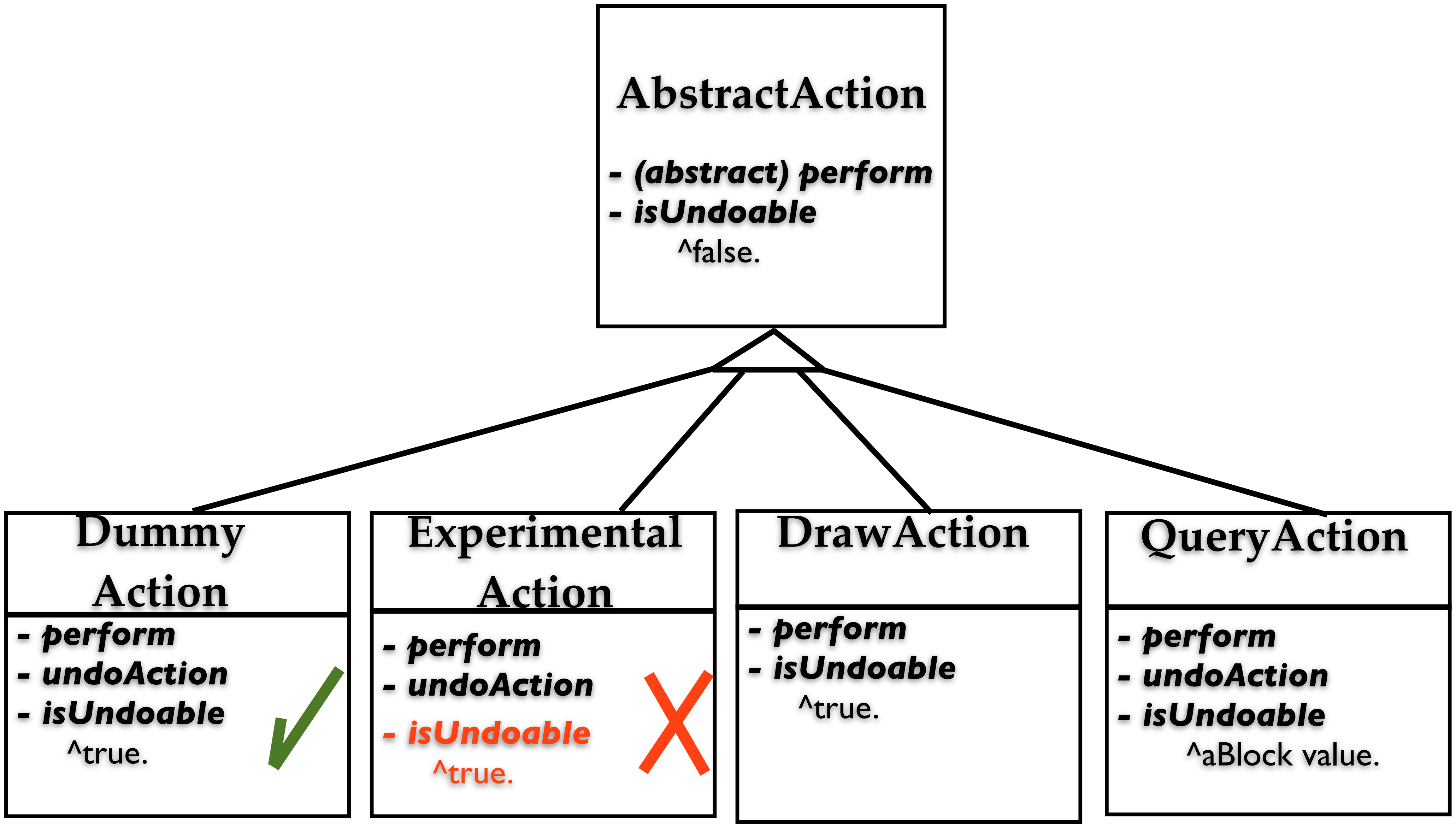
An inconsistency example

The command design pattern

Consistency property:

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

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



2

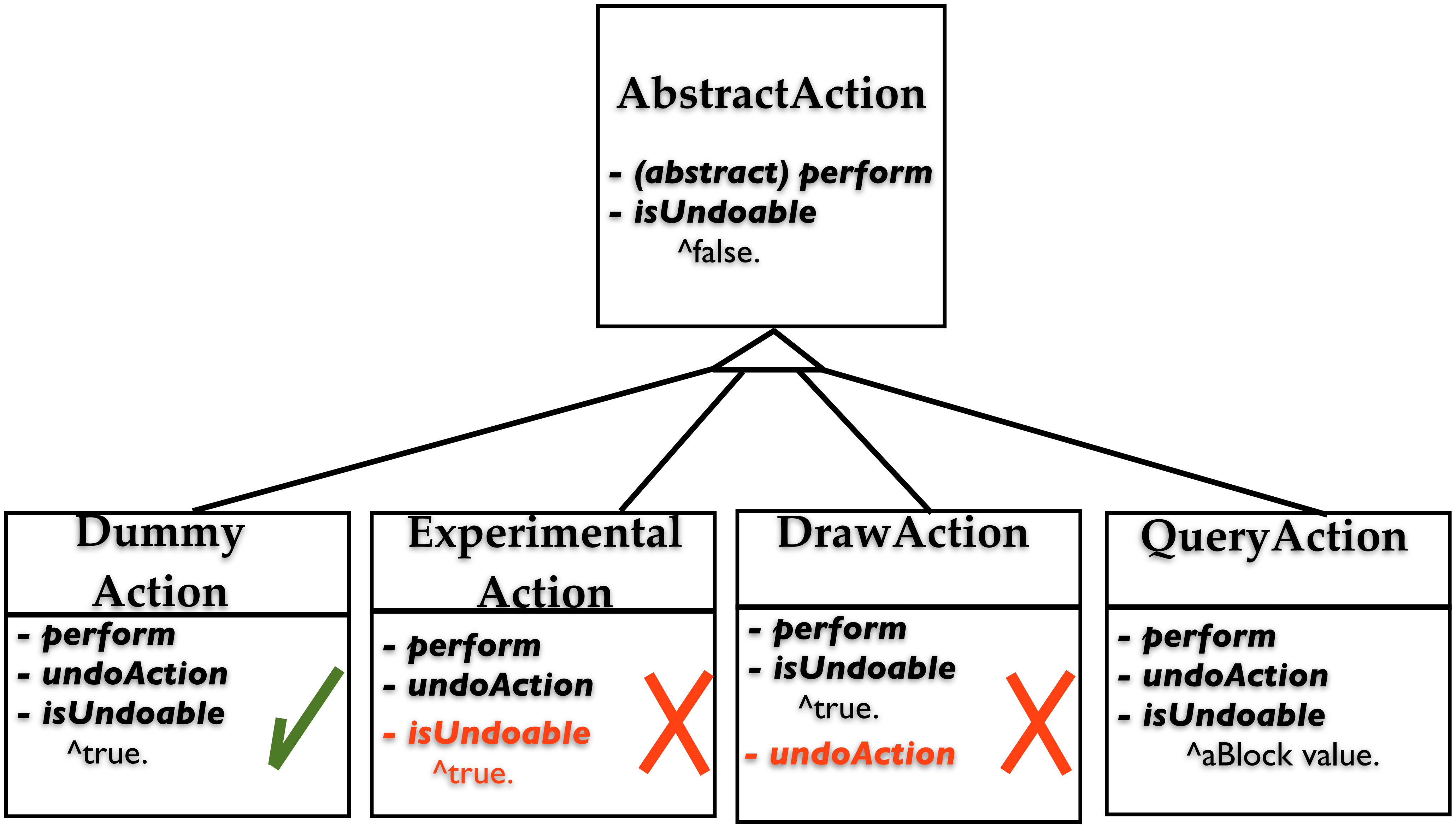
An inconsistency example

The command design pattern

Consistency property:

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

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



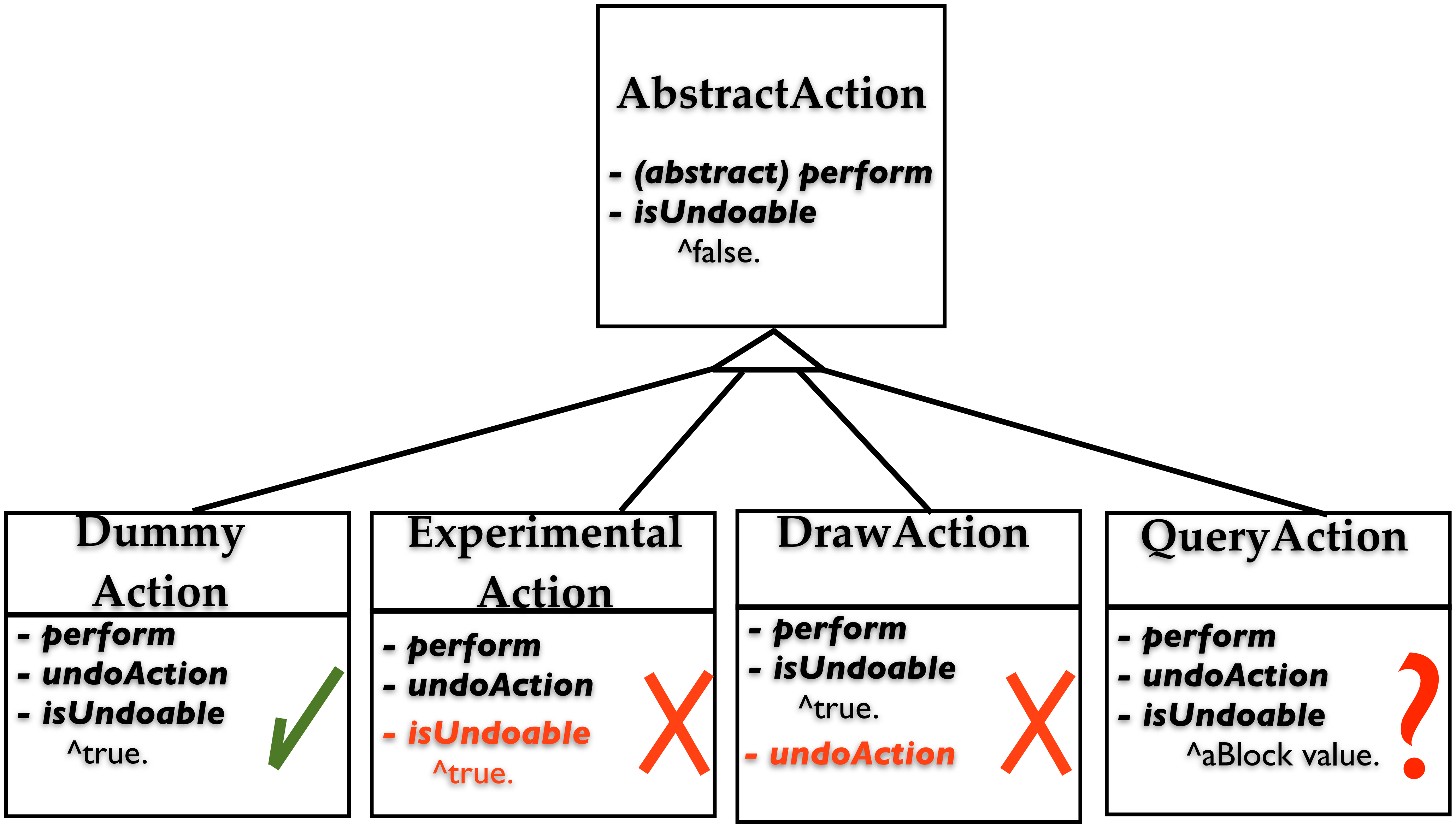
An inconsistency example

The command design pattern

Consistency property:

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

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



2

An inconsistency example

The command design pattern

Consistency property:

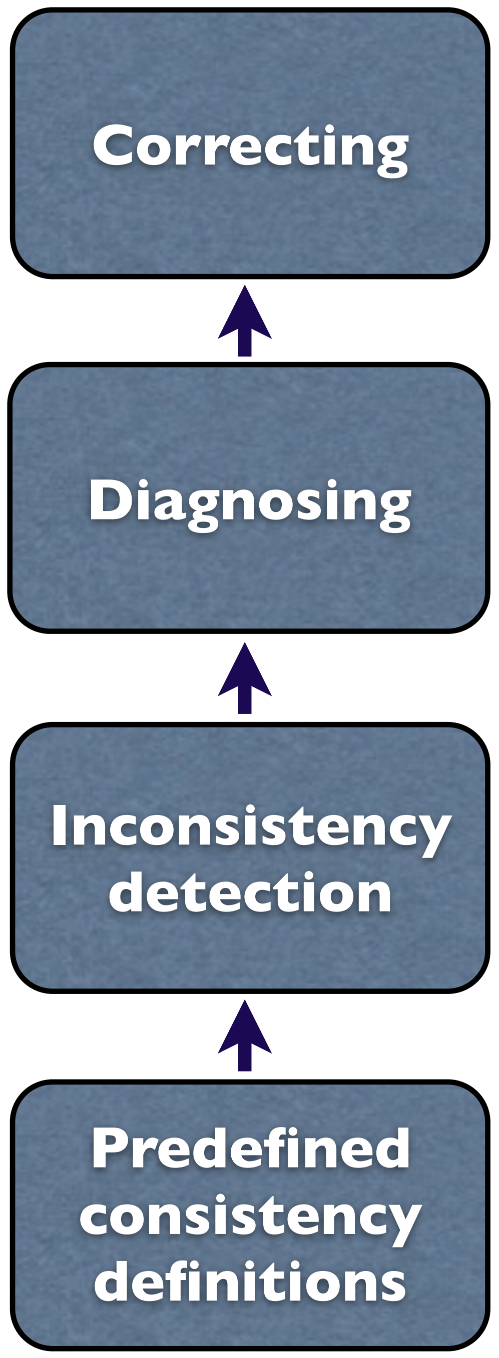
∀ Action, an ***undoAction*** method should be provided

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

User-defined consistency

3

Inconsistency management activities

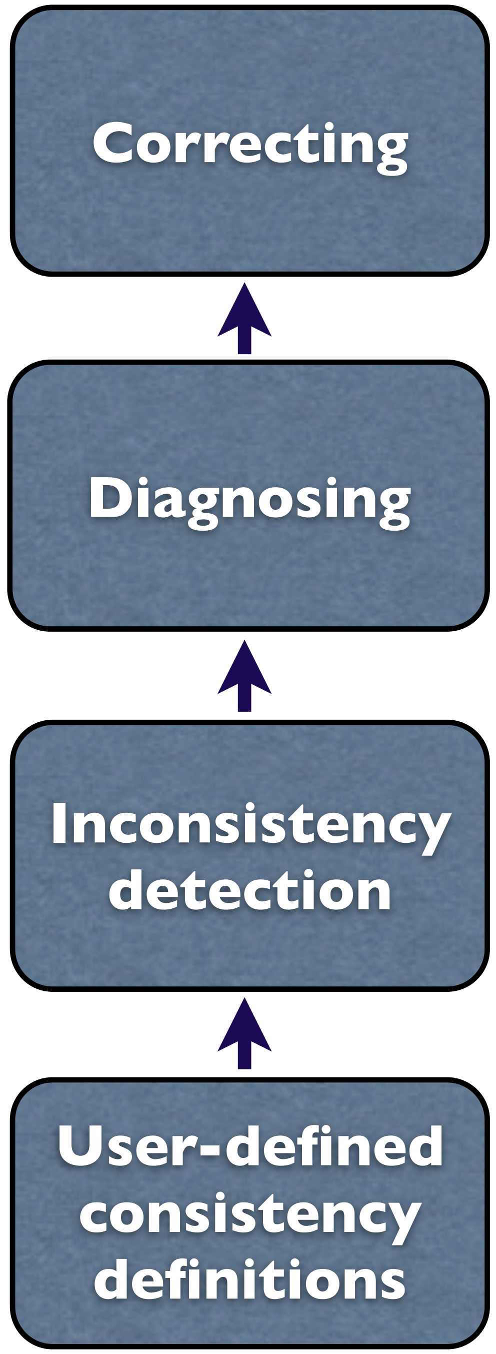


Such as Lint, CheckStyle, etc

Many tools offering support for management of predefined consistencies.

