Extension, Abbreviation and Refinement

-Identifying High-Level Dependence Structures Using Slice-Based Dependence Analysis

Zheng Li

CREST, King’s College London, UK

Centre for Research in Evolution,
Search & Testing

Overview

- Motivation
- Three combination techniques
  - Extension
  - Abbreviation
  - Refinement

Many analysis techniques for program comprehension have been proposed

Domain knowledge

high-level

Pattern recognition
Concept assignment

Source code

low-level

Data-flow analysis
Dependence analysis

Advantages and Disadvantages

	High-level	Low-level
Accuracy	Low	High
Scalability	Yes	No
Human Knowledge	Yes	No

If combine the two?

- High-level techniques can provide a reasonable analysis scope with domain knowledge for low-level analysis techniques, then avoiding the scalability problem of low-level techniques.

- Low-level techniques can improve the accuracy of high-level techniques.

In this thesis

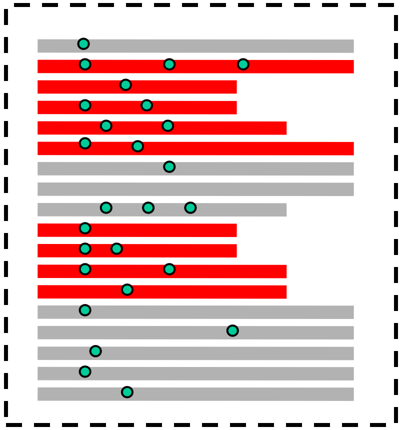
Concept Assignment

Program Slicing

Centre for Research in Evolution, Search & Testing

Concept Assignment

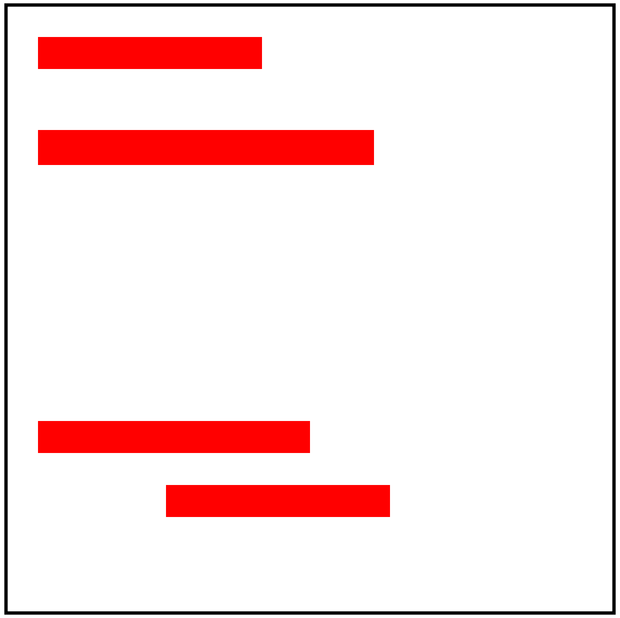
- First defined in 1993 and aimed at comprehension tasks
- allocate specific high-level meaning to specific parts of a program
- Hypothesis-Based Concept Assignment (HB-CA)
  - Existing implementation
  - Uses domain and program semantics
  - Good quality assignments



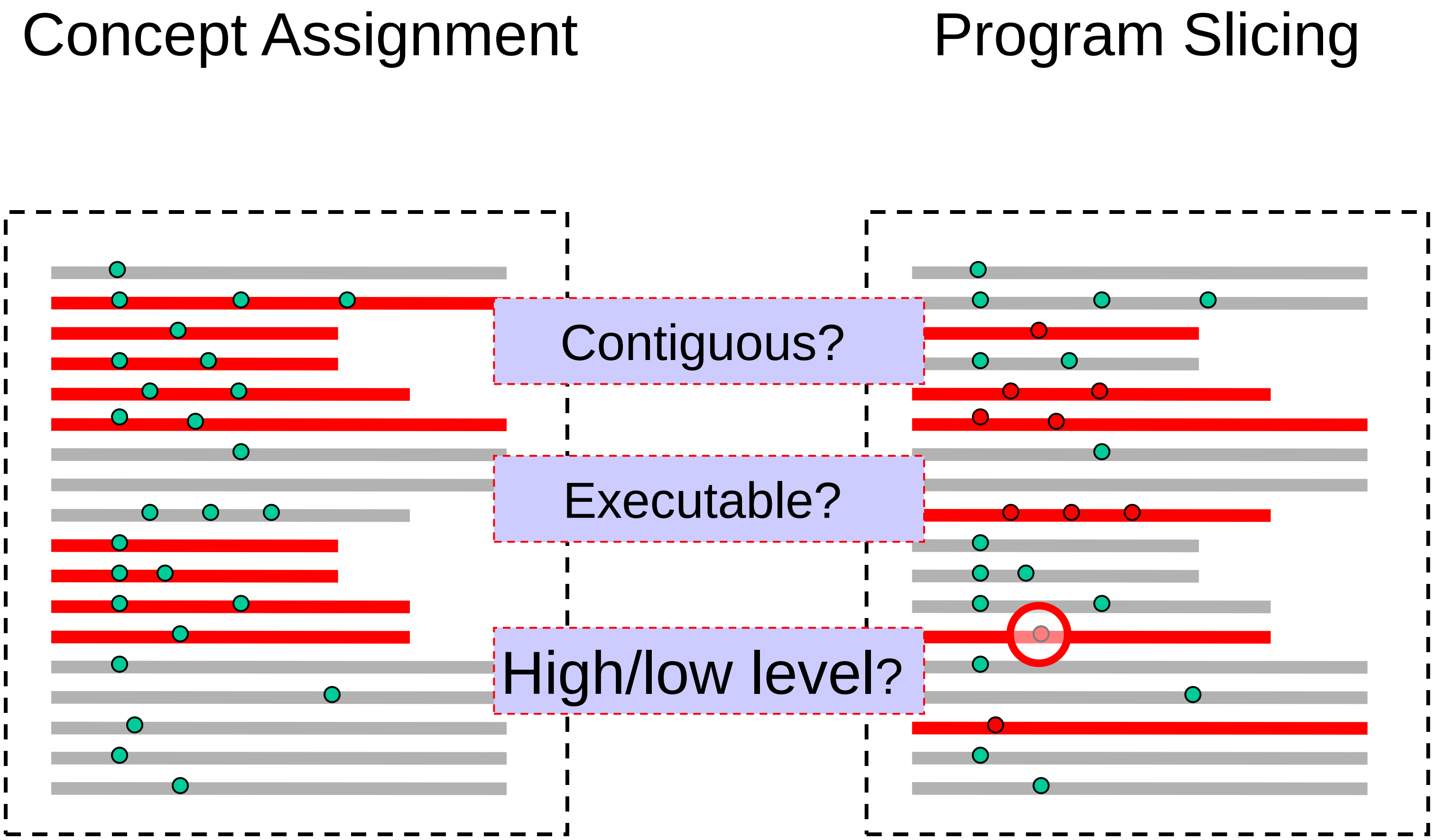
Centre for Research in Evolution, Search & Testing

