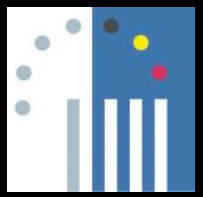


MoVES
Modelling, Verification and Evolution of Software

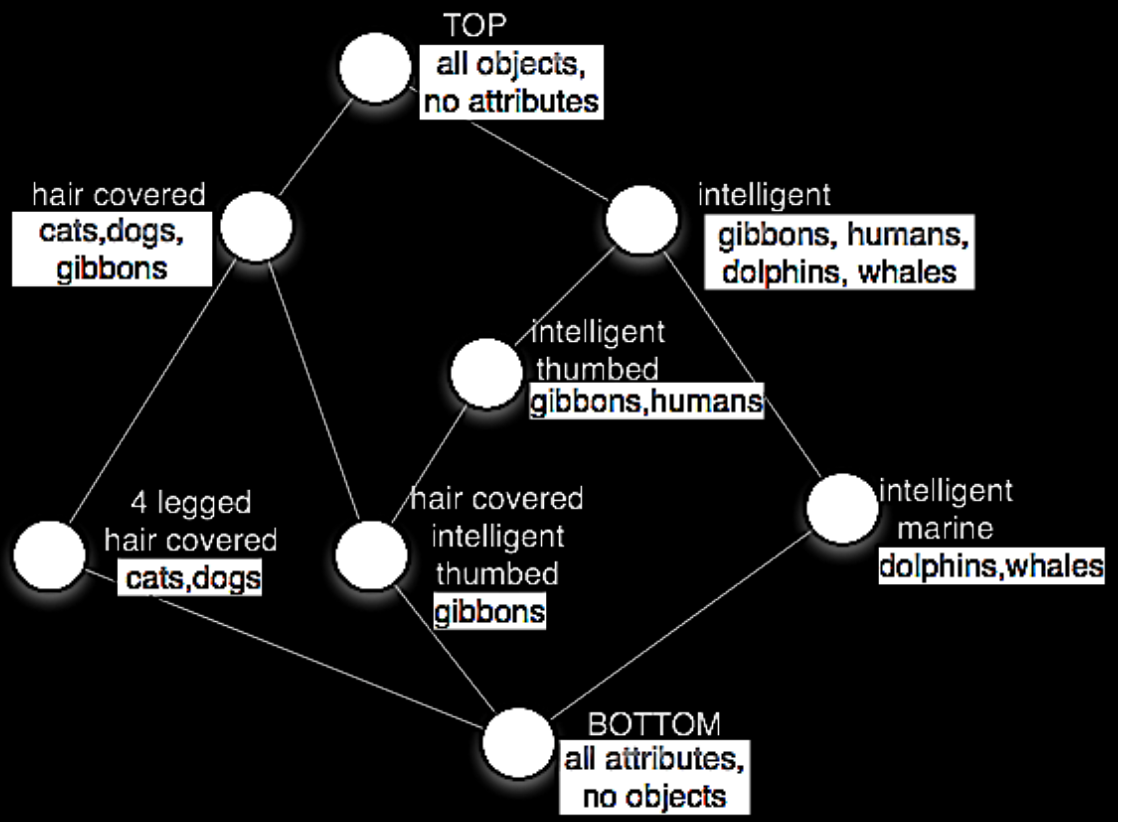
Mining source code for structural regularities

Kim Mens,
Andy Kellens,
Gabriela Arevalo,
Angela Lozano

Problem Context

- [Context]:
  - Programmers often use regularities: coding conventions, design patterns, crosscutting concerns...
  - Enforcing such regularities facilitates maintenance, evolution & comprehension
  - Regularities are often not fully respected (implicit, no support)
- [Goal]:
  - provide automated (tool) support to discover source code (ir)regularities
    - Document them & validate their evolution with IntensiVE (A. Kellens)
    - Fix inconsistencies in regularities with HEALER (S. Castro)
    - Guide maintenance & evolution tasks (future work)

Characterizing Mammals		ATTRIBUTES				
		4 legged	hair covered	intelligent	marine	thumbed
Objects	cats	*	*			
	dogs	*	*			
	dolphins			*	*	
	gibbons		*	*		*
	humans			*		*
	whales			*	*	



Formal Concept Analysis FCA

FCA: Lessons learnt from mining aspects

- DISADVANTAGES
  - almost combinatorial amount of results
    - redundancy
  - requires traversal heuristics (ad-hoc)
  - does not detect exceptional cases
- ADVANTAGES
  - description of the concept
    - shared properties = hint of concept specification
  - does not require a-priori knowledge

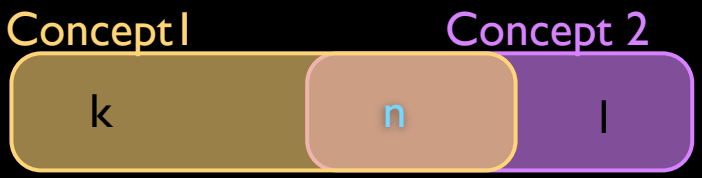
Rules Notation

Concept1 (k) --n m% --> (l) Concept2

should be read as:

- n elements in Concept1 also appear in Concept2
- m% of the elements in Concept1 (a.k.a. confidence = n/n+k) are also in Concept2
  - k elements in Concept1 are NOT in Concept2
  - l elements in Concept2 are NOT in Concept1