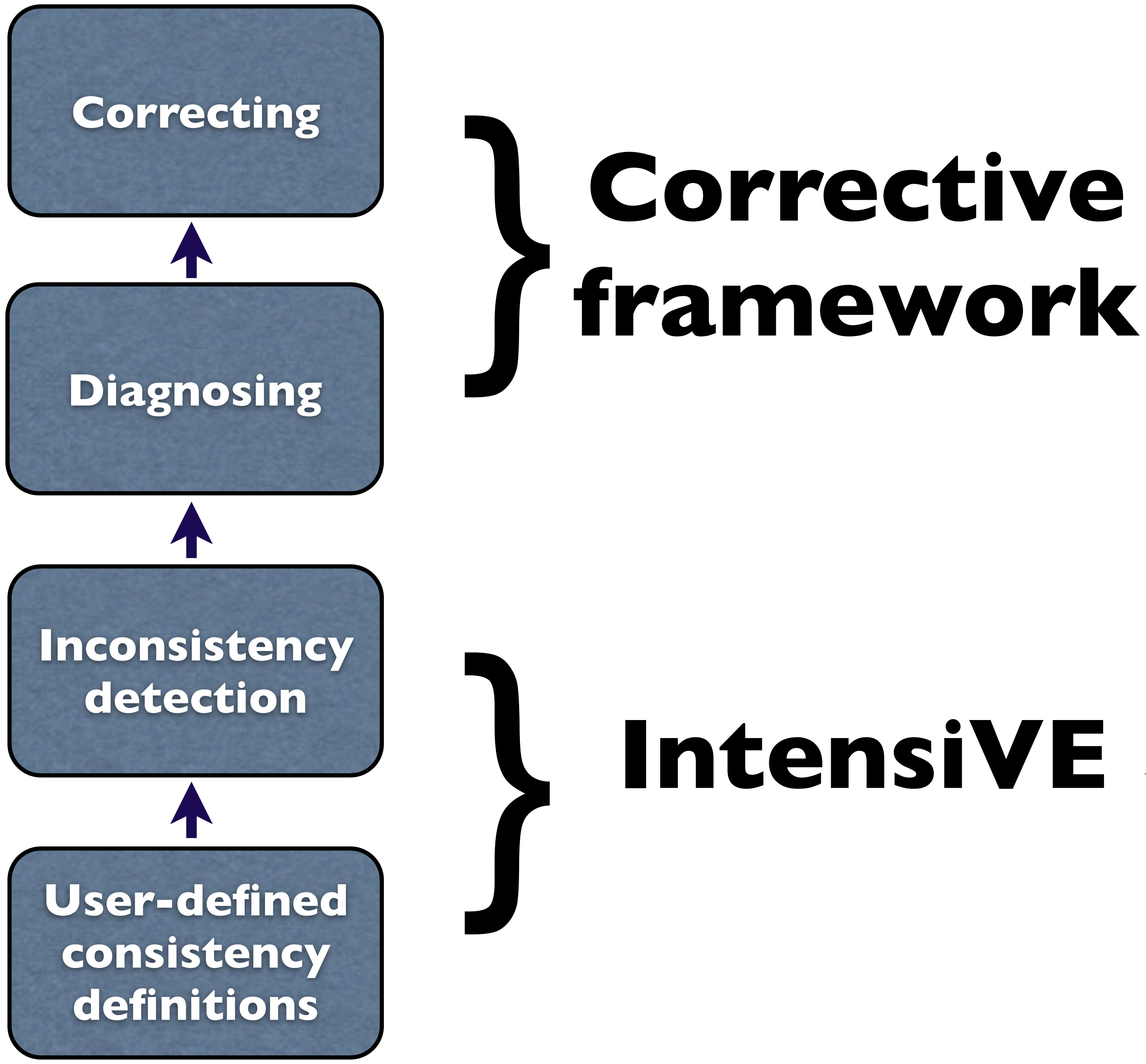
4

Inconsistency management activities



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

Inconsistency management activities

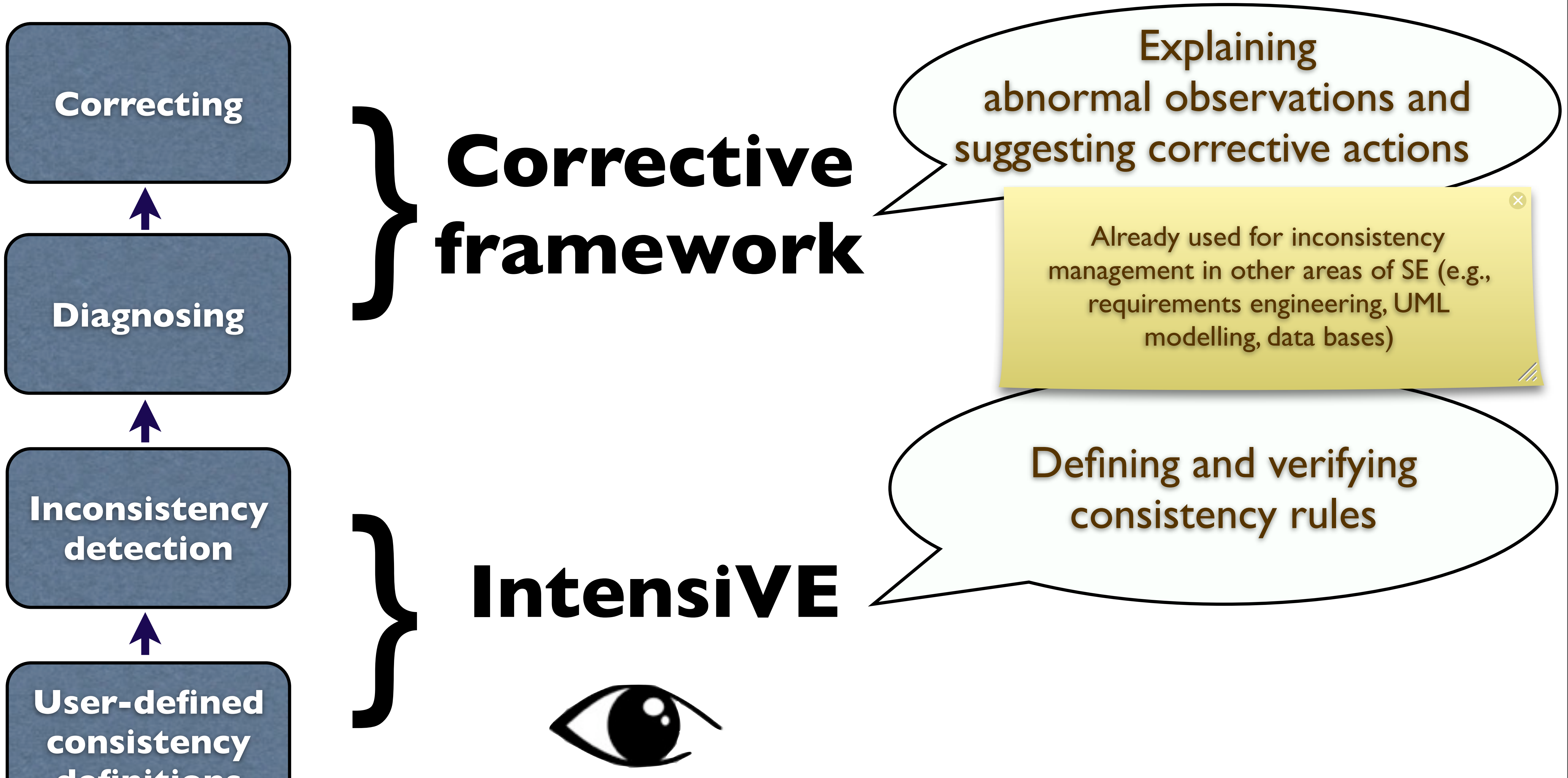


Explaining abnormal observations and suggesting corrective actions

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

Defining and verifying consistency rules

Inconsistency management activities


Correcting
Diagnosing
Inconsistency detection
User-defined consistency definitions
Corrective framework
IntensiVE
Explaining abnormal observations and suggesting corrective actions
Defining and verifying consistency rules

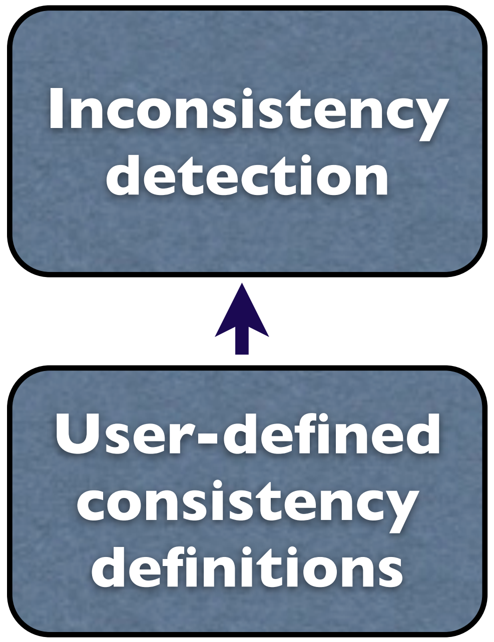
Already used for inconsistency management in other areas of SE (e.g., requirements engineering, UML modelling, data bases)

(http://www.intensive.be)

6

IntensiVE

(http://www.intensive.be)



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

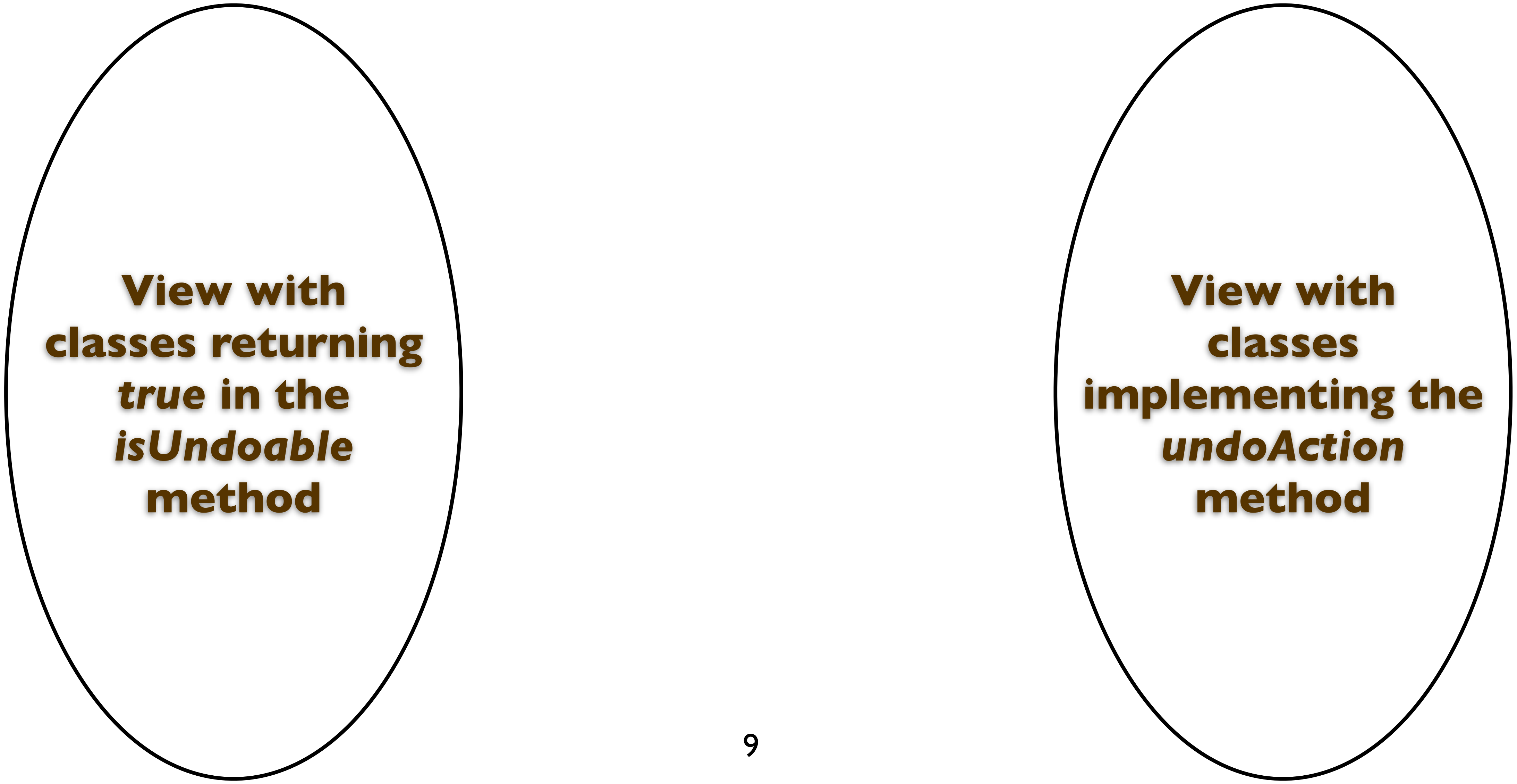
8

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

9

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

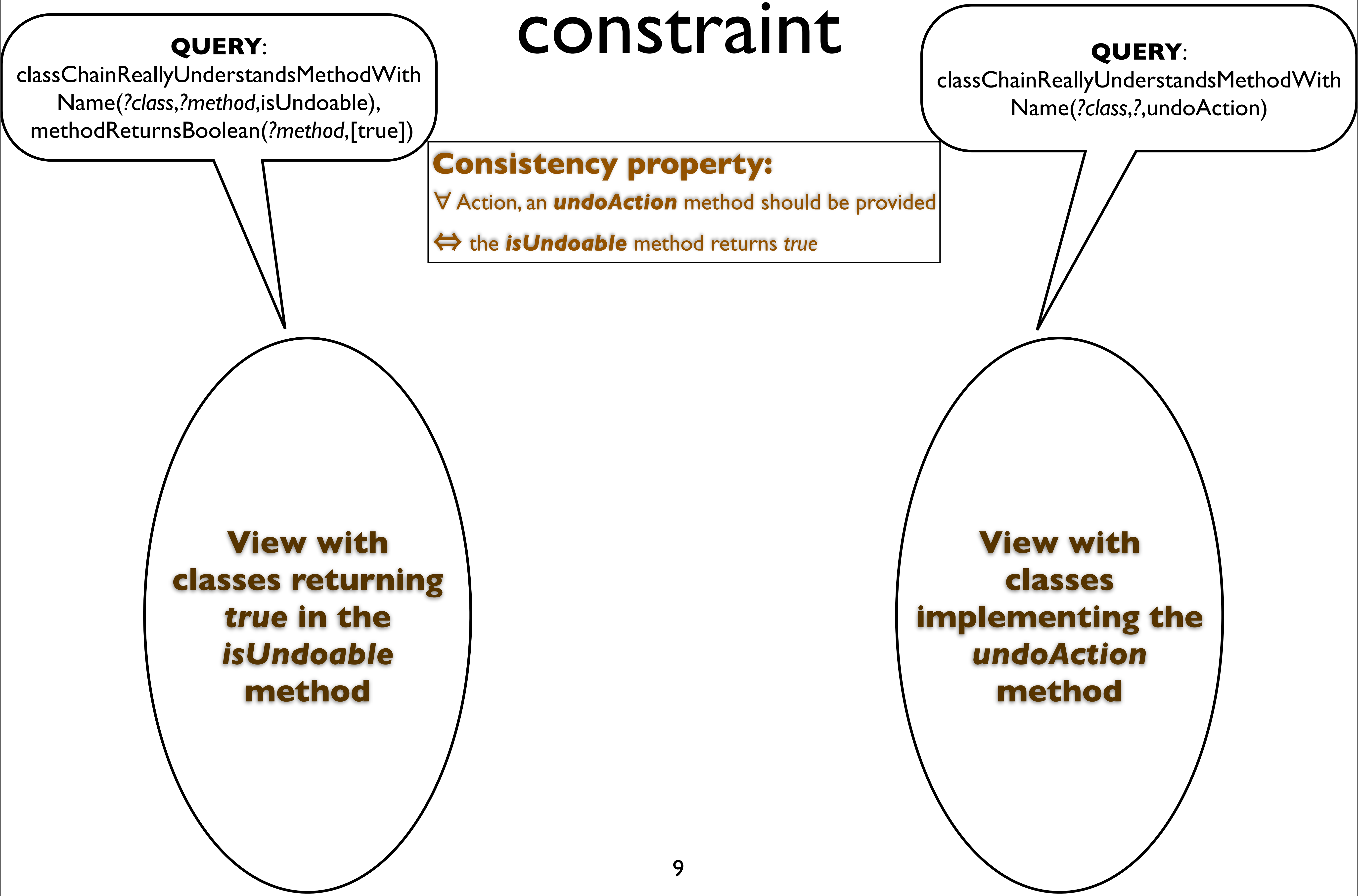
9

Verification of the *Undoable Actions* constraint

Consistency property:

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

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



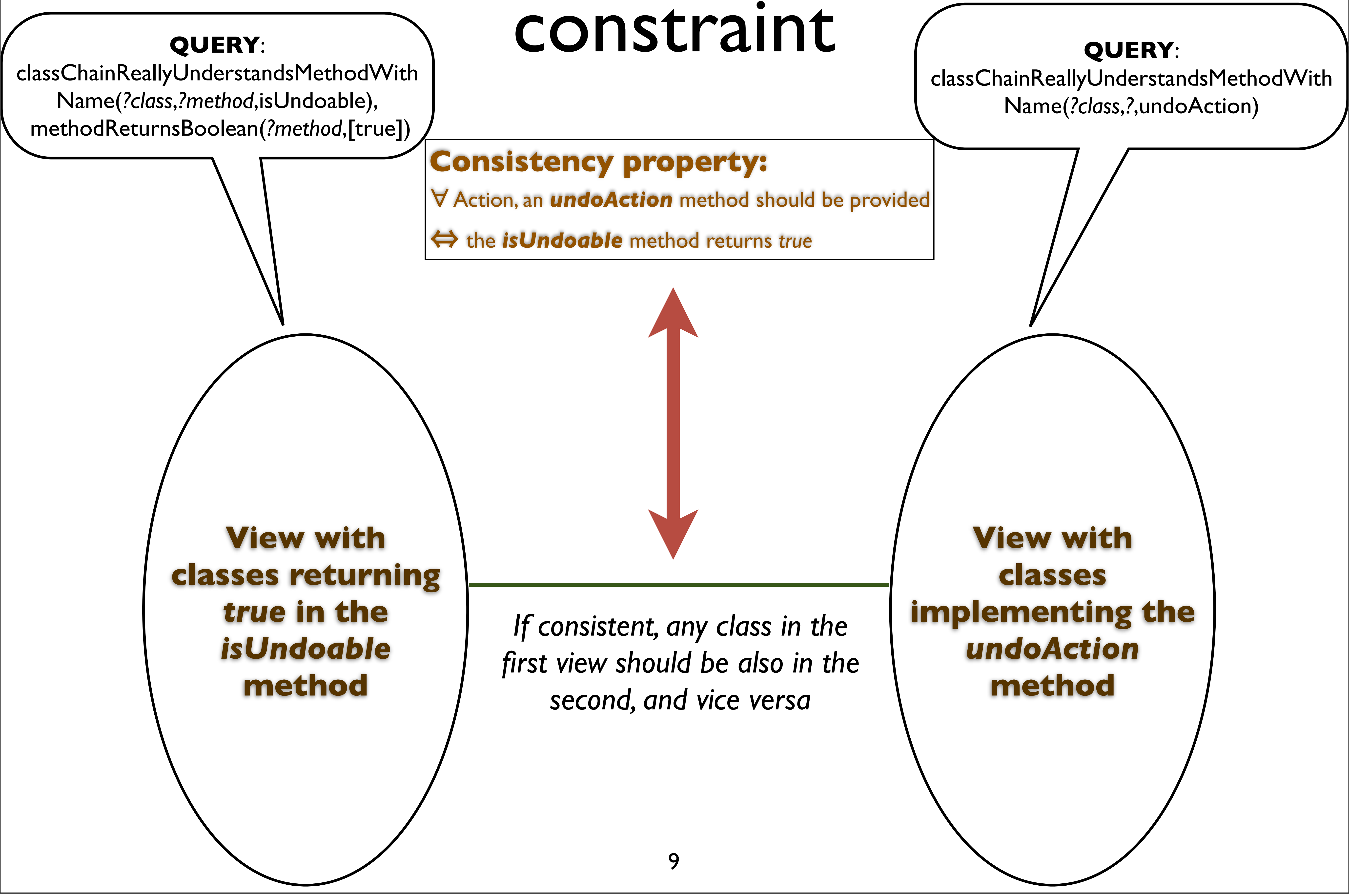
QUERY:
classChainReallyUnderstandsMethodWithName(*?class*,*?method*,isUndoable), methodReturnsBoolean(*?method*,[true])