Program Slicing

which other lines affect the selected line?



we only care about this line



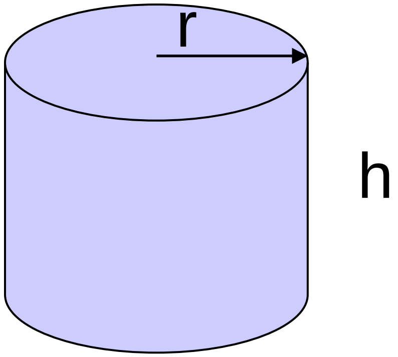
Centre for Research in Evolution,
Search & Testing

Combination 1: Extension

- Concept Slice
  - Using program slicing to ‘extend’ a concept binding by tracing its dependencies
- Algorithm
  - Using concepts as slicing criteria, the concept slice is the union of slices for each program point in the concept

Combination 2: Abbreviation

- Extract key statements within concept bindings
  Less is More!
  - The statements that capture most impact with highest cohesion
  - help to focus attention more rapidly on the core of a concept binding
- Algorithm
  - Intersection of slices with respect to principal variables within a concept binding



```
D=2*r;
perimeter=PI*D;
undersurface=PI*r*r;
sidesurface=perimeter*h;
area=2*undersurface+sidesurface;
volume=undersurface*h;
printf(“\nThe Area is %d\n", area );
printf(“\nThe Volume is %d\n", volume );
```

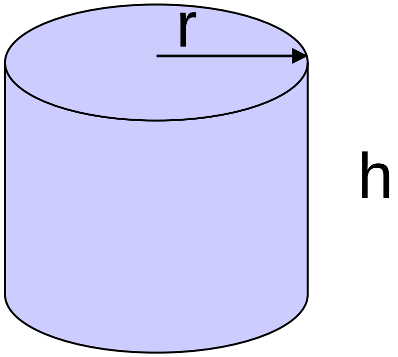
The Results so far

The concept slice has no size explosion.

The identified key statements have high Impact and Cohesion, but some concept bindings do not contain key statements.

Combination 3: Refinement

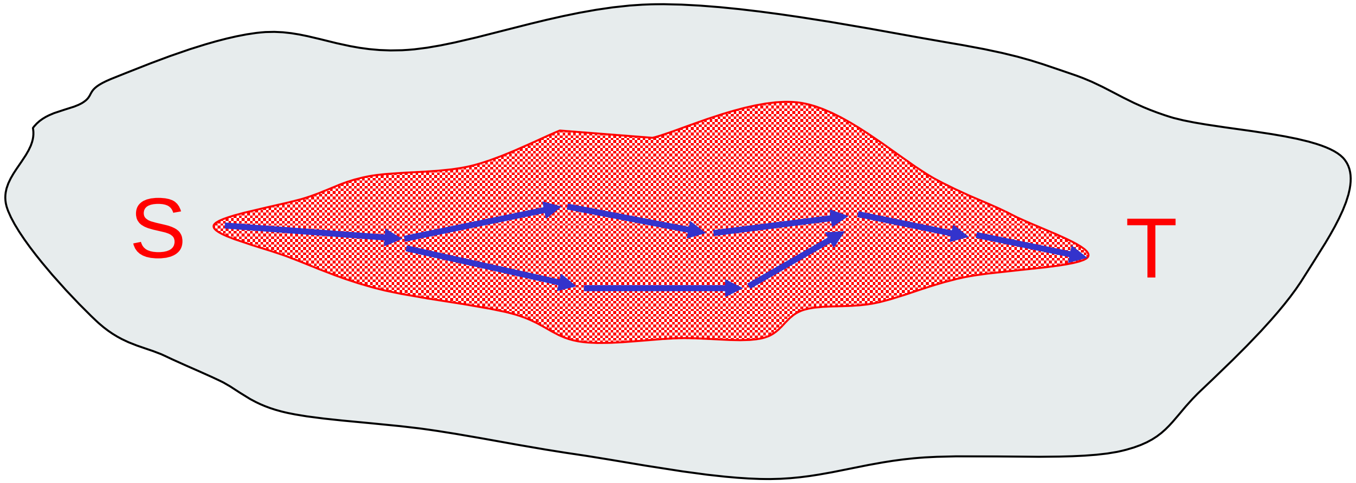
A more accurate dependence based concept binding by removing non-concept-dependent statements