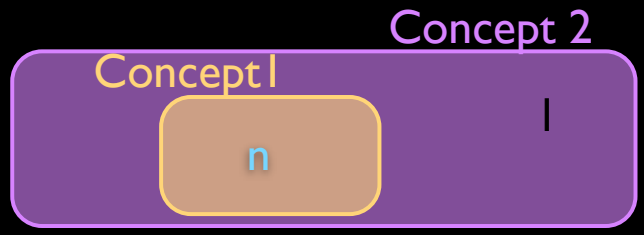
Visually:



- Special case: when m = 100%

Concept1 (0) ==n 100% ==> (l) Concept2

Visually:

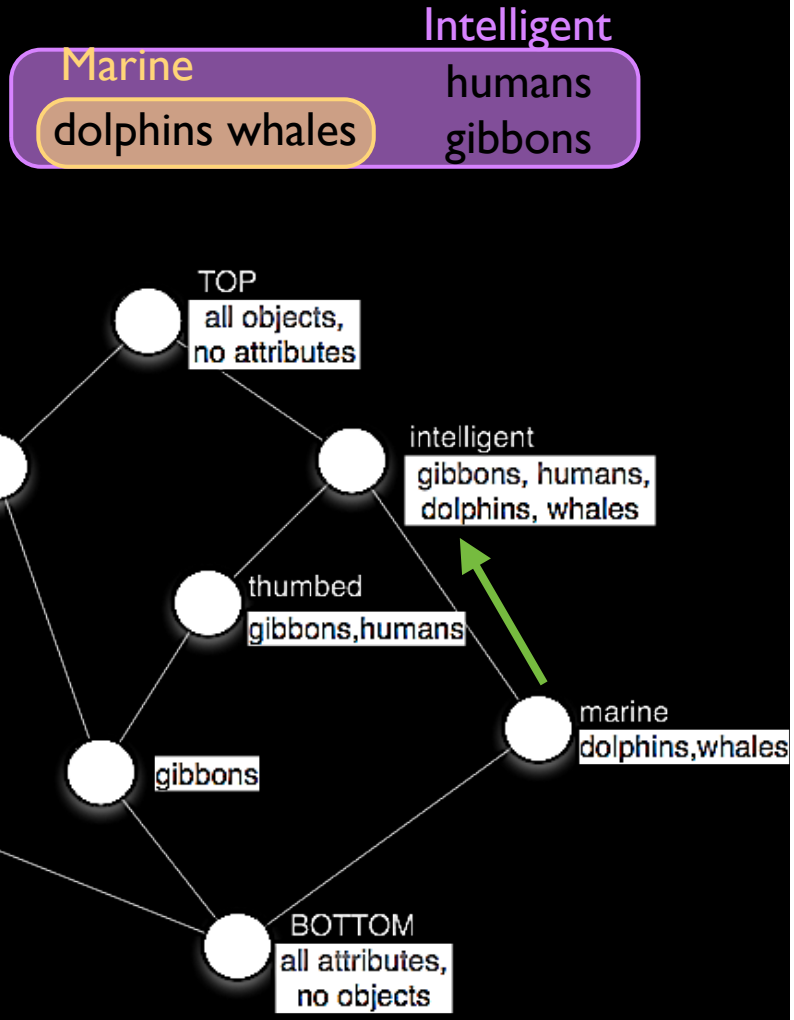


Implications

- Marine (0) ==2 (100%)==> (2) Intelligent
- Marine mammals (closed world)
- Subset relation in lattice

Asssociations

- Hair covered (1) --2 (66%)--> (0) 4 legged
- Most hair-covered mammals are 4 legged; gibbons aren't
- Superset or siblings in the lattice