QUERY:
classChainReallyUnderstandsMethodWithName(*?class*,?,undoAction)

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

9

Verification of the *Undoable Actions* constraint



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

QUERY:
classChainReallyUnderstandsMethodWithName(*?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

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

Verification of alternative views

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

QUERY:
classChainReallyUnderstandsMethodWithName(*?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

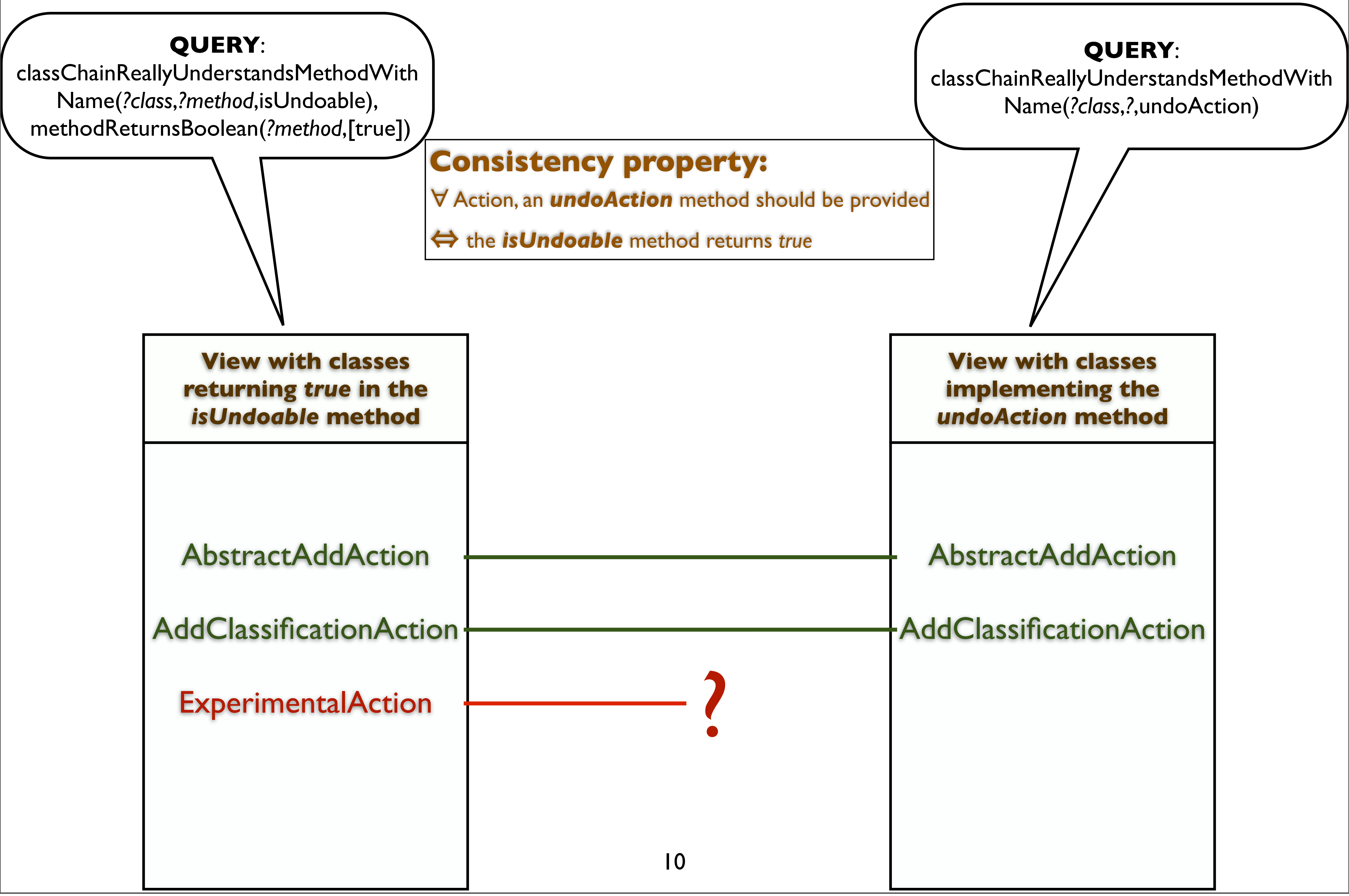
10

Verification of alternative views

Consistency property:

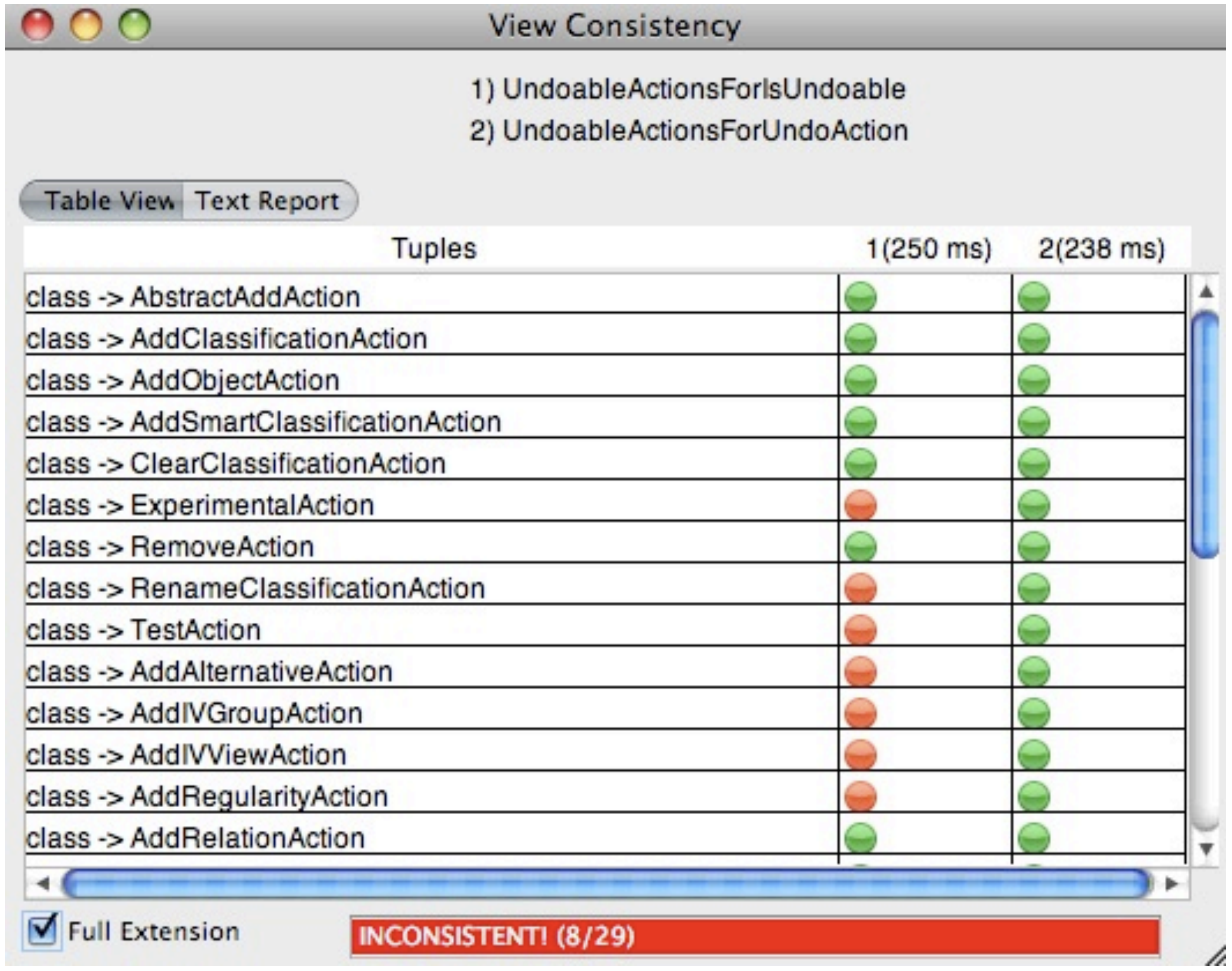
∀ Action, an ***undoAction*** method should be provided

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


QUERY: classChainReallyUnderstandsMethodWithName(*?class*,*?method*,isUndoable), methodReturnsBoolean(*?method*,[true])
QUERY: classChainReallyUnderstandsMethodWithName(*?class*,?,undoAction)
View with classes returning *true* in the *isUndoable* method
View with classes implementing the *undoAction* method
AbstractAddAction
AddClassificationAction
ExperimentalAction
?
AbstractAddAction
AddClassificationAction

IntensiVE

(consistency checking)



View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Table View | Text Report

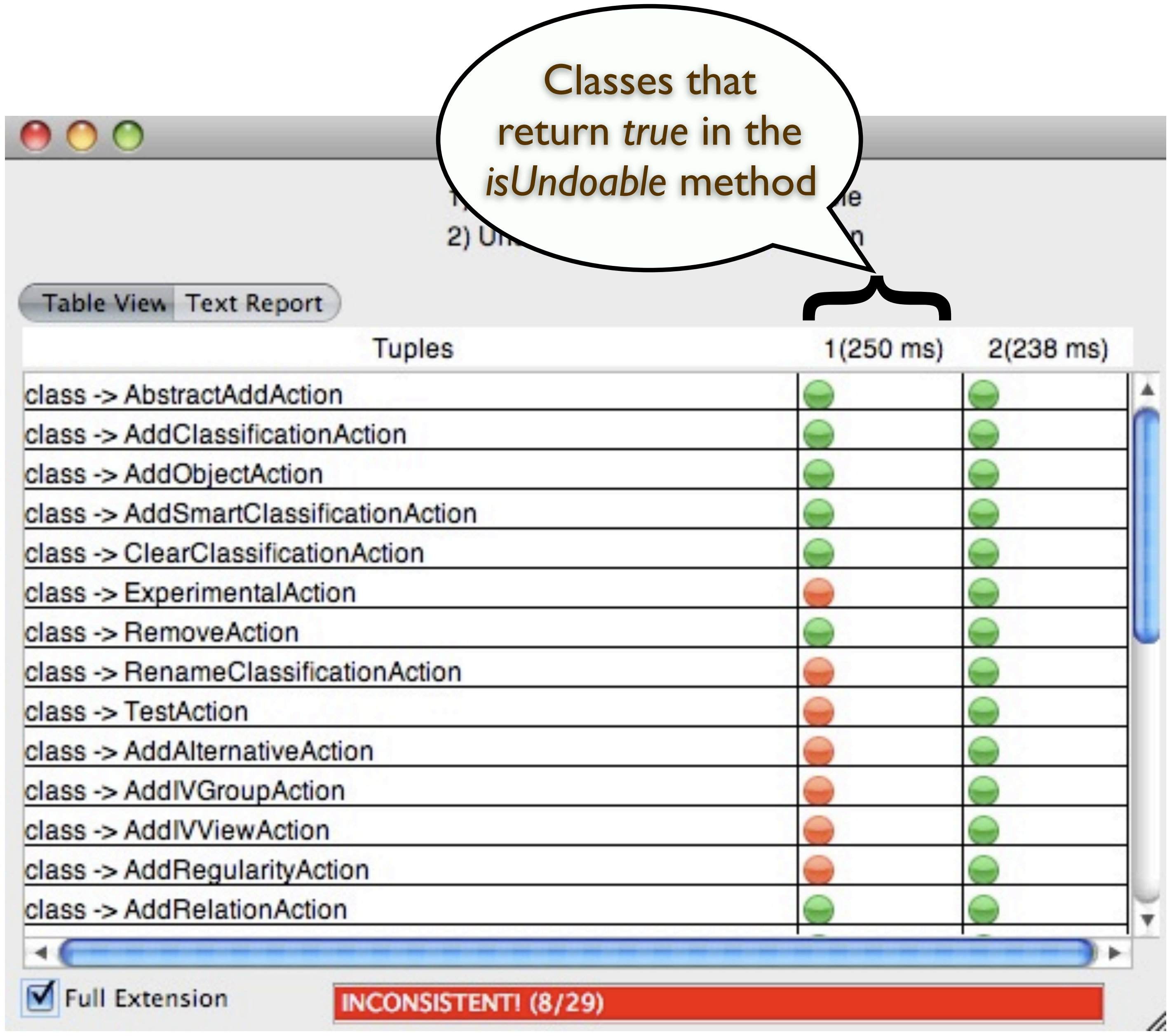
Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassificationAction	🟢	🟢
class -> AddObjectAction	🟢	🟢
class -> AddSmartClassificationAction	🟢	🟢
class -> ClearClassificationAction	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴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

[✓] Full Extension

INCONSISTENT! (8/29)

IntensiVE

(consistency checking)



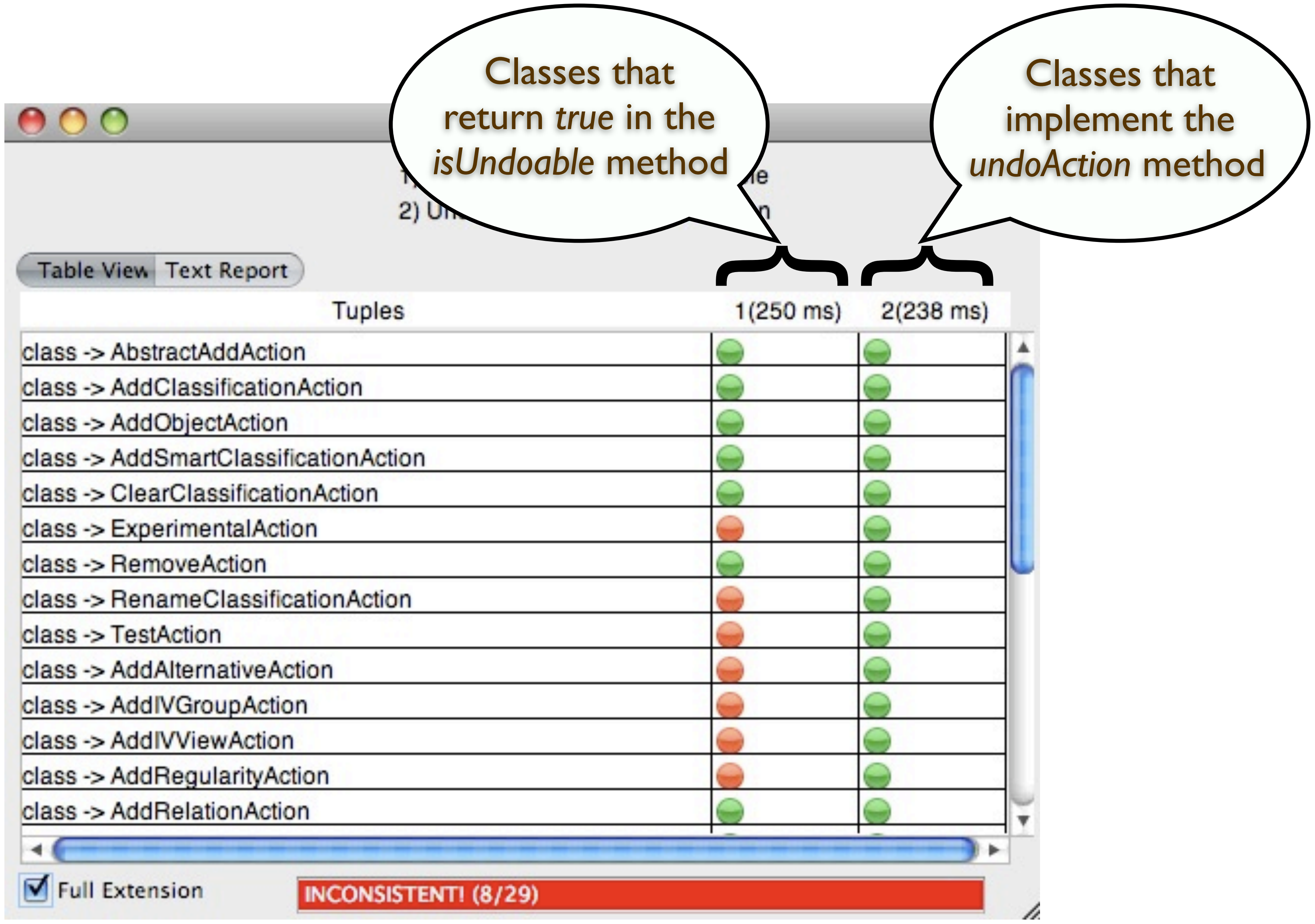
Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassificationAction	🟢	🟢
class -> AddObjectAction	🟢	🟢
class -> AddSmartClassificationAction	🟢	🟢
class -> ClearClassificationAction	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴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

INCONSISTENT! (8/29)

11

IntensiVE

(consistency checking)


Classes that return *true* in the *isUndoable* method
Classes that implement the *undoAction* method
Table View
Text Report
Full Extension