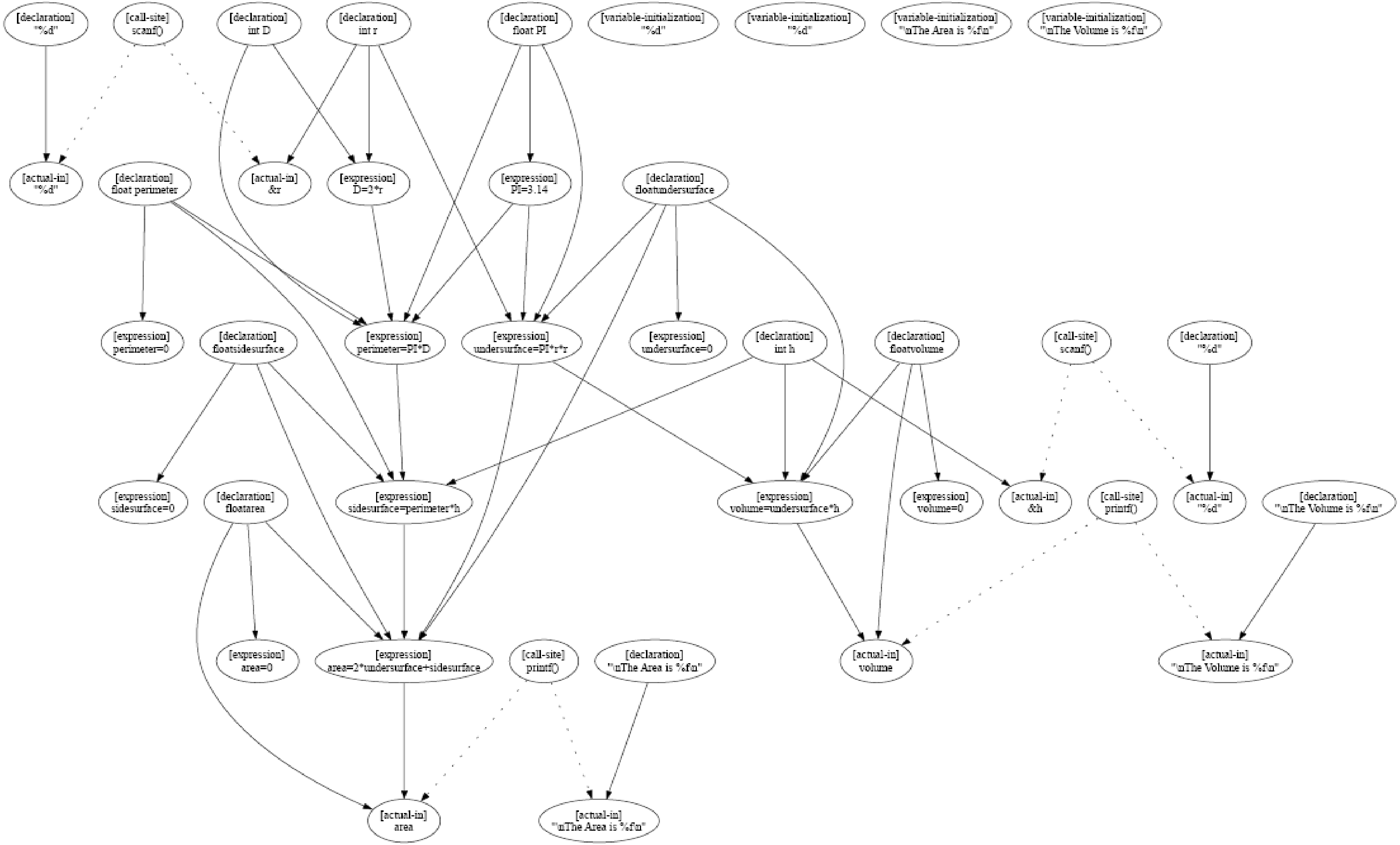
```
D=2*r;
perimeter=PI*D;
undersurface=PI*r*r;
sidesurface=perimeter*h;
area=2*undersurface+sidesurface;
volume=undersurface*h;
printf("\nThe Area is %d\n",  area);
printf("\nThe Volume is %d\n", volume);
```

Program Chopping

Given source *S* and target *T*, what program points transmit effects from *S* to *T*?



Centre for Research in Evolution, Search & Testing


[declaration] "%d"
[call-site] scanf()
[declaration] int D
[declaration] int r
[declaration] float PI
[variable-initialization] "%d"
[variable-initialization] "%d"
[variable-initialization] "\nThe Area is %f\n"
[variable-initialization] "\nThe Volume is %f\n"
[actual-in] "%d"
[declaration] float perimeter
[actual-in] &r
[expression] D=2*r
[expression] PI=3.14
[declaration] floatundersurface
[expression] perimeter=0
[declaration] floatsidesurface
[expression] perimeter=PI*D
[expression] undersurface=PI*r*r
[expression] undersurface=0
[declaration] int h
[declaration] floatvolume
[call-site] scanf()
[declaration] "%d"
[expression] sidesurface=0
[declaration] floatarea
[expression] sidesurface=perimeter*h
[expression] volume=undersurface*h
[expression] volume=0
[actual-in] &h
[call-site] printf()
[actual-in] "%d"
[declaration] "\nThe Volume is %f\n"
[expression] area=0
[expression] area=2*undersurface+sidesurface
[call-site] printf()
[declaration] "\nThe Area is %f\n"
[actual-in] volume
[actual-in] "\nThe Volume is %f\n"
[actual-in] area
[actual-in] "\nThe Area is %f\n"