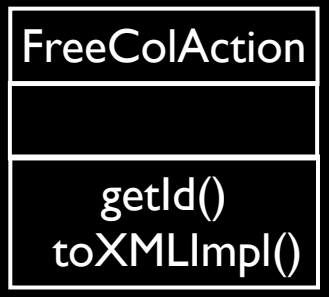


CASES

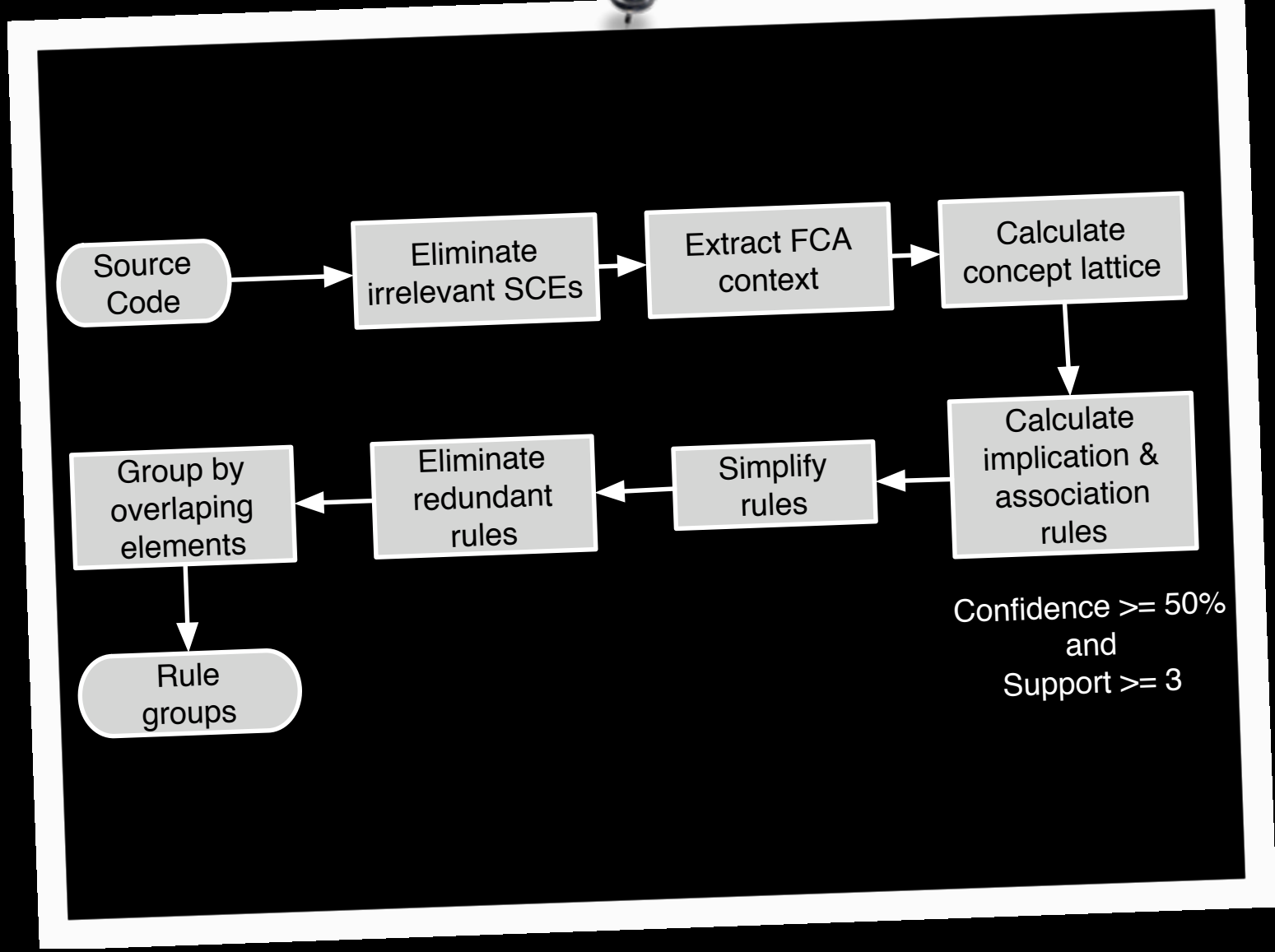
- Case 1 : IntensiVE
  - Tool to validate if regularities documented through several views are respected
  - Smalltalk
  - 270 classes; 2729 methods
- Case 2 : Freecol
  - Colonization game (graphic, multiplayer)
  - Java
  - 382 classes; 3252 methods

OUR Approach

- Objects = Classes, & Attributes =
  - (K) has keyword (in class name)
  - (I) implements a particular method / message
  - (H) in hierarchy of class

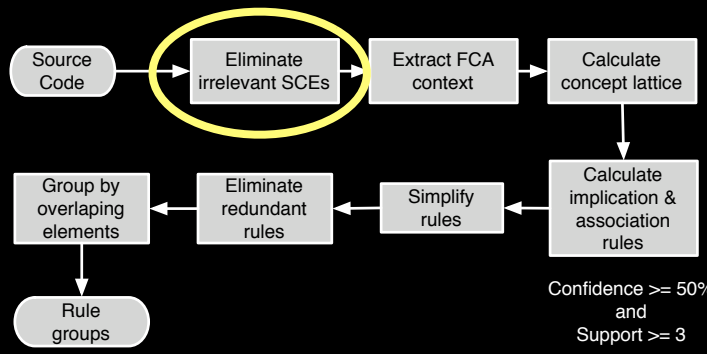


- E.g.
  - FreeColAction:
  - K>Action, H>FreeColAction, I>getId, I>toXMLImpl


Source Code
Eliminate irrelevant SCEs
Extract FCA context
Calculate concept lattice
Calculate implication & association rules
Simplify rules
Eliminate redundant rules
Group by overlaping elements
Rule groups
Confidence >= 50%
and
Support >= 3

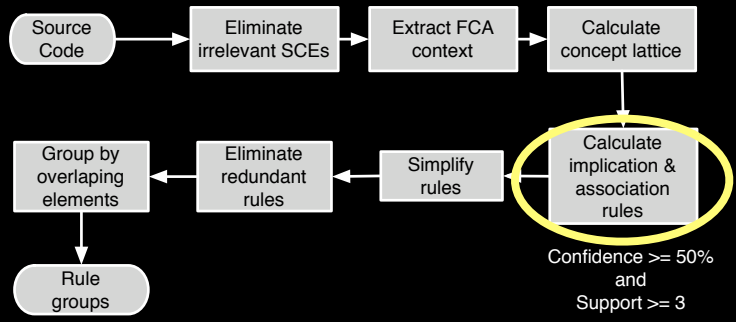
algorithm

eliminate SCEs



- Irrelevant entities:
  - Object class
  - Test classes & methods

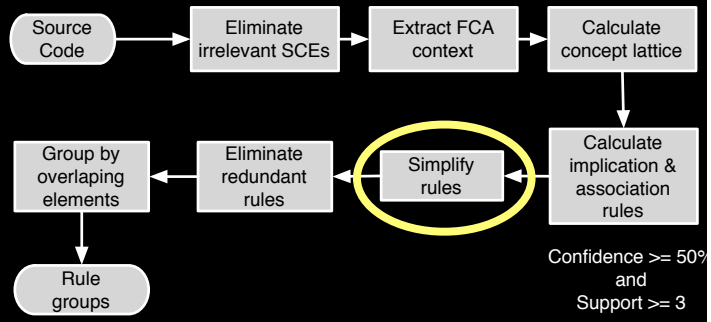
rule calculation



- Calculate implications:
  - traverse child-parent relations of key nodes*
- Calculate associations:
  - traverse all parent-child relation of key nodes*
  - traverse all key nodes* pairs that have no connection in the lattice
- Filter relations with confidence & support below thresholds
  - confidence $\leq$ 75% and support $\geq$ 3