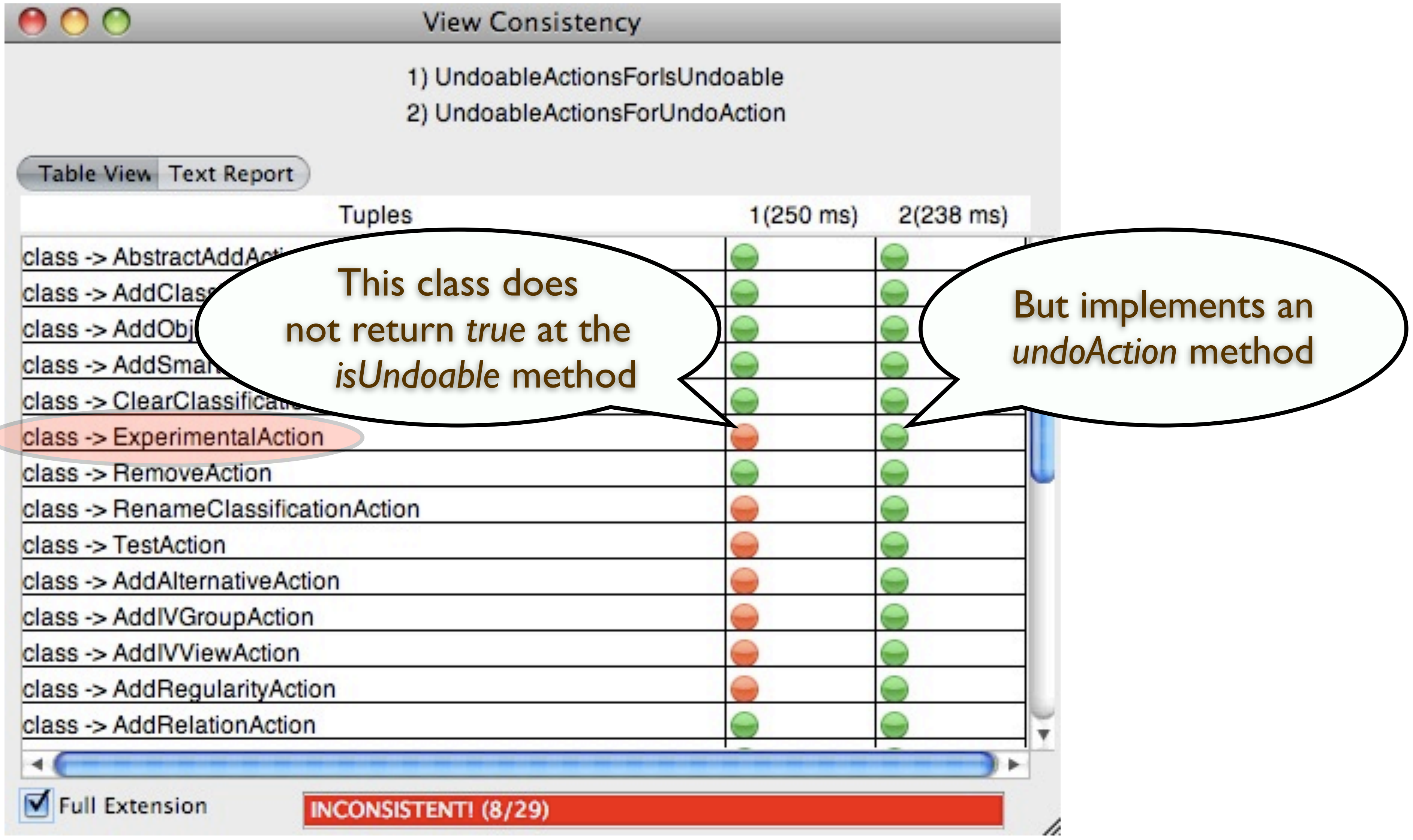
Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAction	🟢	🟢
class -> AddClassificationAction	🟢	🟢
class -> AddObjectAction	🟢	🟢
class -> AddSmartClassificationAction	🟢	🟢
class -> ClearClassificationAction	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴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

INCONSISTENT! (8/29)

11

IntensiVE

(consistency checking)



View Consistency
Table View
Text Report
Full Extension
This class does not return *true* at the *isUndoable* method
But implements an *undoAction* method

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAct	🟢	🟢
class -> AddClass	🟢	🟢
class -> AddObj	🟢	🟢
class -> AddSmar	🟢	🟢
class -> ClearClassifica	🟢	🟢
class -> ExperimentalAction	🔴	🟢
class -> RemoveAction	🟢	🟢
class -> RenameClassificationAction	🔴	🟢
class -> TestAction	🔴	🟢
class -> AddAlternativeAction	🔴	🟢
class -> AddIVGroupAction	🔴	🟢
class -> AddIVViewAction	🔴	🟢
class -> AddRegularityAction	🔴	🟢
class -> AddRelationAction	🟢	🟢

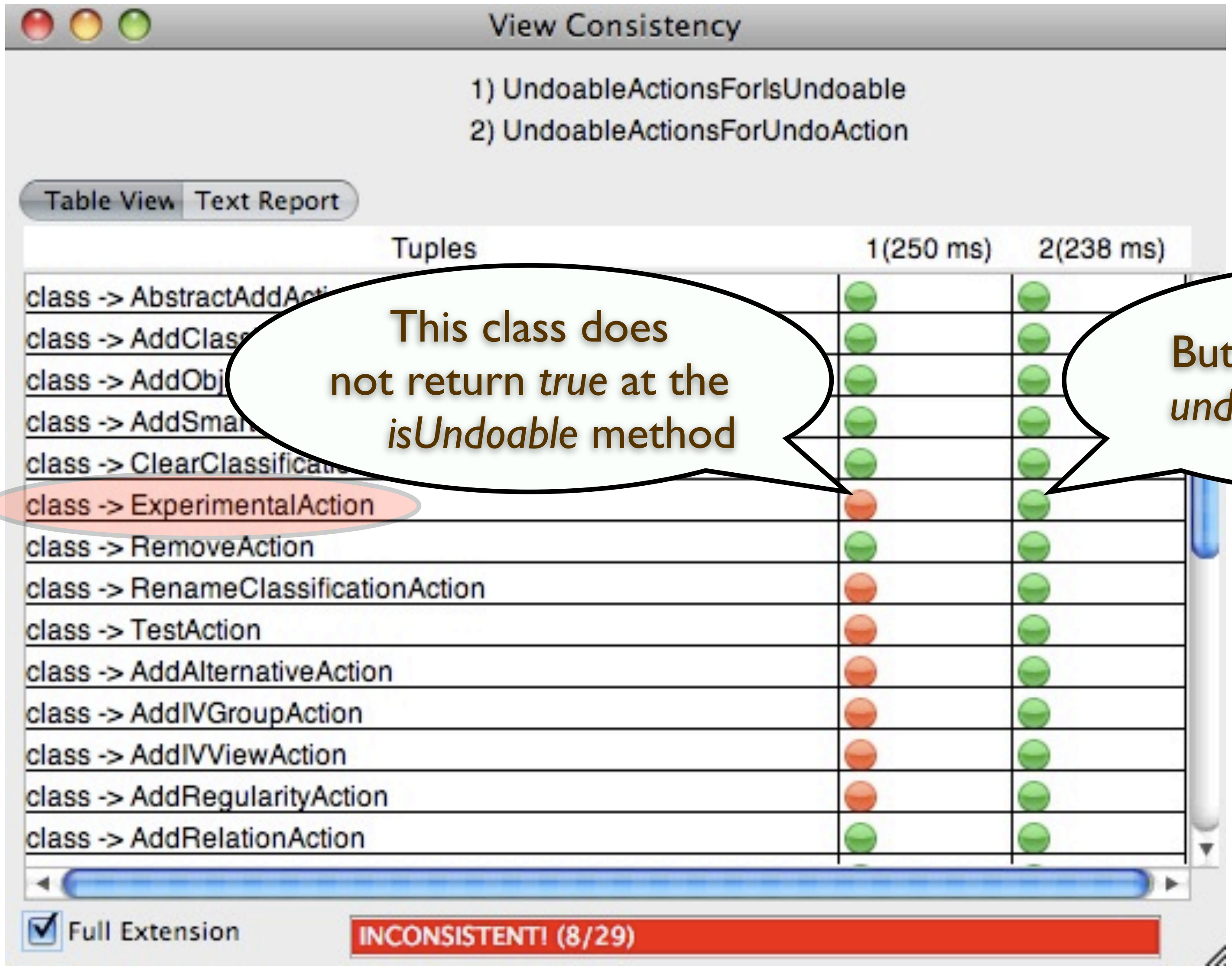
INCONSISTENT! (8/29)

12

IntensiVE

(consistency checking)

ExperimentalAction



View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

Tuples	1(250 ms)	2(238 ms)
class -> AbstractAddAct	🟩	🟩
class -> AddClass	🟩	🟩
class -> AddObj	🟩	🟩
class -> AddSmar	🟩	🟩
class -> ClearClassifica	🟩	🟩
class -> ExperimentalAction	🟥	🟩
class -> RemoveAction	🟩	🟩
class -> RenameClassificationAction	🟥	🟩
class -> TestAction	🟥	🟩
class -> AddAlternativeAction	🟥	🟩
class -> AddIVGroupAction	🟥	🟩
class -> AddIVViewAction	🟥	🟩
class -> AddRegularityAction	🟥	🟩
class -> AddRelationAction	🟩	🟩

☑ Full Extension

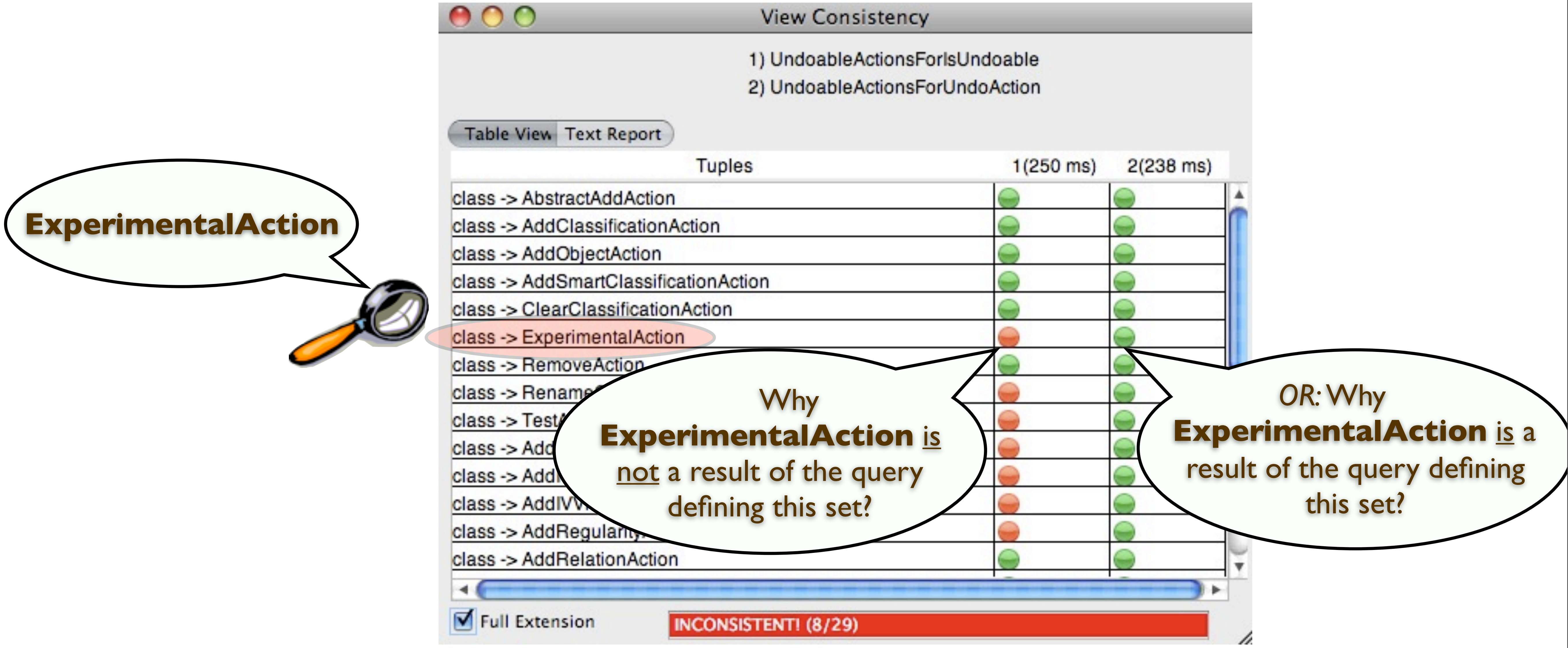
INCONSISTENT! (8/29)

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

But implements an *undoAction* method

IntensiVE

(diagnosing inconsistencies)


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

View Consistency

1) UndoableActionsForIsUndoable
2) UndoableActionsForUndoAction

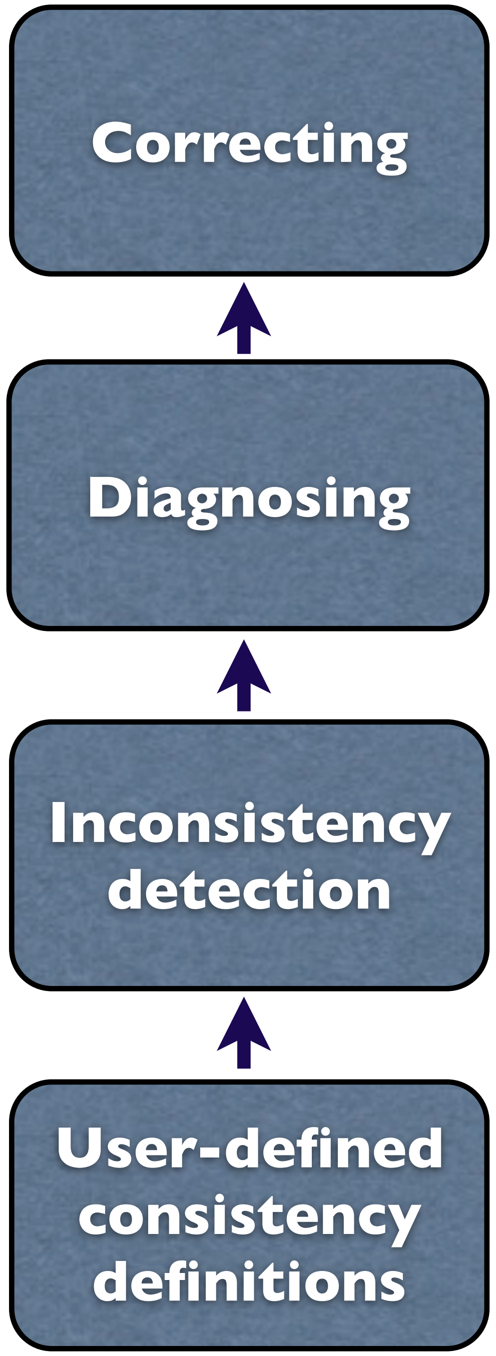
Table View | Text Report

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 -> AddIV	🔴	🟢
class -> AddRegular	🔴	🟢
class -> AddRelationAction	🟢	🟢

☑ Full Extension

INCONSISTENT! (8/29)

Inconsistency management activities



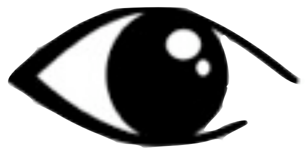
Corrective framework

Explaining abnormal observations and suggesting corrective actions

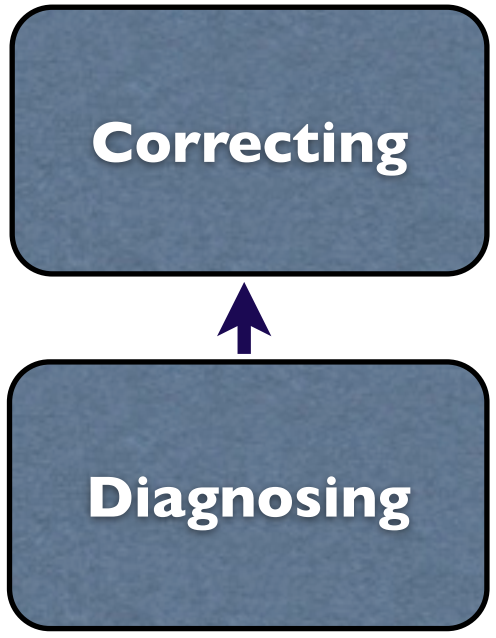
Already used for inconsistency management in other areas of SE (e.g., requirements engineering, UML modelling, data bases)

IntensiVE

Defining and verifying consistency rules



(http://www.intensive.be)



Corrective framework

What is diagnosis?

Generating hypotheses explaining abnormal observations

Abductive reasoning

16

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***.