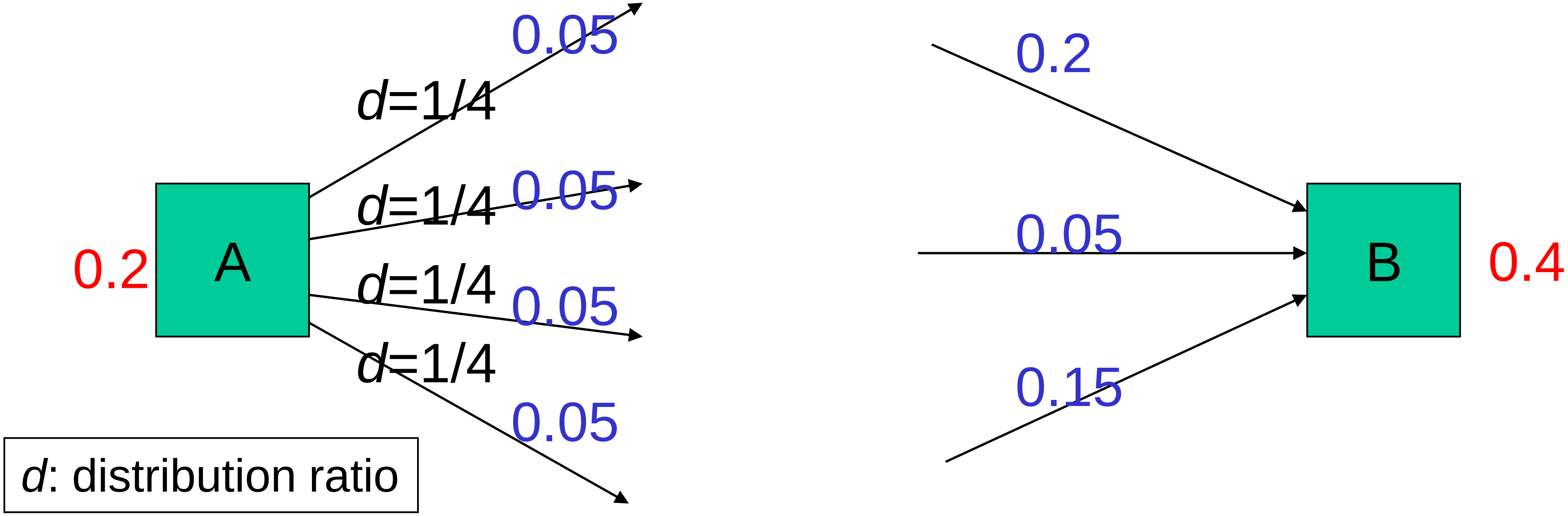
Vertex Rank Model

- Google’s Page Rank Model
- Dependence is transitive
- the weight of a vertex will be distributed following the outgoing edges and inherited through incoming edges.

Weight of Nodes

- sum of all node weights = 1
- weight of node represents the importance of dependence of a vertex

Weights of Edges



- Node weight is distributed to each outgoing edge
- Edge weights are collected at the destination node
- sum of all outgoing edge weights = origin node weight
- sum of all incoming edge weights = destination node weight

Centre for Research in Evolution, Search & Testing

Definition of Weights

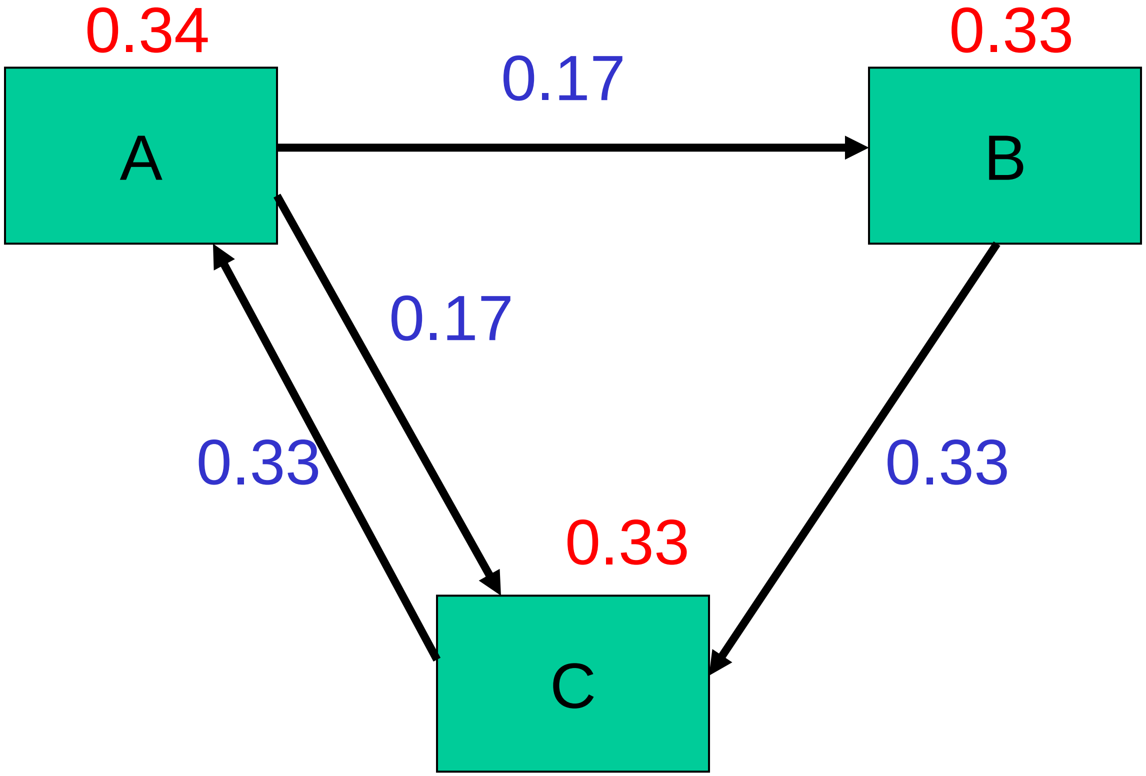
$$\begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}^t \cdot \begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix}$$

W: node weight vector

D^t: transposed matrix of distribution ratios

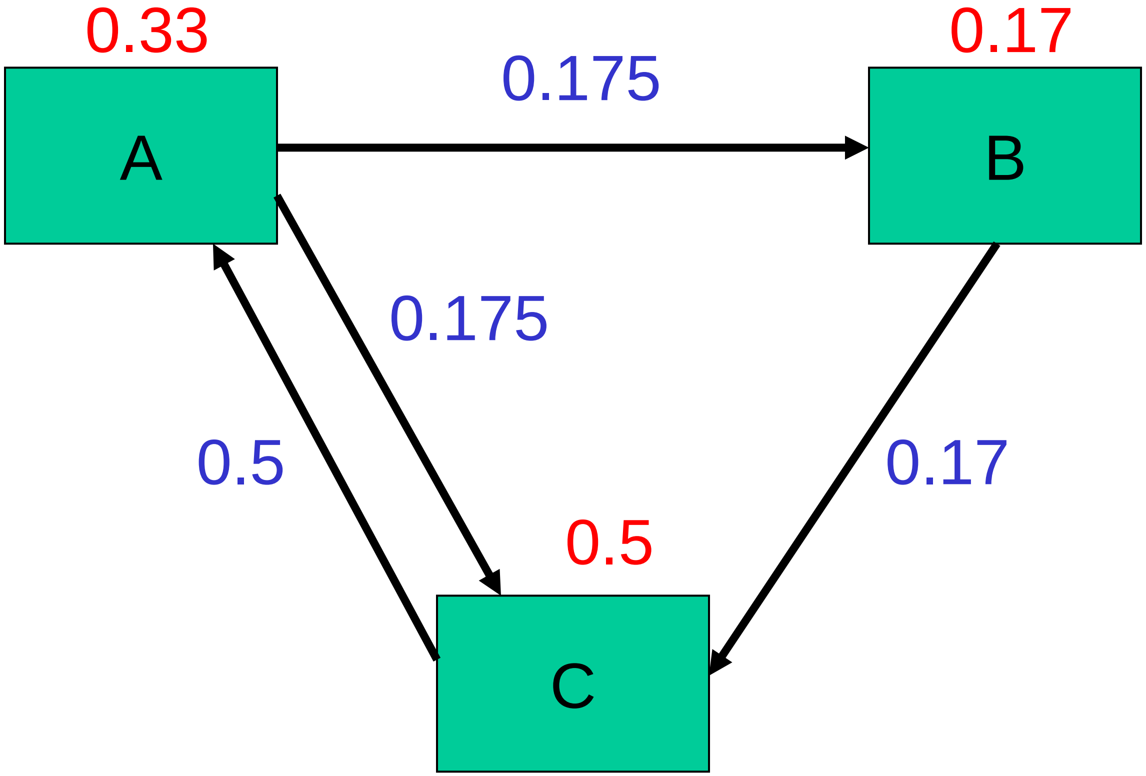
Centre for Research in Evolution, Search & Testing