* those that add attributes or objects to the lattice

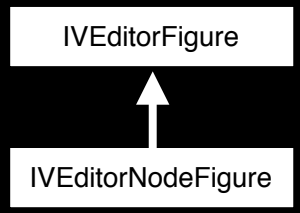
simplify rules



GOAL: Eliminate redundant properties of a rule

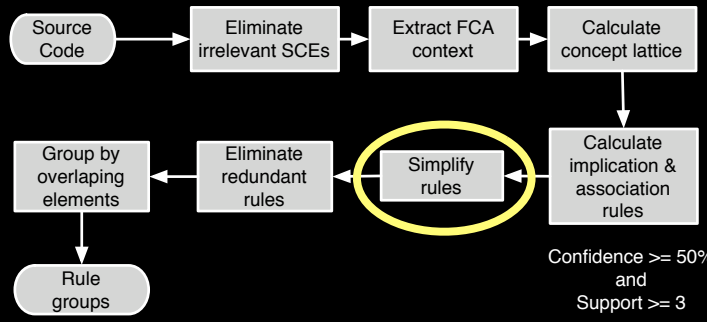
H>IVEditorNodeFigure ~~H>IVEditorFigure~~ ----> K>'IVEditor' K>'Figure'

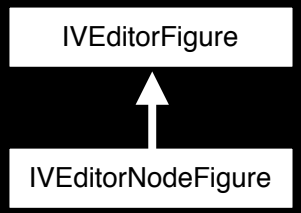
H>IVEditorNodeFigure ====> H>IVEditorFigure



because IVEditorFigure does not add information to the rule

simplify rules





H>IVEditorNodeFigure ==100%==> H>IVEditorFigure

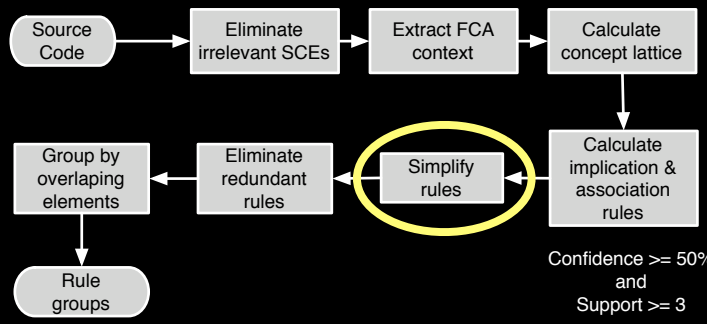
H>IVEditorNodeFigure, H>IVEditorFigure ----> K>'IVEditor', K>'Figure'

K>'IVEditor', K>'Figure' ----> H>IVEditorNodeFigure, H>IVEditorFigure

K>'IVEditor', K>'Figure', H>IVEditorFigure ----> H>IVEditorNodeFigure

K>'IVEditor', K>'Figure', H>IVEditorNodeFigure ----> H>IVEditorFigure

H>Intensional.IVIntensiVEAction (0) ==24 (100%)==>
(0) I>#undoAction,H>Classifications2.AbstractAction,I>#performAction

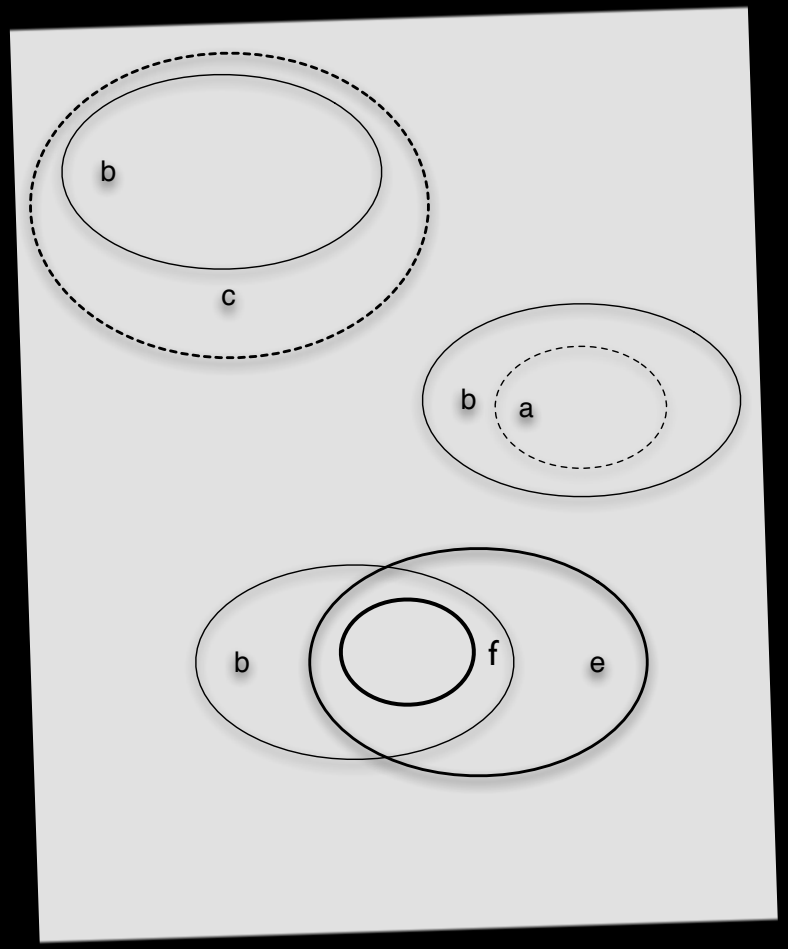


simplify rules:

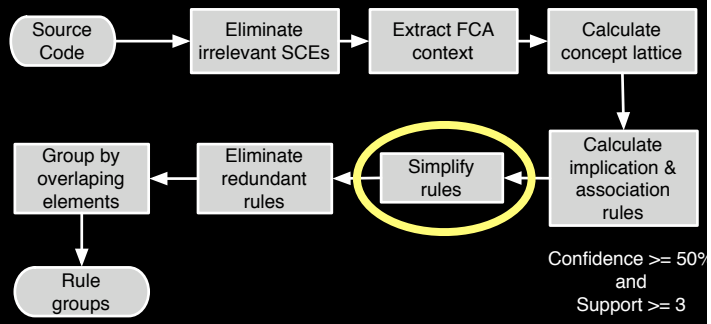
Priority to apply implications

Suppose

(R) a,b,c,e,f → h



Is there an order to apply several implications to remove redundancies from R?



simplify rules:

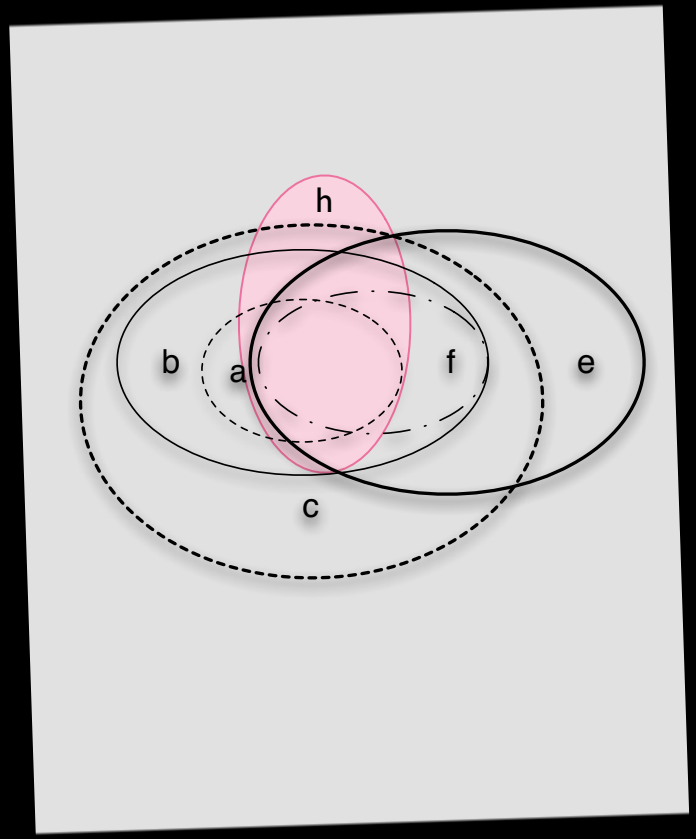
Priority to apply implications

(X) b ⇒ c

(Y) a ⇒ b

(Z) f ⇒ b,e

(R) a,b,c,e,f → h



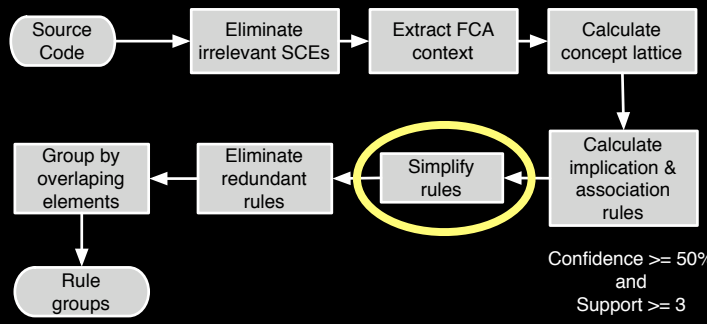
Y, Z, X (R) a,b,c,e,f → h

Z, Y, X (R) a,b,c,e,f → h

X, Z, Y (R) a,b,c,e,f → h

simplify rules:

Priority to apply implications



$A \leq B \Leftrightarrow$ A.condition $\subseteq$ B.conclusion

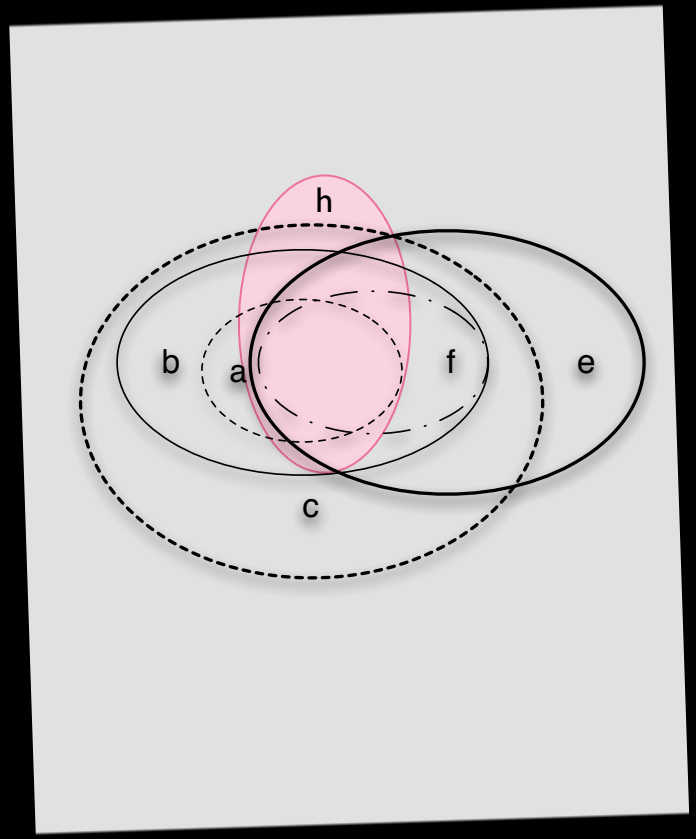
$\vee$ B. conclusion $\subseteq$ A. conclusion

(X) b ⇒ c

(Y) a ⇒ b

(Z) f ⇒ b,e

(R) a,b,c,e,f → h



X 1st:

X≤Z because b ⊆ b,e (X.condition ⊆ Z.conclusion)

X≤Y because b ⊆ b (X.condition ⊆ Y.conclusion)

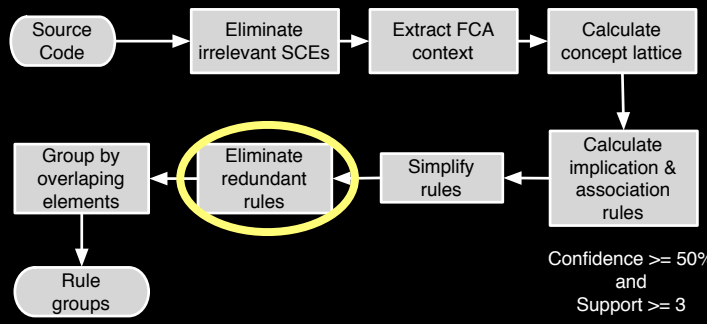
applying X: (XtoR) a,b ,e,f → h

Z 2nd:

Z≤Y because b ⊆ b+e (Y. conclusion ⊆ Z. conclusion)

applying Z: (Zto(XtoR)) a ,f → h

eliminate rules



- Eliminate unrelated sets

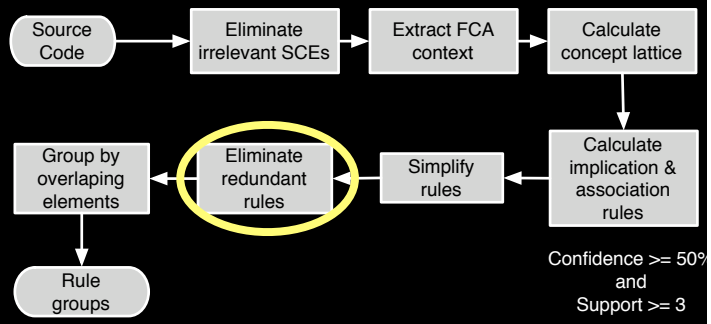
K>'Colopedia' (0) ==7 (100%)==> (86) I>'actionPerformed'

a (exc. condition) --(matches)→ (exc. conclusion) b

i.e. the average number of exceptions is below a quarter of any of the sets

$$\frac{\frac{(\text{exc. condition})}{(\text{condition size})} + \frac{(\text{exc. conclusion})}{(\text{conclusion size})}}{2} \leqslant 0.25$$

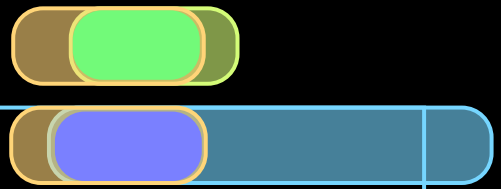
eliminate rules



- Similar rules that conclude SubClass or SuperClass.

Property>X -- matchesSub--> (exc. conclusion sub) H>SubClass

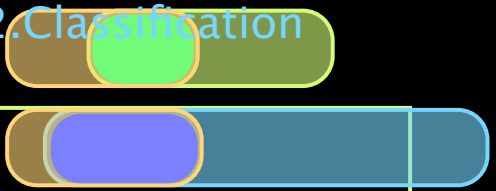
Property>X -- matchesSuper--> (exc. conclusion super) H>SuperClass



eliminate the super rule if it just adds noise (~ matches, ++ exceptions):

$$\frac{\frac{(\text{matchesSub})}{(\text{condition size})}}{\frac{(\text{matchesSuper})}{(\text{condition size})}} \geqslant 0.9 \quad \text{and} \quad \frac{(\text{exc. conclusion sub})}{(\text{exc. conclusion super})} \leqslant 0.9$$

deleted K>'Classification' (0) --5 (100%)--> (24) H>Classifications2.AbstractClassification
K>'Classification' (0) --5 (100%)--> (0) H>Classifications2.Classification