17

Abduction is suitable for

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

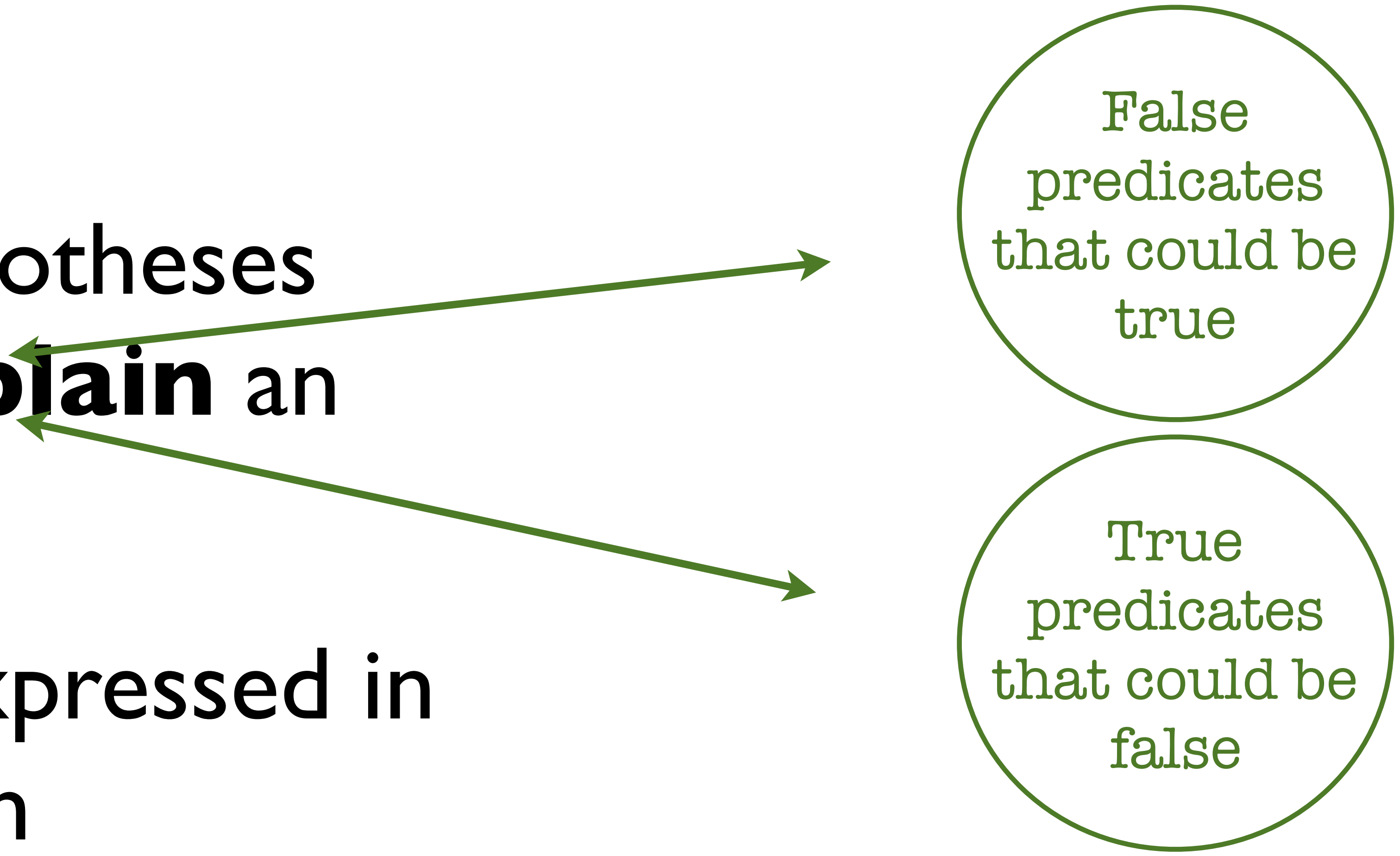
False predicates that could be true

17

Abduction is suitable for

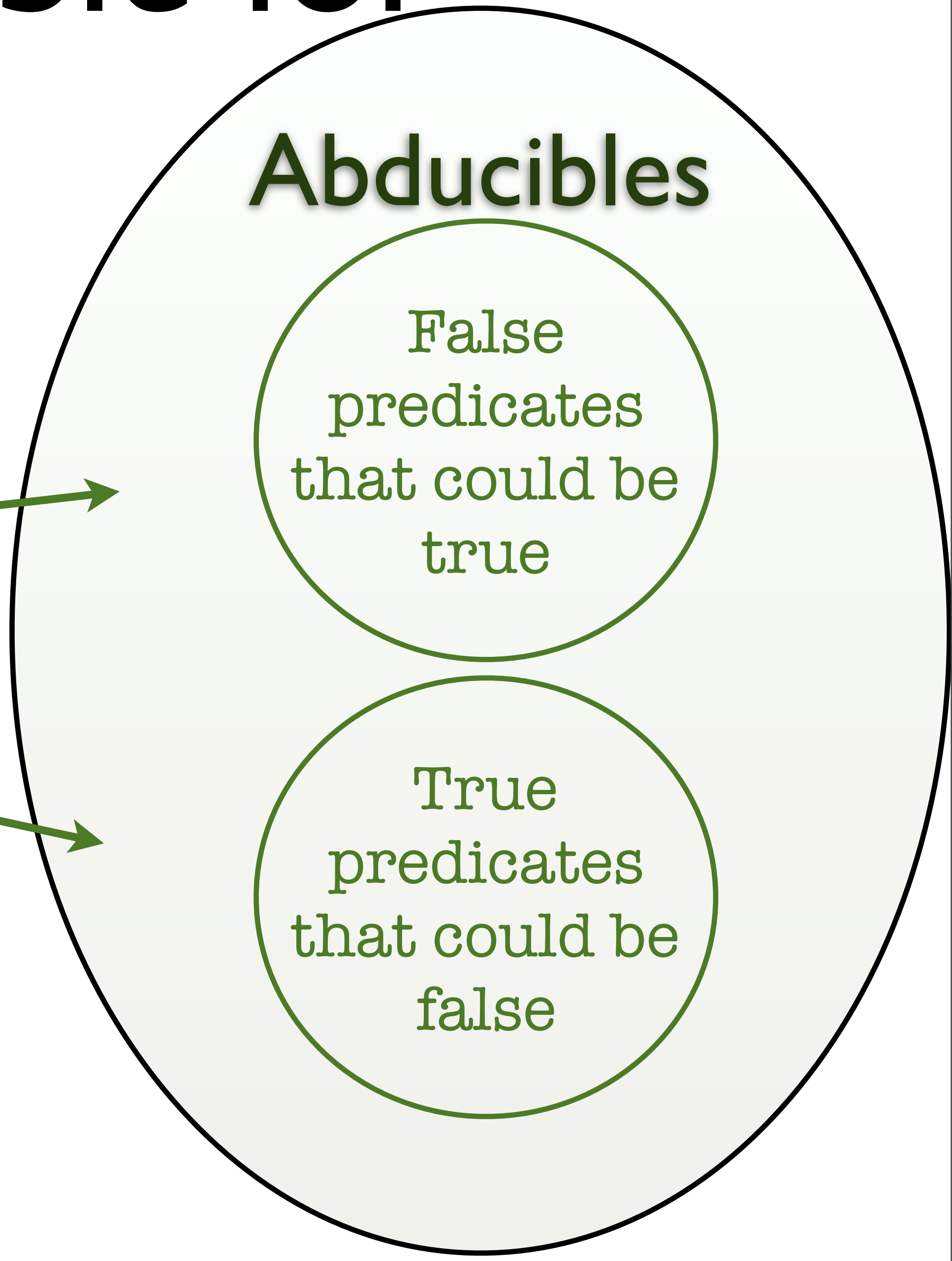
- Generating hypotheses that would **explain** an evidence.

- 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***.


Abducibles
False predicates that could be true
True predicates that could be false

We can think in "**Abducibles**" as "**Correctables**"

17

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:

failing

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

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

Abducibles

bird(?x).

broken-wing(?x).

Observation:

flies(opus)

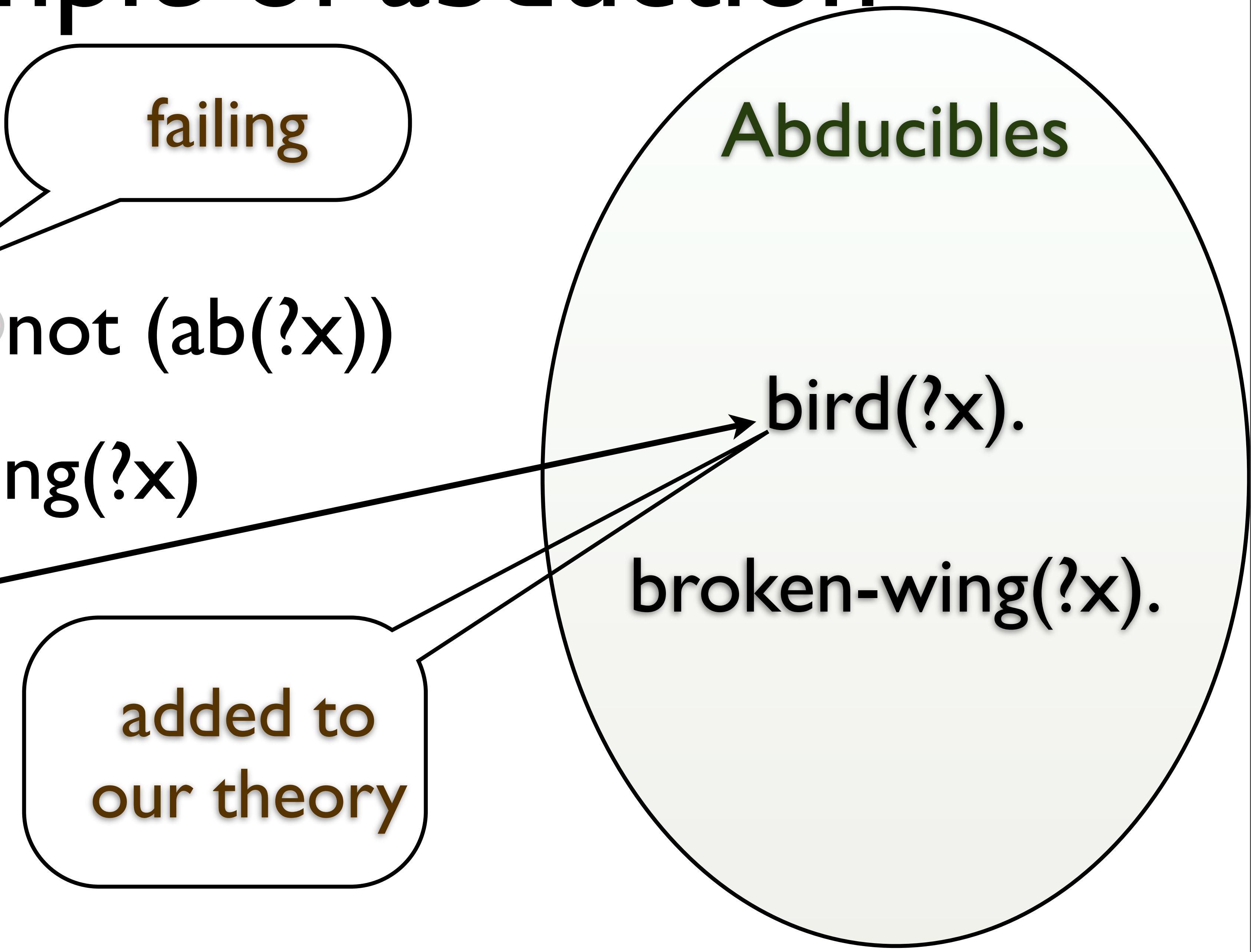
An example of abduction

Theory:

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

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

bird(opus)



Observation:

flies(opus)

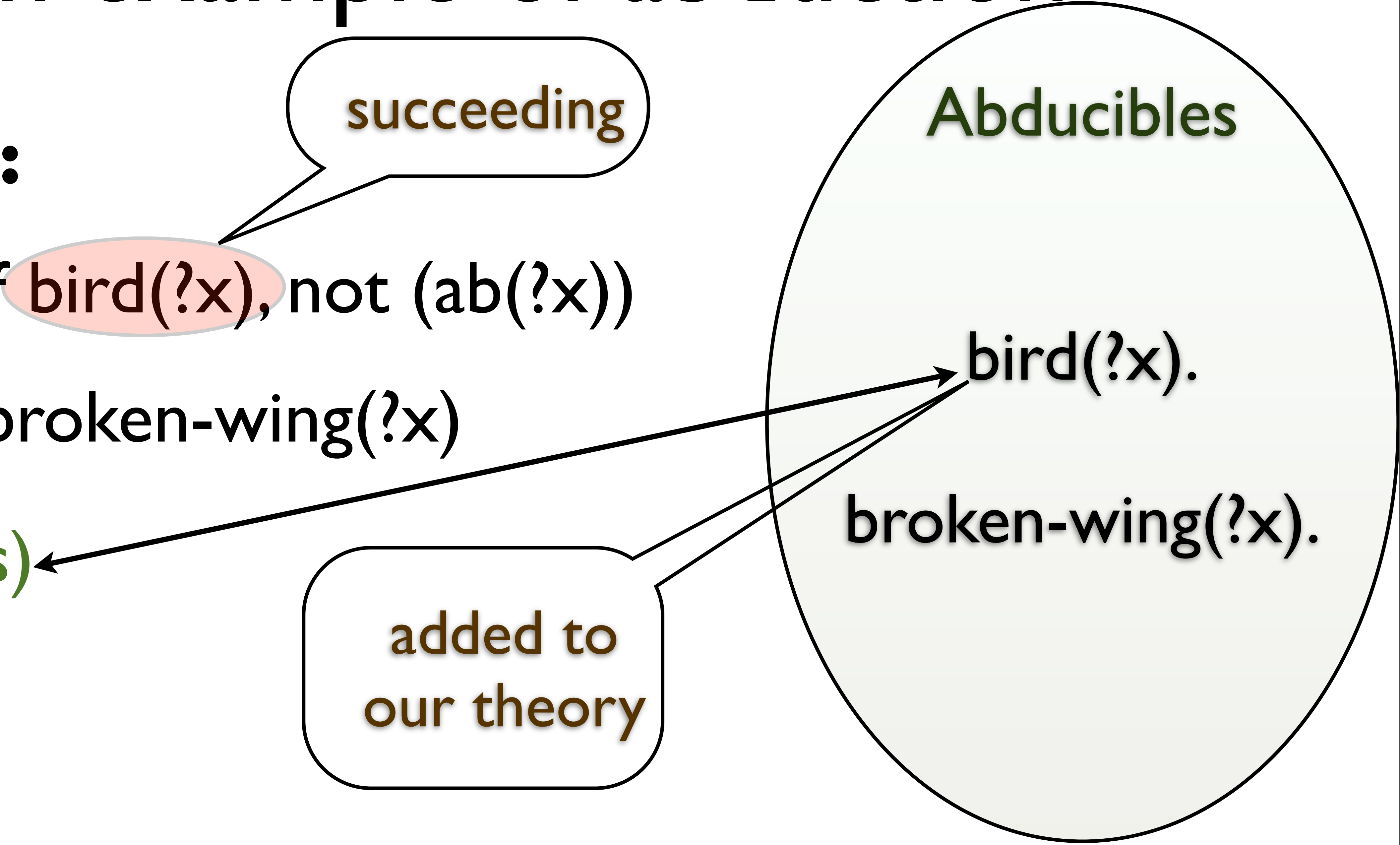
An example of abduction

Theory:

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

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

bird(opus)



Observation:

flies(opus)

18

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).

19

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).

Observation:

flies(tweety)

19

An example of abduction

Theory:

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

failing

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

bird(tweety)

broken-wing(tweety)

Abducibles

bird(?x).

broken-wing(?x).

Observation:

flies(tweety)

19

An example of abduction

Theory:

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

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

bird(tweety)

~~broken-wing(tweety)~~

failing

Abducibles

bird(?x).

broken-wing(?x).

retracted from our theory

Observation:

flies(tweety)

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

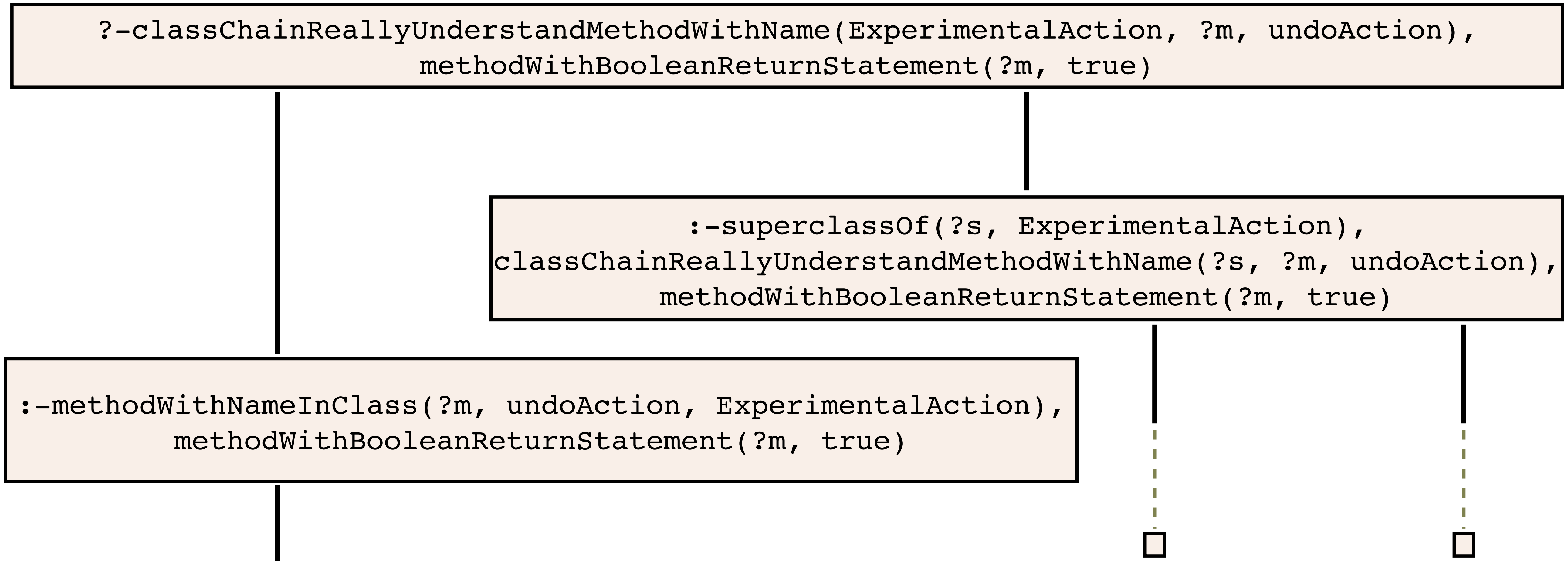
Abducibles

bird(?x).

broken-wing(?x).