Propagating Weights



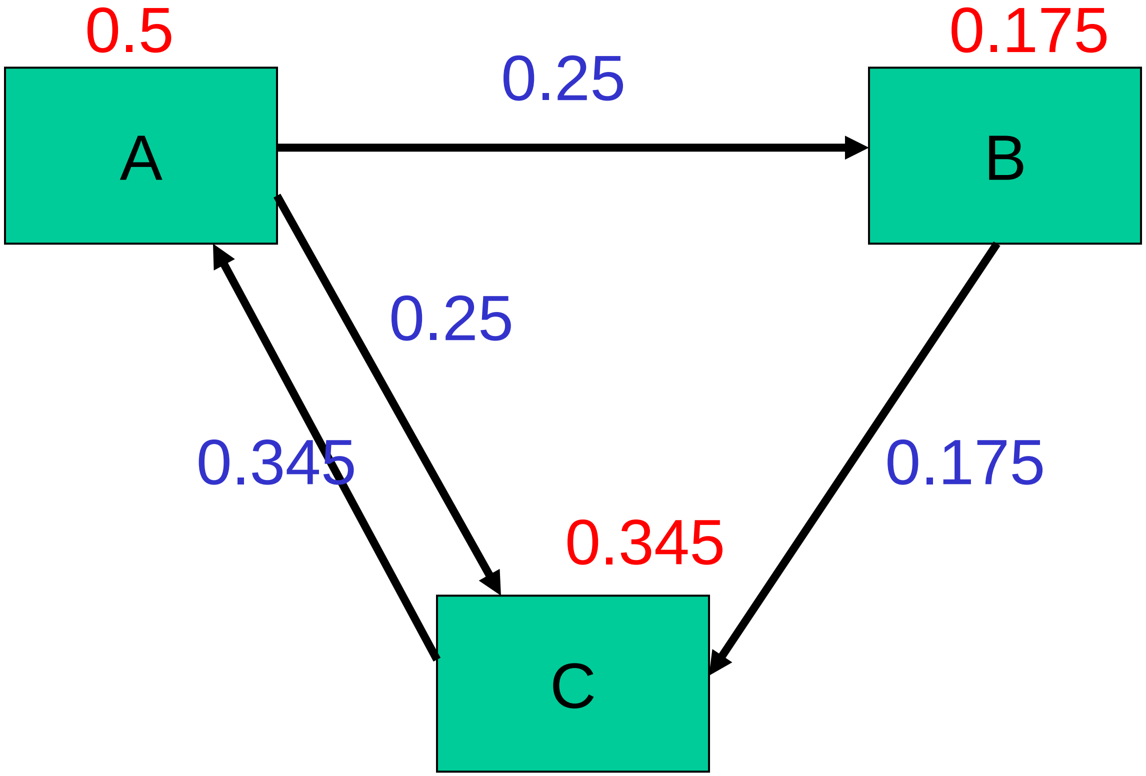
Centre for Research in Evolution, Search & Testing