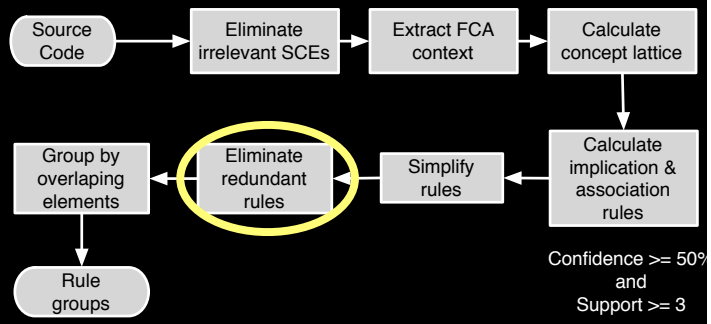


eliminate the sub rule if it has lower confidence:

$$\frac{(\text{matchesSub})}{(\text{condition size})} \leqslant \frac{(\text{matchesSuper})}{(\text{condition size})}$$

I>'shouldBeEnabled' (0) --46 (100%)--> (7) H>'FreeColAction'
deleted I>'shouldBeEnabled' (2) --44 (96%)--> (7) H>'MapboardAction'

eliminate rules



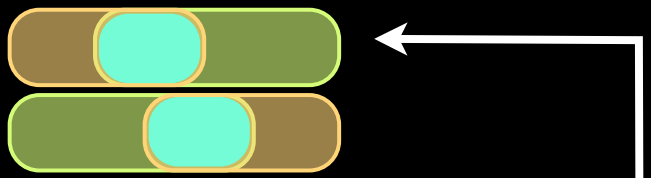
- Conclude the root of the classes in the app. does not add any information:

  Property>X ----> H>RootClass

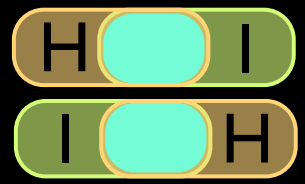
- When having converse pairs of rules,

  Property>X ----> Property>Y

  Property>Y ----> Property>X

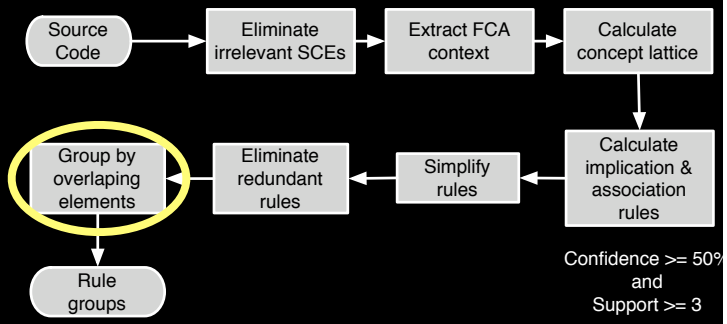
  

  - prefer the one with better confidence.
  - If similar confidence, prefer the one that starts with a the condition of **stronger semantics**.

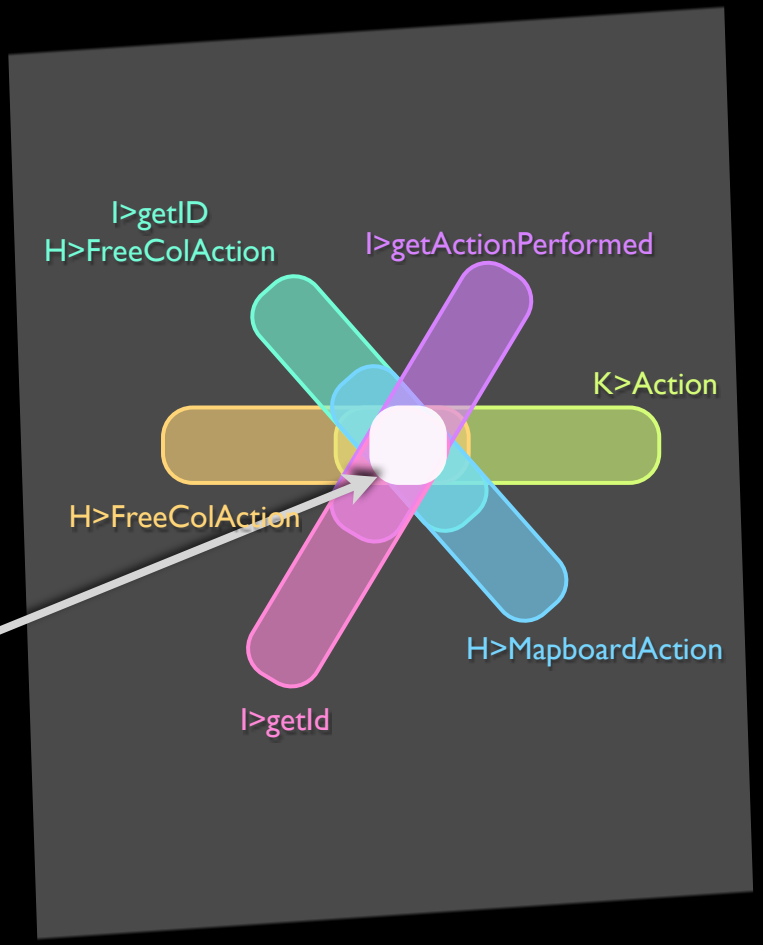
    

    H > I, H > K, I=K

group rules



- Those rules that share at least 85% of their matches are be grouped together
  - threshold comes from analysis of results (might change depending on the case study)
- They represent common properties of a set of source code entities
- These groups are ordered by number of matches



RESULTS:
Reduction of information to process

- IntensiVE [270 classes; 2729 methods]
  - Concepts: 1289 / Relations: 4390
  - Rules: 325 / Groups: 50

- Freecol [382 classes; 3252 methods]
  - Concepts: 1261 / Relations: 5149
  - Rules: 134 / Groups: 42

RESULTS: Freecol (rules)

- K → H = 2 rules
  - The concept described by the keyword is confined to classes in the hierarchy K>'Mission'→ H>'Mission'
- K → K = 4 rules
  - Combined words Free+Col, Trade+Route, Free+Col+Menu, etc.
- K → I = 8 rules
  - Classes named *Keyword* should implement the method I

  e.g. K>'Info'→ I>'update', K>'Action'→ I>'actionPerformed', K>'Thread'→ I>'run', K>'Mission'→ I>'doMission', etc.