retracted from our theory

How to choose abducible predicates



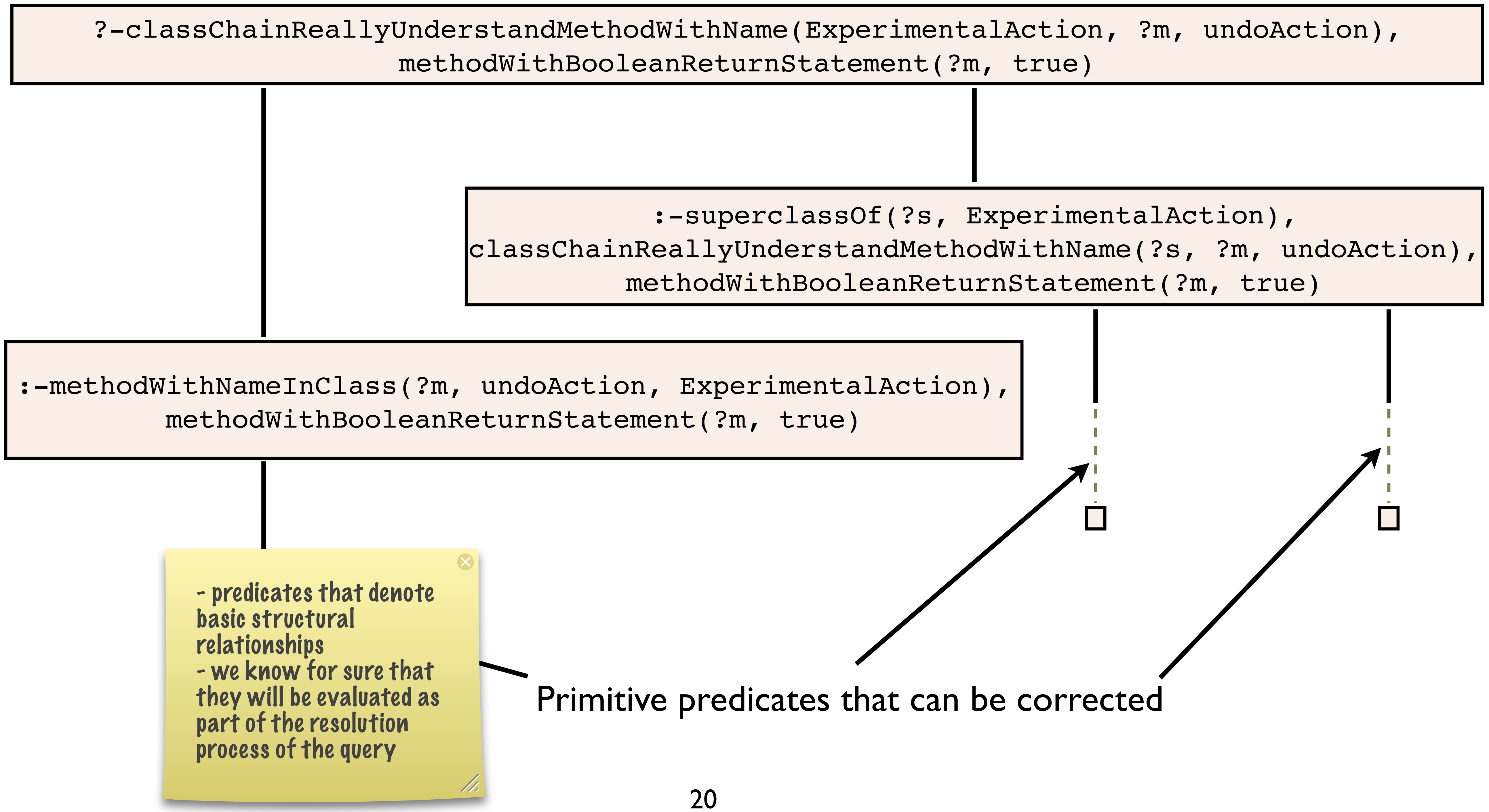
```
?-classChainReallyUnderstandMethodWithName(ExperimentalAction, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)
```

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

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

- 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



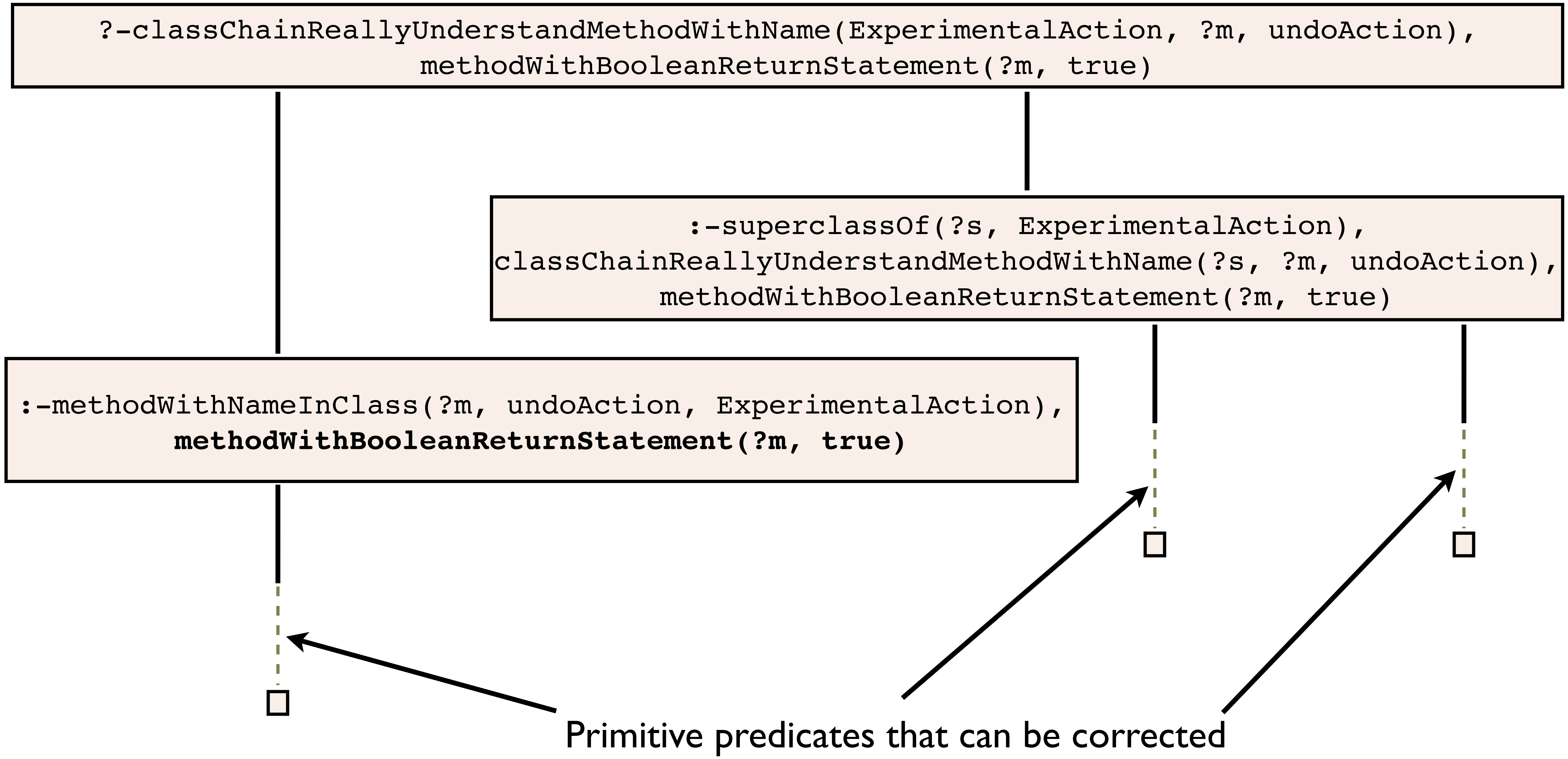
```
?-classChainReallyUnderstandMethodWithName(ExperimentalAction, ?m, undoAction),
    methodWithBooleanReturnStatement(?m, true)
```

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

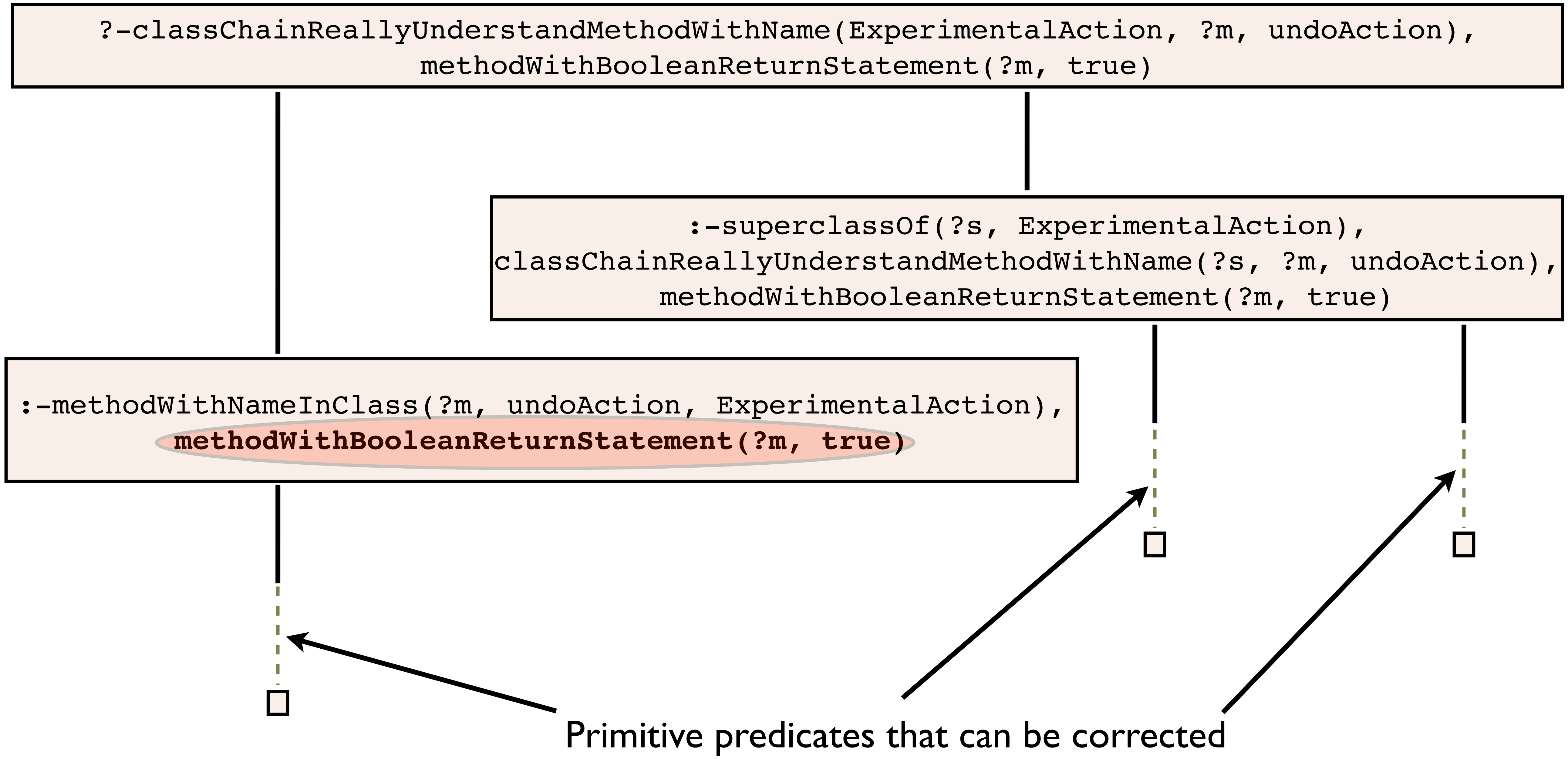
```
:-methodWithNameInClass(?m, undoAction, ExperimentalAction),
    methodWithBooleanReturnStatement(?m, true)
```

- 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


?-classChainReallyUnderstandMethodWithName(ExperimentalAction, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)
:-superclassOf(?s, ExperimentalAction),
classChainReallyUnderstandMethodWithName(?s, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)
:-methodWithNameInClass(?m, undoAction, ExperimentalAction),
methodWithBooleanReturnStatement(?m, true)
Primitive predicates that can be corrected

How to choose abducible predicates

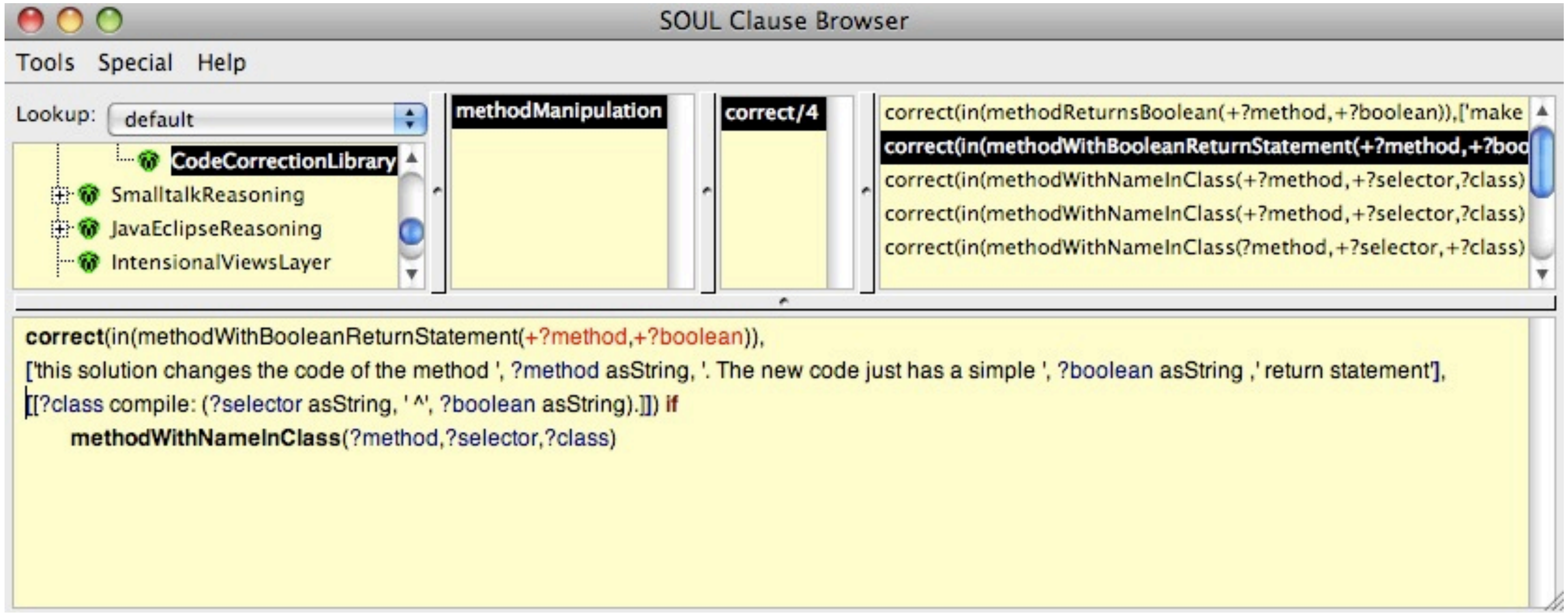

?-classChainReallyUnderstandMethodWithName(ExperimentalAction, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)
:-superclassOf(?s, ExperimentalAction),
classChainReallyUnderstandMethodWithName(?s, ?m, undoAction),
methodWithBooleanReturnStatement(?m, true)
:-methodWithNameInClass(?m, undoAction, ExperimentalAction),
methodWithBooleanReturnStatement(?m, true)
Primitive predicates that can be corrected

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*)



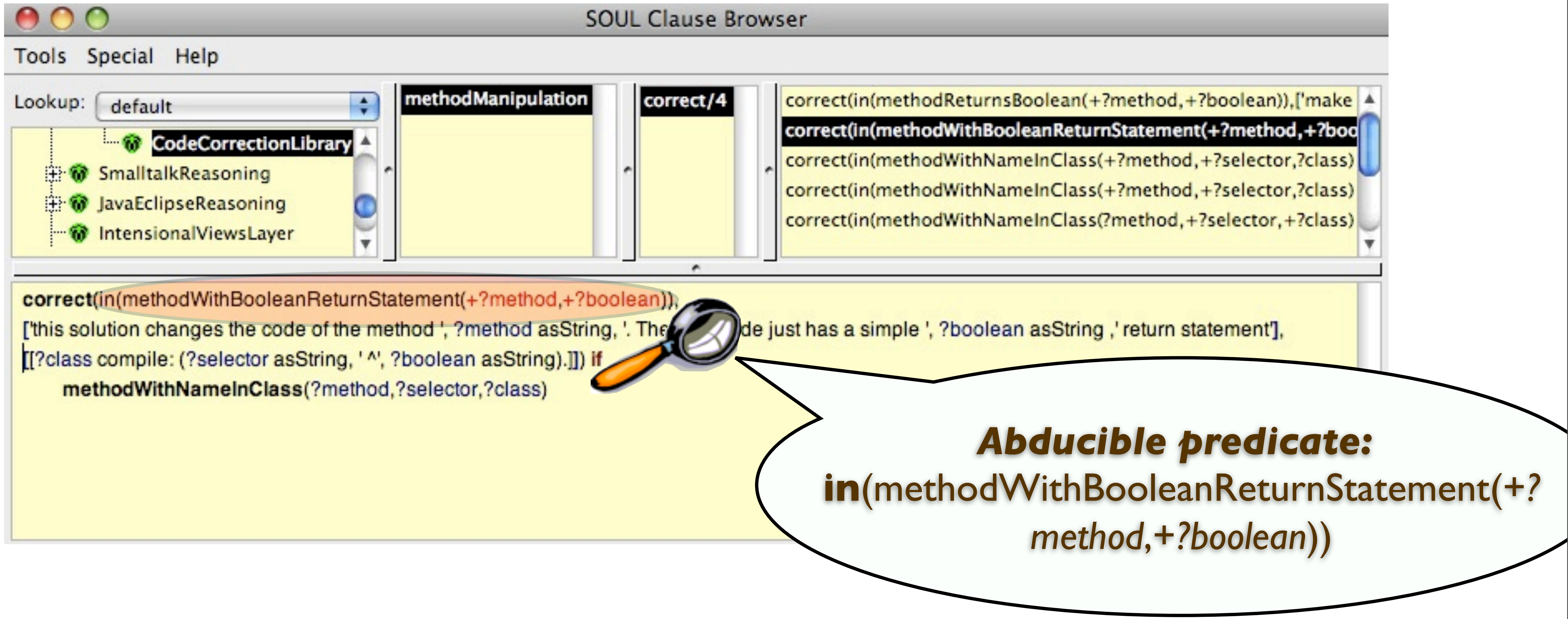
```
correct(in(methodReturnsBoolean(+?method,+?boolean)),['make
correct(in(methodWithBooleanReturnStatement(+?method,+?boo
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(?method,+?selector,+?class)
```

```
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)
```