Propagating Weights

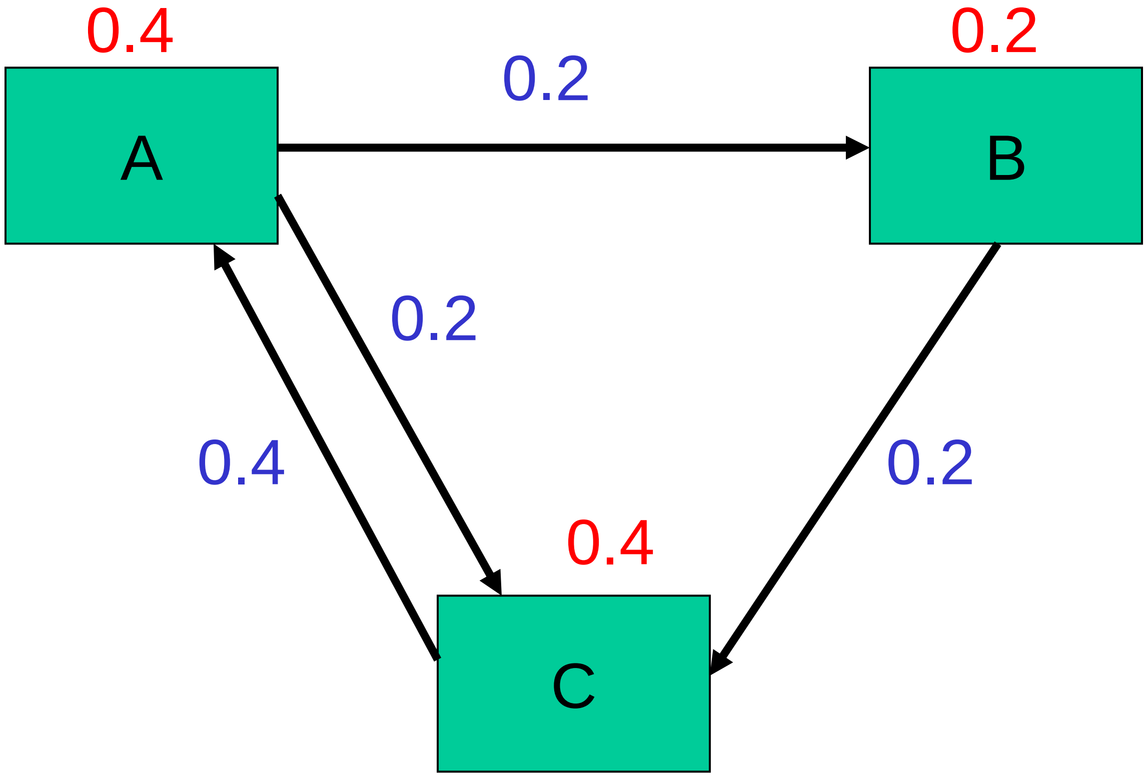


Centre for Research in Evolution, Search & Testing

Propagating Weights



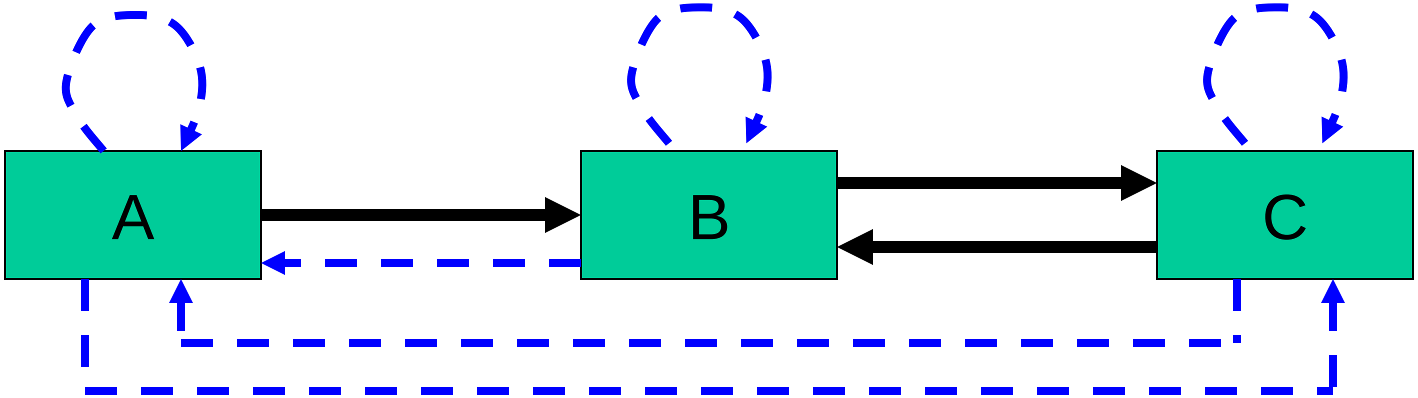
Propagating Weights



- Stable weight assignment
  - next-step weights are the same as previous ones

Centre for Research in Evolution,
Search & Testing

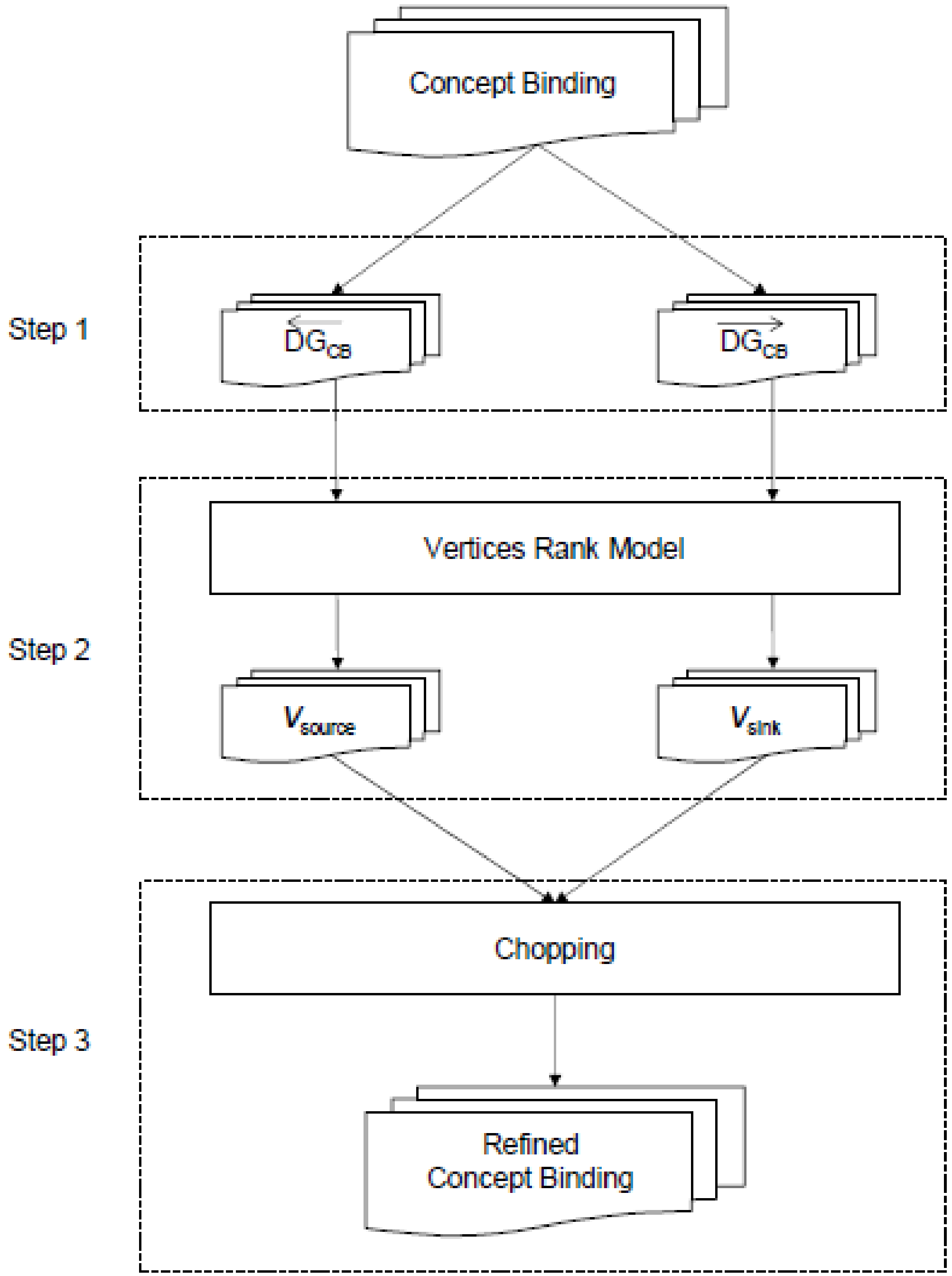
Pseudo Use Relation



- Weight computation does not always converge
- Add a pseudo edge from a node to another, if there is no 'real' edge
- Distribution ratios:

  pseudo edges << real edges

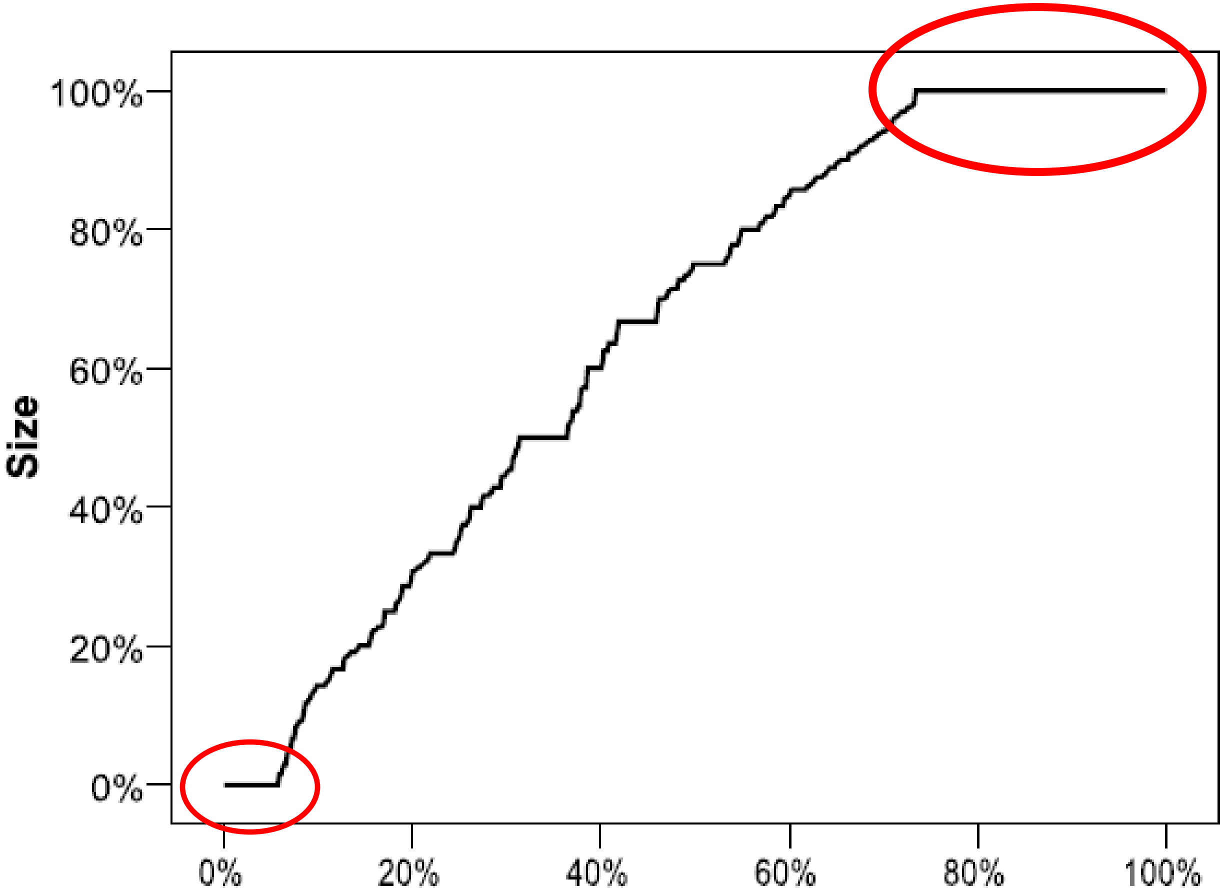
Centre for Research in Evolution, Search & Testing


Concept Binding
Step 1
$\overleftarrow{DG}_{CB}$
$\overrightarrow{DG}_{CB}$
Step 2
Vertices Rank Model
V_{source}
V_{sink}
Step 3
Chopping
Refined Concept Binding