RESULTS: Freecol (rules)

- H → K = 9 rules
  - Classes in the hierarchy can be described by the keyword. e.g.
  - H>ReportPanel → K>'Panel' K>'Report'
  - H>'NetworkRequestHandler'→ K>'Handler'
  - H>'InputHandler'→ K>'Input'
  - H>'OptionUpdater'→ K>'UI'
  - 'TradeItem'→ K>'Item'

RESULTS: Freecol (rules)

- H->I = 28 rules
  - Classes in the hierarchy should implement the method. e.g.
  - H>'NetworkRequestHandler'→ I>'handle'
  - H>'OptionMap' → I>'addDefaultOptions'
  - H>'Location' → I>'getGoodsContainer'→ I>'getLocationName'
  - H>'MapIterator' → I>'nextPosition'
  - H>'MapboardAction' → I>'getId' → I>'shouldBeEnabled'
  - H>'OptionUpdater' → I>'updateOption',I>'unregister'
  - H>'TradeItem' → I>'makeTrade'
  - H>'PersistentObject'→ I>'readFromXMLImpl' → I>'toXML'

RESULTS: Freecol (rules)

- I->I = 71 rules
  - Implementation protocols. e.g.
  - I>'getColony'→ I>'getXMLElementTagName'→ I>'toXMLImpl' →I>'readFromXMLImpl'
  - I>'installUI' → I>'createUI'
  - I>'getTransportDestination'→ I>'doMission' →I>'dispose'
  - I>'contains' →I>'add' →I>'newTurn' →I>'remove'
  - I>'toXML' →I>'getXMLElementTagName' →I>'readFromXMLImpl'
  - I>'requestFocus' → I>'actionPerformed' →I>'initialize'
  - I>'setOwner' → I>'newTurn' → I>'getTile'
  - I>'setName'→ I>'getName'

RESULTS: Freecol (groups)

- Classes in the hierarchy FreeColAction are named *Action*, and tend to implement getId and actionPerformed
- 50 matches, 6 rules
  - H>'MapboardAsction' (1) --50 (98%)----> (3) I>'getId'
  - H>'FreeColAction' (0) ==53 (100%)==> (3) K>'Action'
  - H>'FreeColAction' (3) --50 (94%)--> (43) I>'actionPerformed'
  - I>'getId' (3) --50 (94%)--> (43) I>'actionPerformed'
  - I>'getId',H>'FreeColAction' (0) --52 (100%)--> (4) K>'Action'
  - K>'Action' (5) --51 (91%)--> (42) I>'actionPerformed'

RESULTS: Freecol (groups)

- Most of the classes that implement initialize belong to the hierarchy of FreeColPanel
  - initialize prepares a panel to be displayed
  - 36 matches, 1 rules
    - I>'initialize' (7) --36 (84%)--> (9) H>'FreeColPanel'

- Classes that implement toXMLImpl also implement getXMLElementTagName
  - toXMLImpl writes an XML representation of the object to a stream.
  - getXMLElementTagName gets the tag name that represents the object
  - Exception is FreeColAction, which is the XML root
  - 44 matches, 1 rules
    - I>'toXMLImpl' (1) --44 (98%)--> (16) I>'getXMLElementTagName'

RESULTS: IntensiVE

- Regularities documented & found
  - Interface
    - Action protocol & undoable protocol
    - Compilation
    - Relation evaluators
    - Cache / Save / Remove on definitions
    - Intension Editors (partial protocol)
    - Instantiable views
    - Constraint editors
    - Evaluators
  - Naming convention
    - Unit testing, View hierarchy
  - Interface + Naming convention
    - Quantifiers (naming + partial interface)

RESULTS: IntensiVE

- Regularities found & NOT documented
  - Interfaces
    - IntensiVE Explorer Visualization
    - Checkable entities
    - Fuzzy quantifiers
    - Query generation for visual querying
    - Context-menu in visual query language
    - Figure rendering
    - Special classifications
  - Naming conventions
    - Figures, Exceptions, Visualization, Classifications, Exceptions to views, Result pairs, Reporters.
  - Interfaces + Naming conventions
    - Starbrowser shells

CONCLUSIONS

- Use FCA with objects = source code entities
- As attributes = several types of properties
- Calculate implications
  - to mine for intension of regularity rather than extension
  - not just entities that match regularity but explicit specification of regularity
- Allow for variations and irregularities = association rules
- To overcome previous pitfalls and make regularities explicit

THREATS & LIMITATIONS

- Redundant information
  - There are groups that are sub-sets of other groups
- All results are correct but....
  - some regularities found might be due to chance and not as conscious development decisions
  - interpretation of results require to assume a close world
- Usefulness is subjective
  - i.e. separating useful from useless results
- Data analyzed could be more of semantic

CURRENT & FUTURE WORK:

- ...Running the same case studies mixing the results of Classes and Methods
- Calculate which percentage of the irregularities of a group are indeed an error
- Use the results to guide the developer while adding or modifying SCEs
- Use a similar approach to mine feature dependencies