23

Defining *positive* explanations

(*new facts will be added to the theory*)



```
correct(in(methodReturnsBoolean(+?method,+?boolean)),['make
correct(in(methodWithBooleanReturnStatement(+?method,+?boo
correct(in(methodWithNameInClass(+?method,+?selector,?class))
correct(in(methodWithNameInClass(+?method,+?selector,?class))
correct(in(methodWithNameInClass(?method,+?selector,+?class))
```

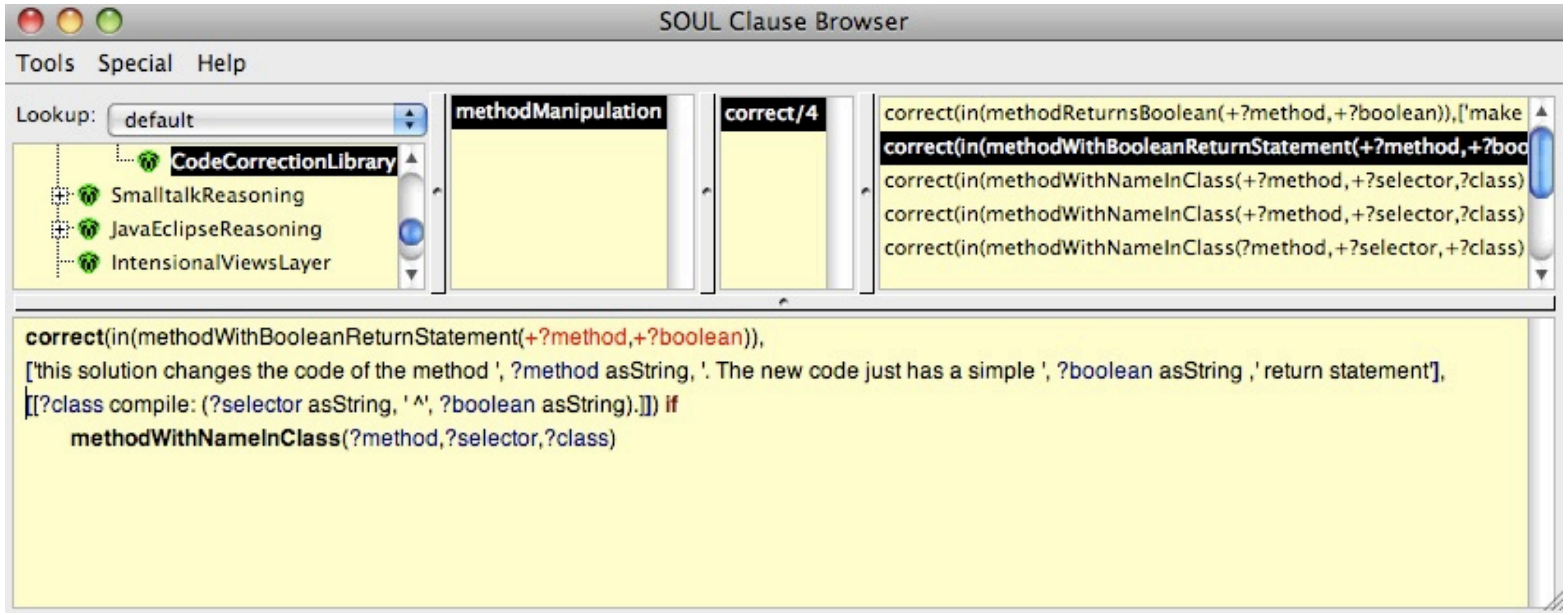
```
correct(in(methodWithBooleanReturnStatement(+?method,+?boolean)),
['this solution changes the code of the method ', ?method asString, '. The     de just has a simple ', ?boolean asString ,' return statement'],
[[?class compile: (?selector asString, ' ^', ?boolean asString).]]) if
    methodWithNameInClass(?method,?selector,?class)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

Abducible predicate:
in(methodWithBooleanReturnStatement(+?*method*,+?*boolean*))

23

Defining *positive* explanations

(*new facts will be added to the theory*)



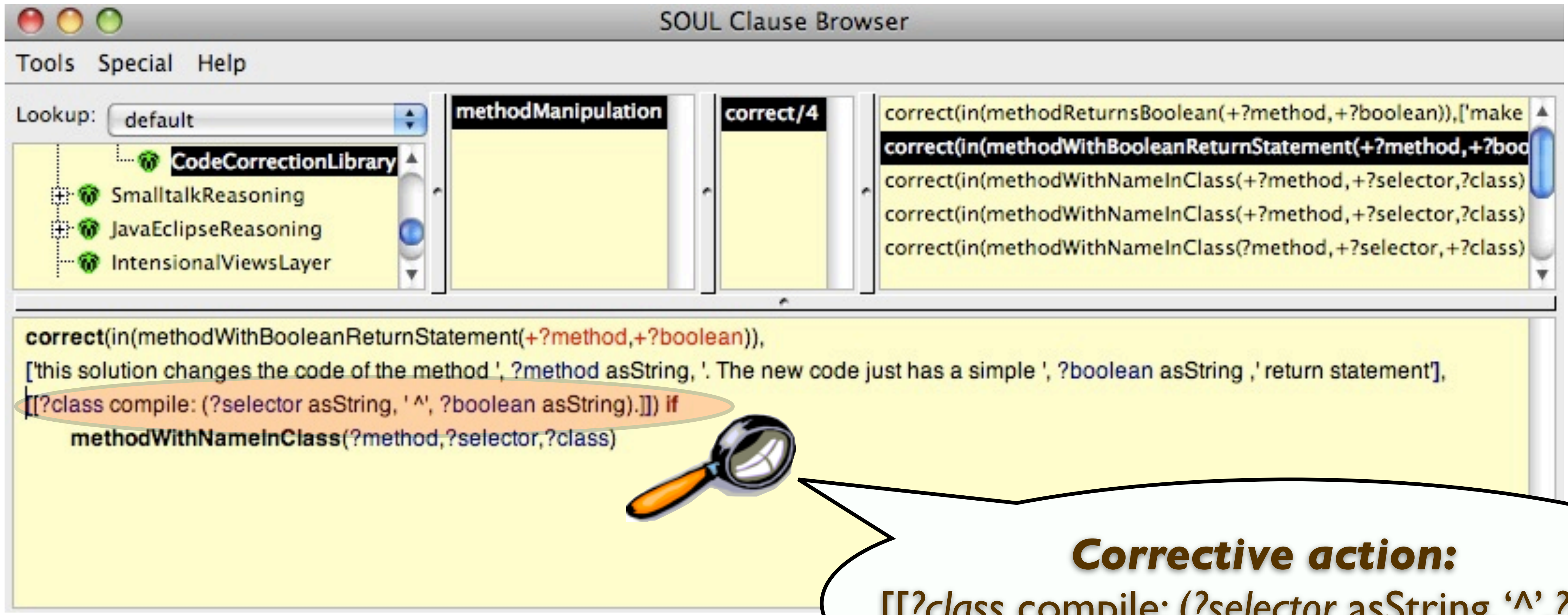
```
correct(in(methodReturnsBoolean(+?method,+?boolean)),['make
correct(in(methodWithBooleanReturnStatement(+?method,+?boo
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(?method,+?selector,+?class)
```

```
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)
```

24

Defining *positive* explanations

(*new facts will be added to the theory*)



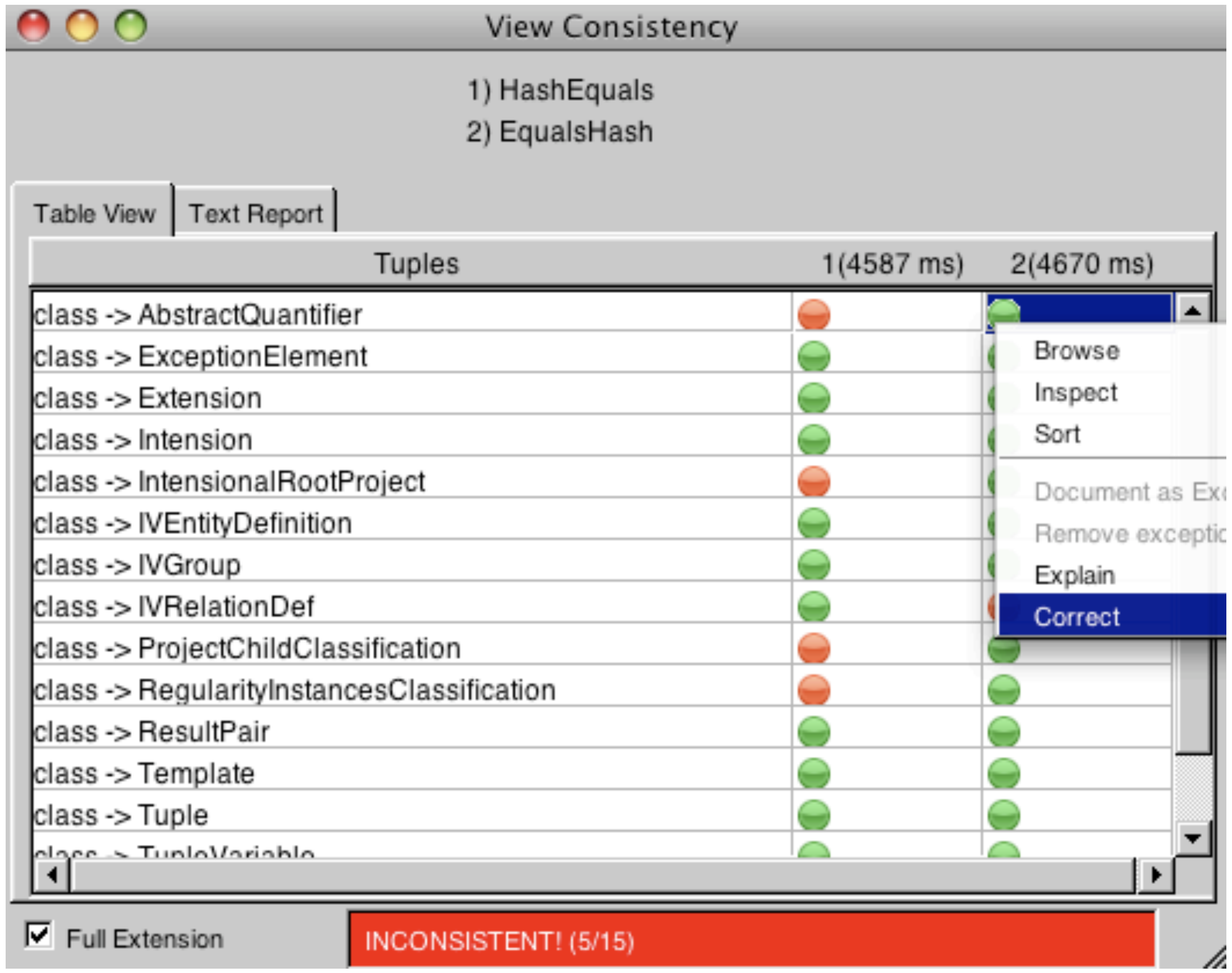
```
correct(in(methodReturnsBoolean(+?method,+?boolean)),['make
correct(in(methodWithBooleanReturnStatement(+?method,+?boo
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(+?method,+?selector,?class)
correct(in(methodWithNameInClass(?method,+?selector,+?class)
```