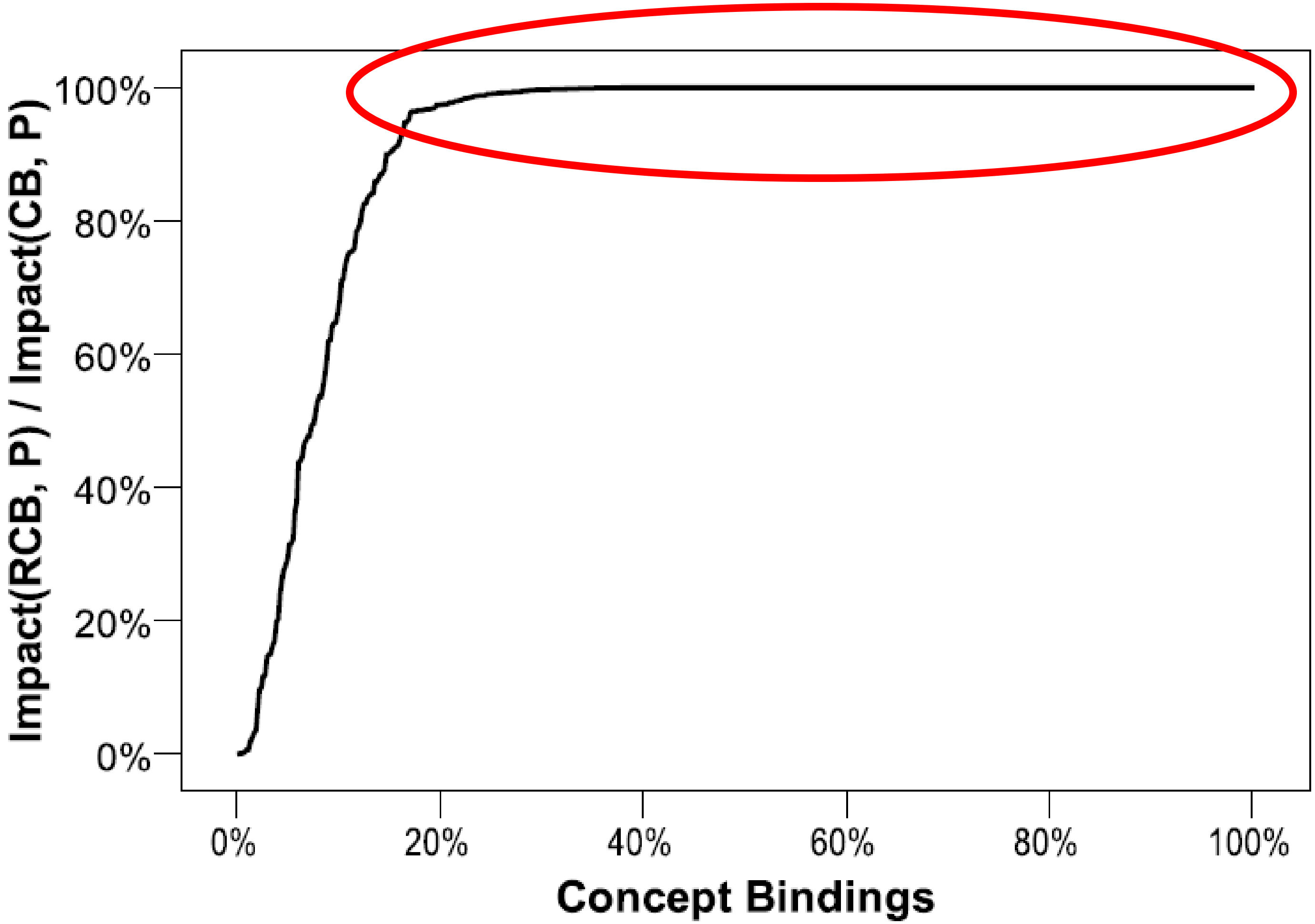
Empirical Study

- Tools
  - WeSCA and CodeSurfer
- 10 Subject programs
  - Open source and industry code
  - More than 600 concept bindings are extracted
- Dependence based metrics are defined
- Statistical analysis

Size reduction

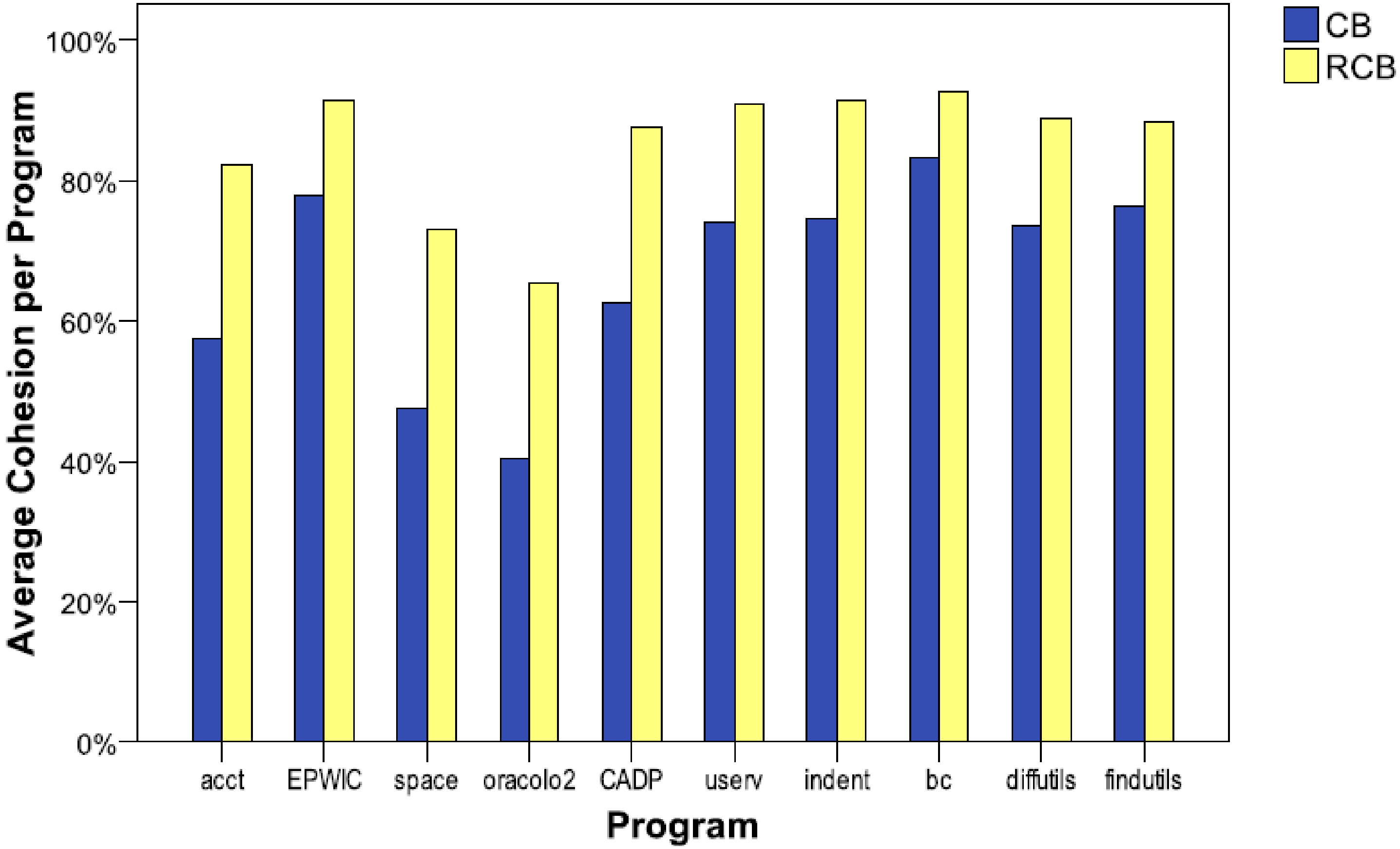


Impact



Impact(RCB, P) / Impact(CB, P)
100%
80%
60%
40%
20%
0%
0%
20%
40%
60%
80%
100%
Concept Bindings

Cohesion



Centre for Research in Evolution, Search & Testing

Summary

- The combination of approaches can be fully automated and implemented.

- Concept refinement is better than concept extension and concept abbreviation.

Questions?