```
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)
```

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

24

Correcting inconsistencies

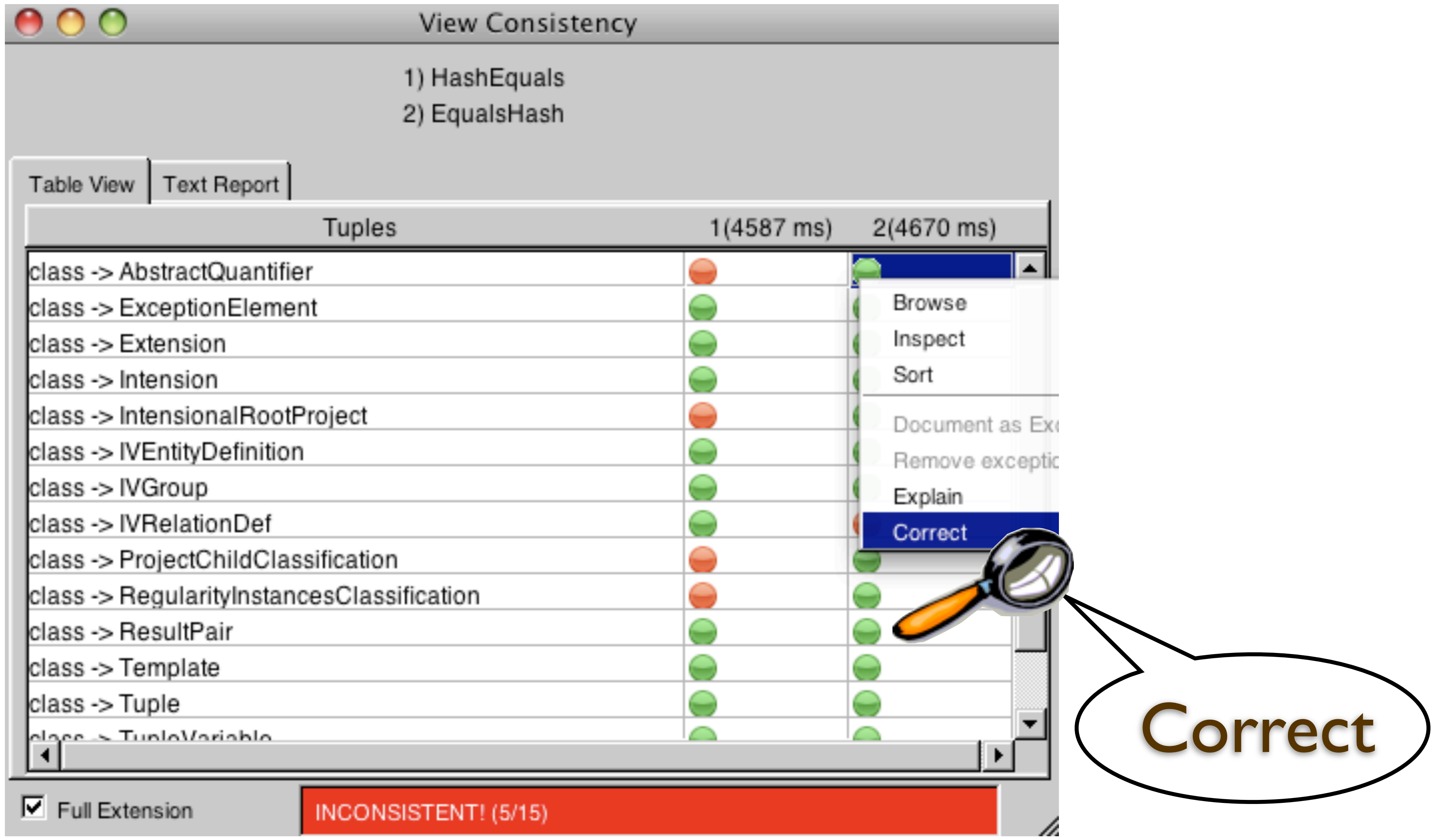


1) HashEquals
2) EqualsHash

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	🟢	🟢

INCONSISTENT! (5/15)

Correcting inconsistencies


View Consistency

1) HashEquals
2) EqualsHash

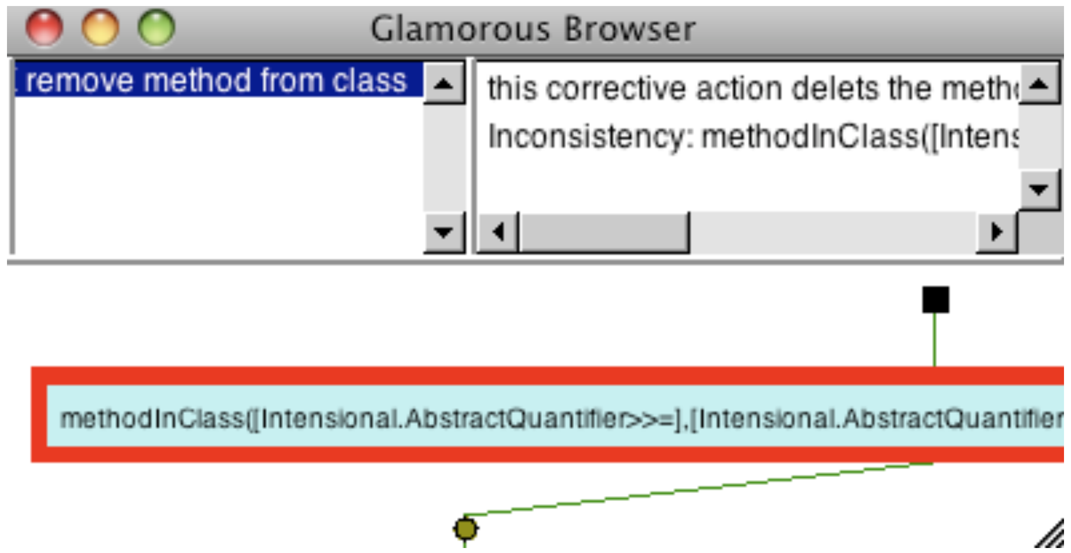
Table View | Text Report

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



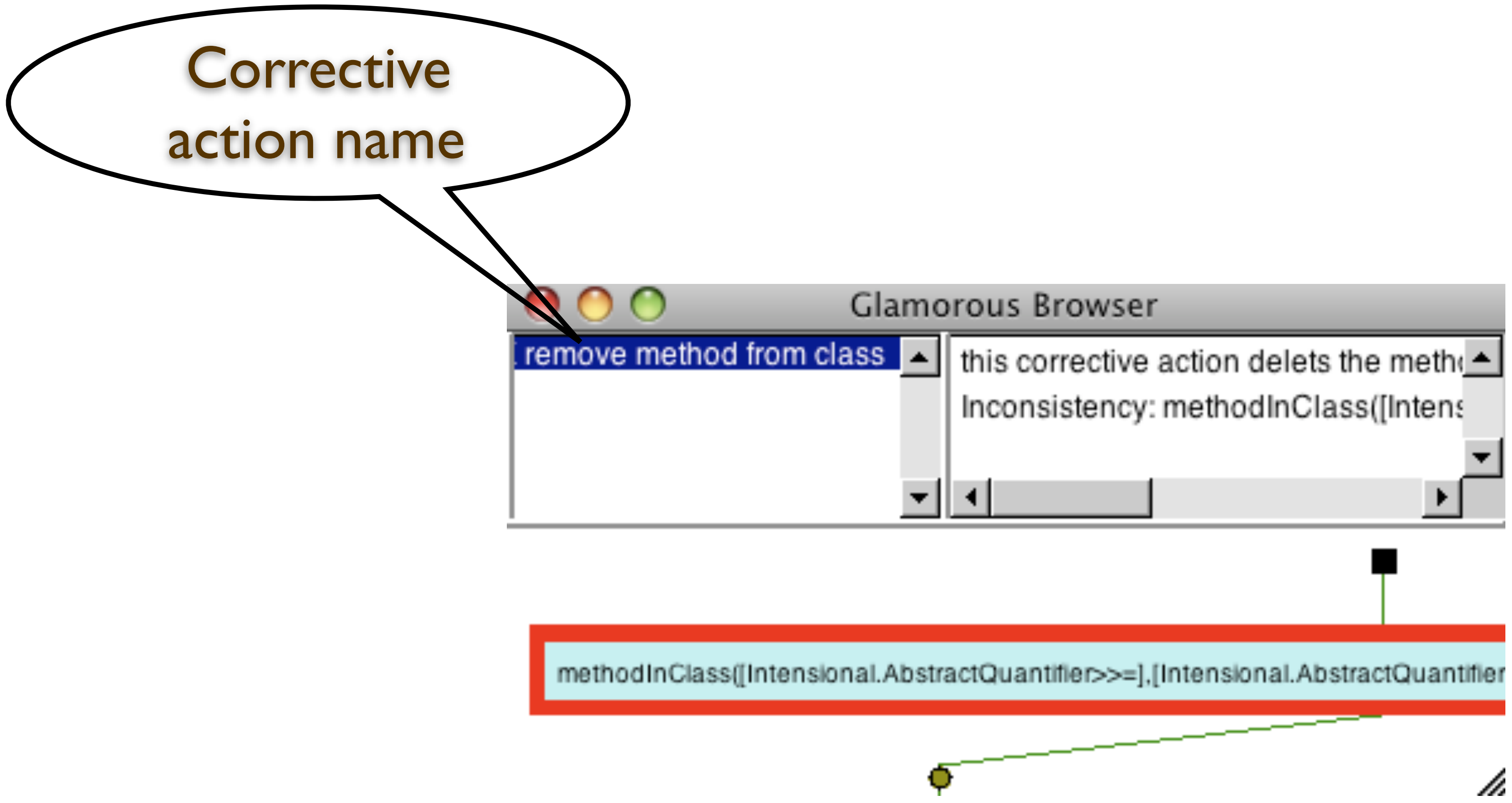
remove method from class

this corrective action delets the meth

Inconsistency: methodInClass([Intens

methodInClass([Intensional.AbstractQuantifier>>=],[Intensional.AbstractQuantifier

The corrective browser



remove method from class

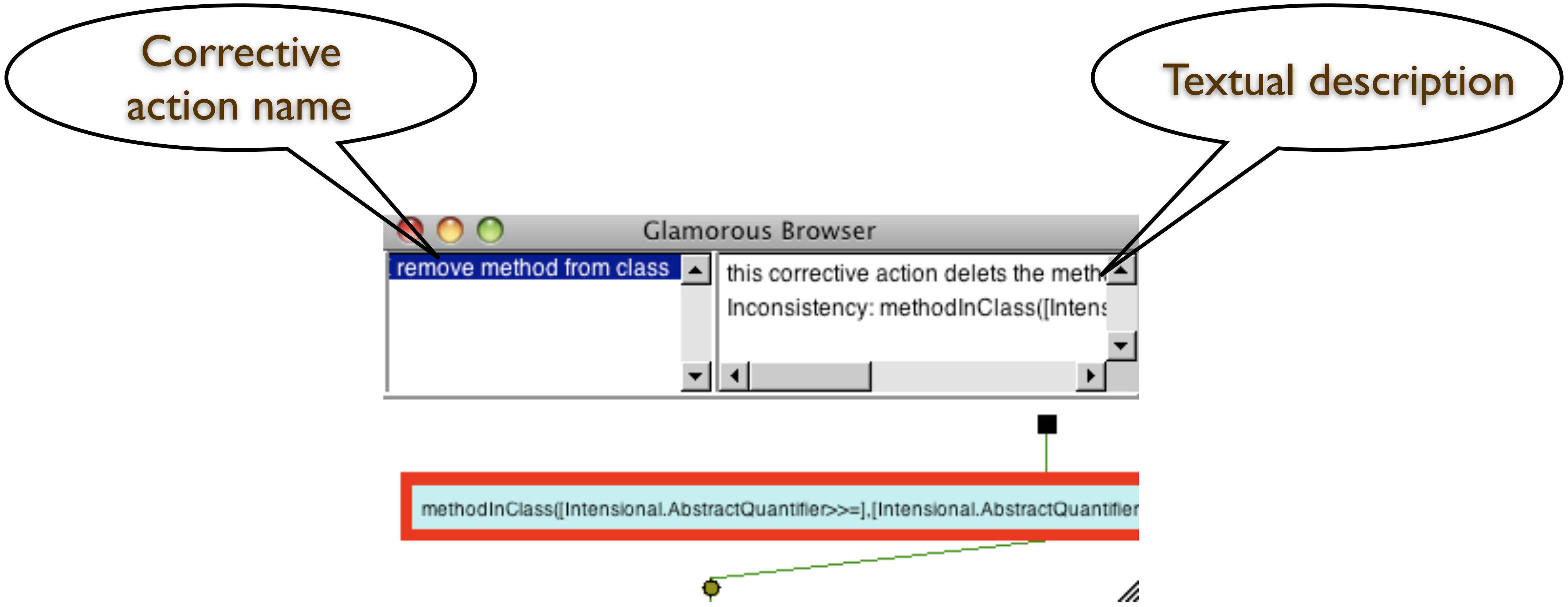
this corrective action delets the meth

Inconsistency: methodInClass([Intens

methodInClass([Intensional.AbstractQuantifier>>=],[Intensional.AbstractQuantifier

26

The corrective browser



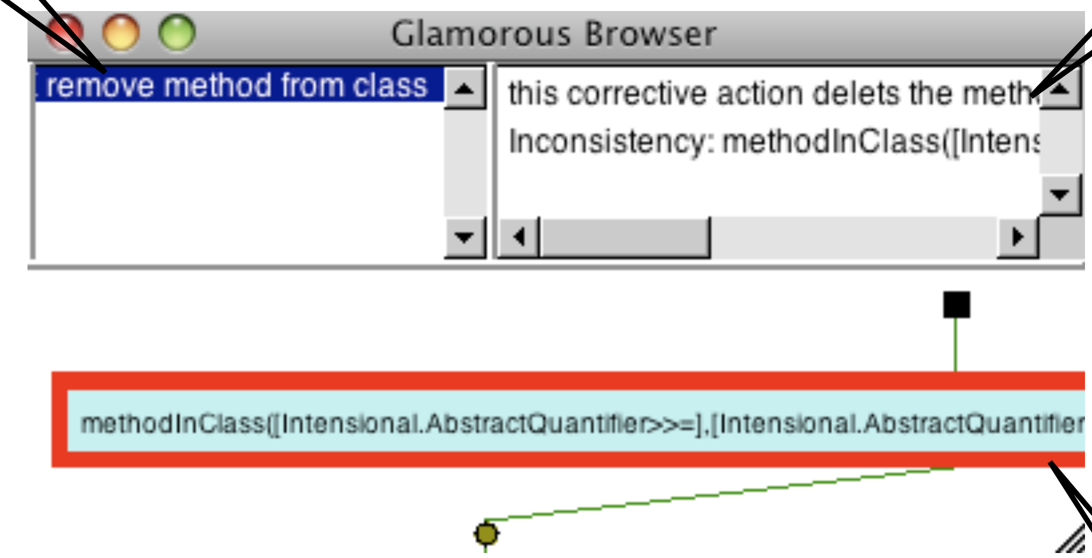
remove method from class

this corrective action delets the meth
Inconsistency: methodInClass([Intens

methodInClass([Intensional.AbstractQuantifier>>=],[Intensional.AbstractQuantifier

26

The corrective browser



remove method from class

this corrective action delets the meth
Inconsistency: methodInClass([Intens

methodInClass([Intensional.AbstractQuantifier>>=],[Intensional.AbstractQuantifier

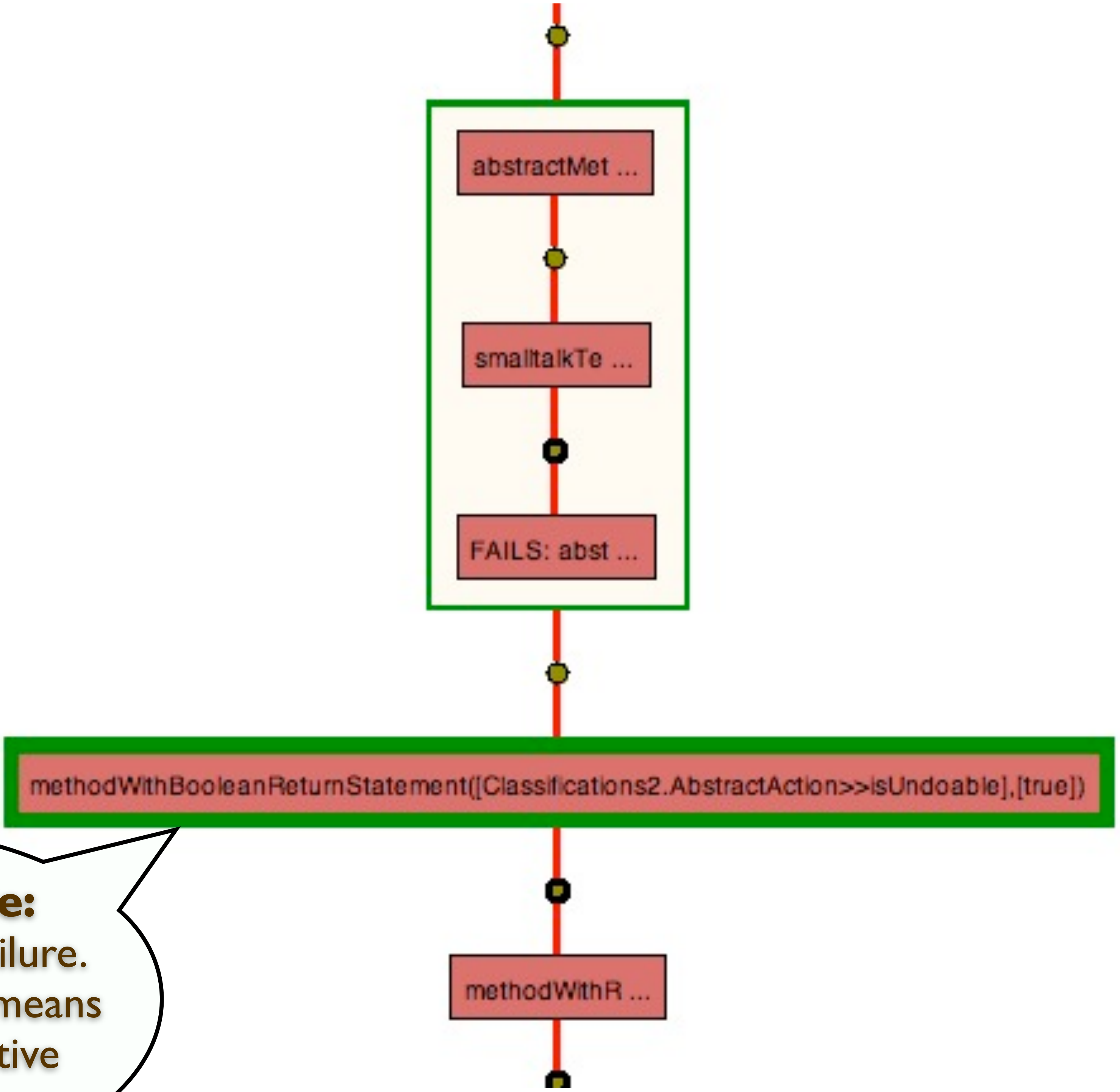
Corrective action name

Textual description

Visualization in the proof three of where the corrective action was found

26

Visualisation of the nodes with corrective actions

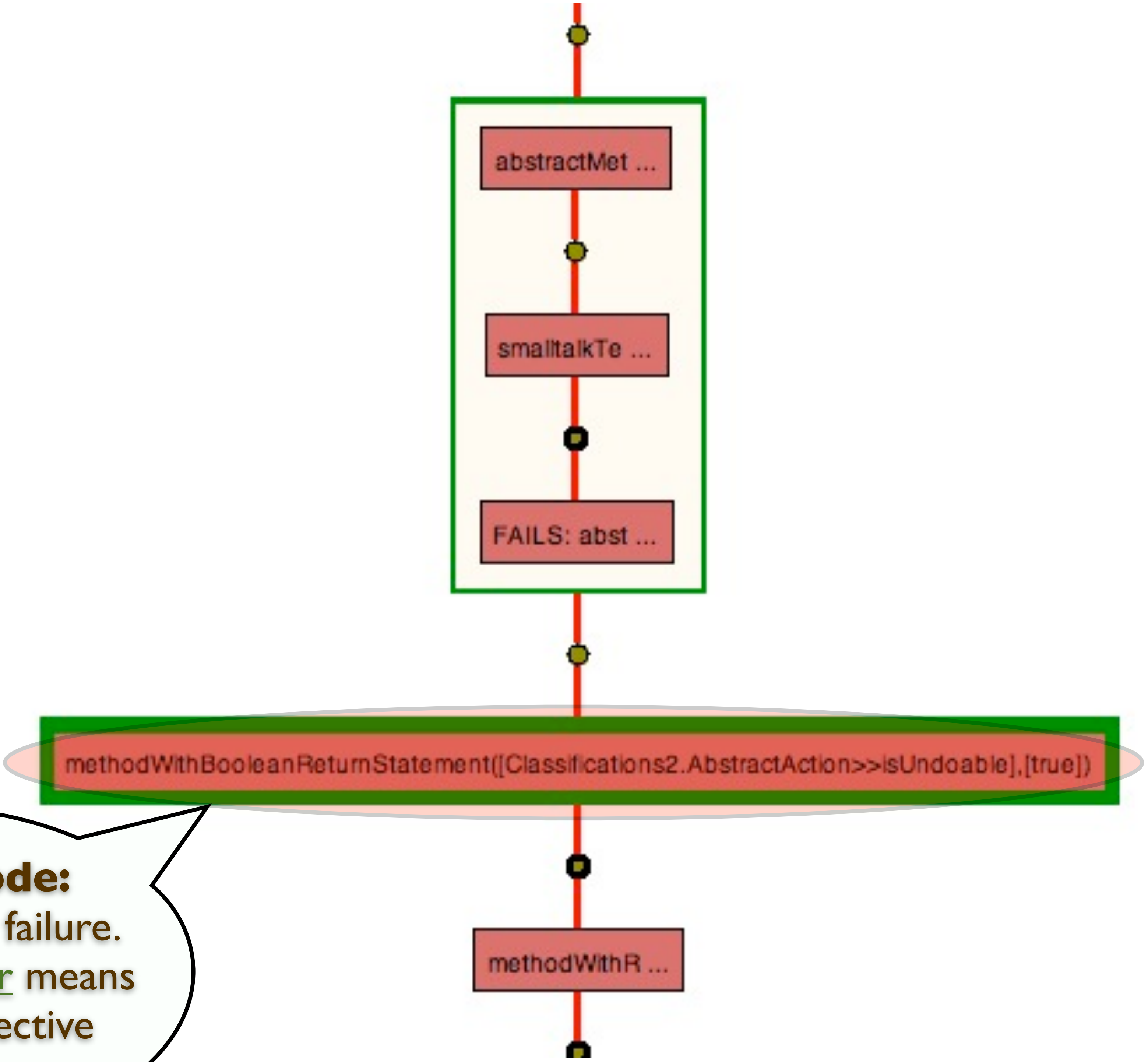

abstractMet ...
smalltalkTe ...
FAILS: abst ...
methodWithBooleanReturnStatement([Classifications2.AbstractAction>>isUndoable],[true])
methodWithR ...

Color code:

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

27

Visualisation of the nodes with corrective actions


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

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

27

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