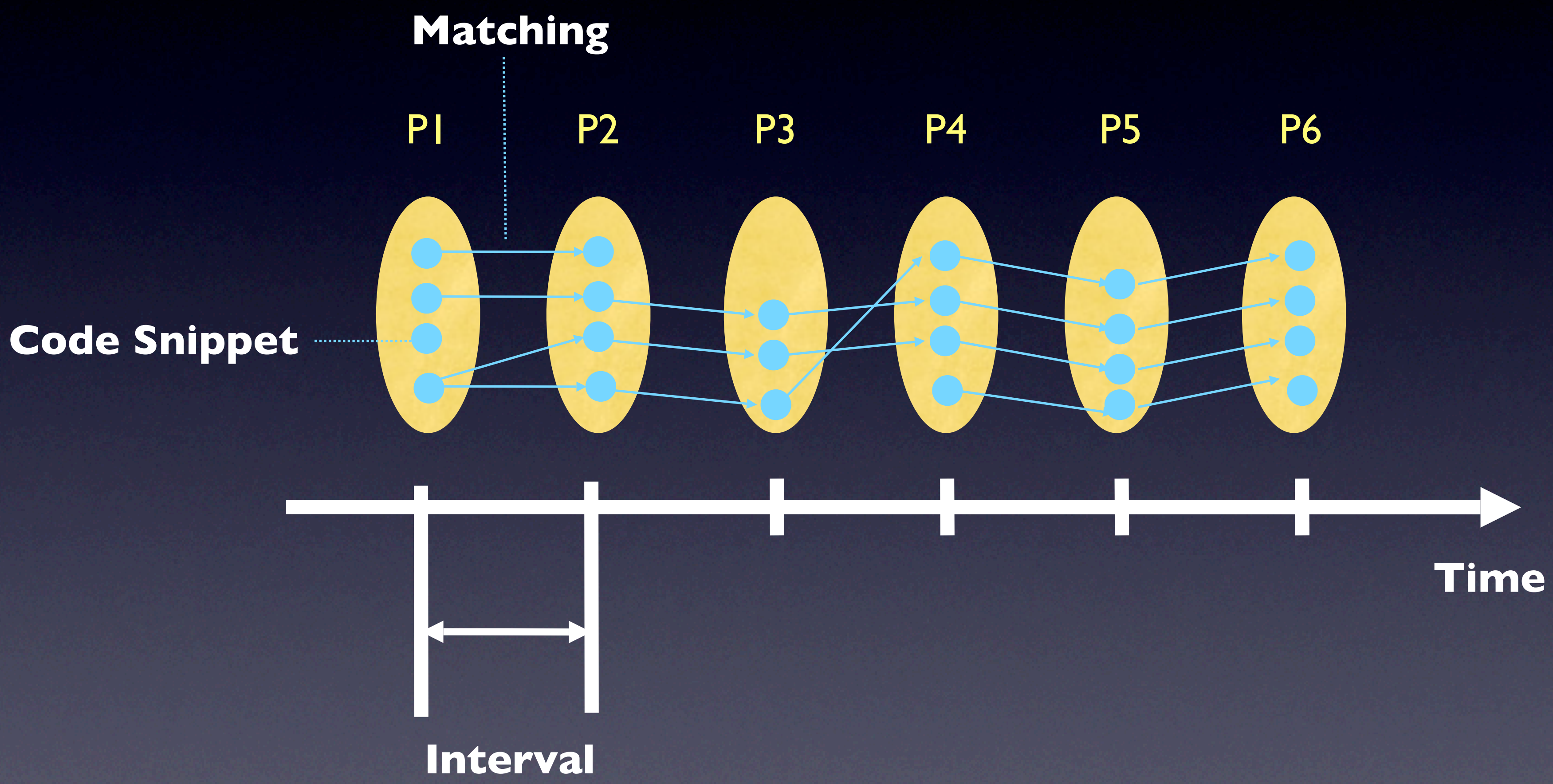
Matching Program Elements for Multi-Version Program Analyses

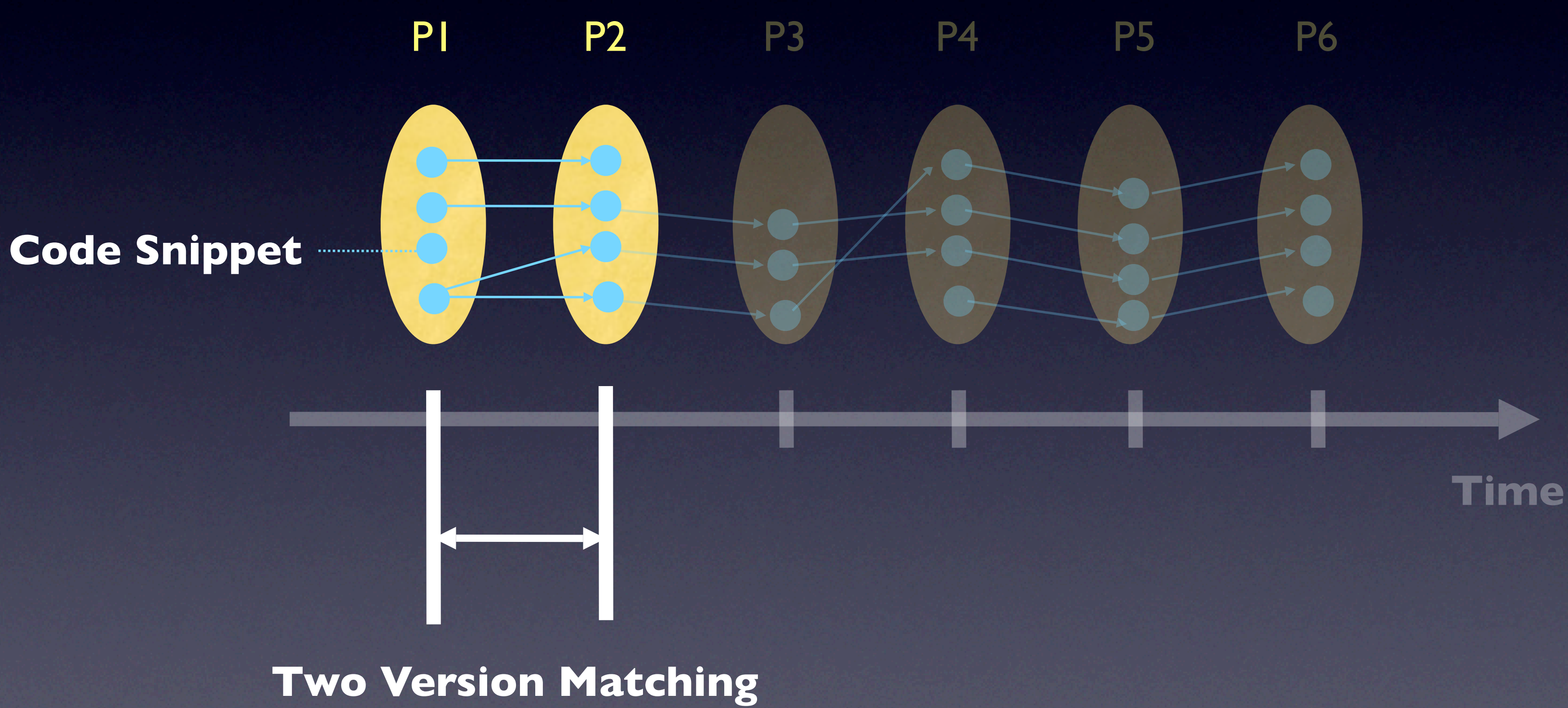
Miryung Kim, David Notkin
University of Washington

The Third International Workshop on Mining Software Repositories, Shanghai China, 2006

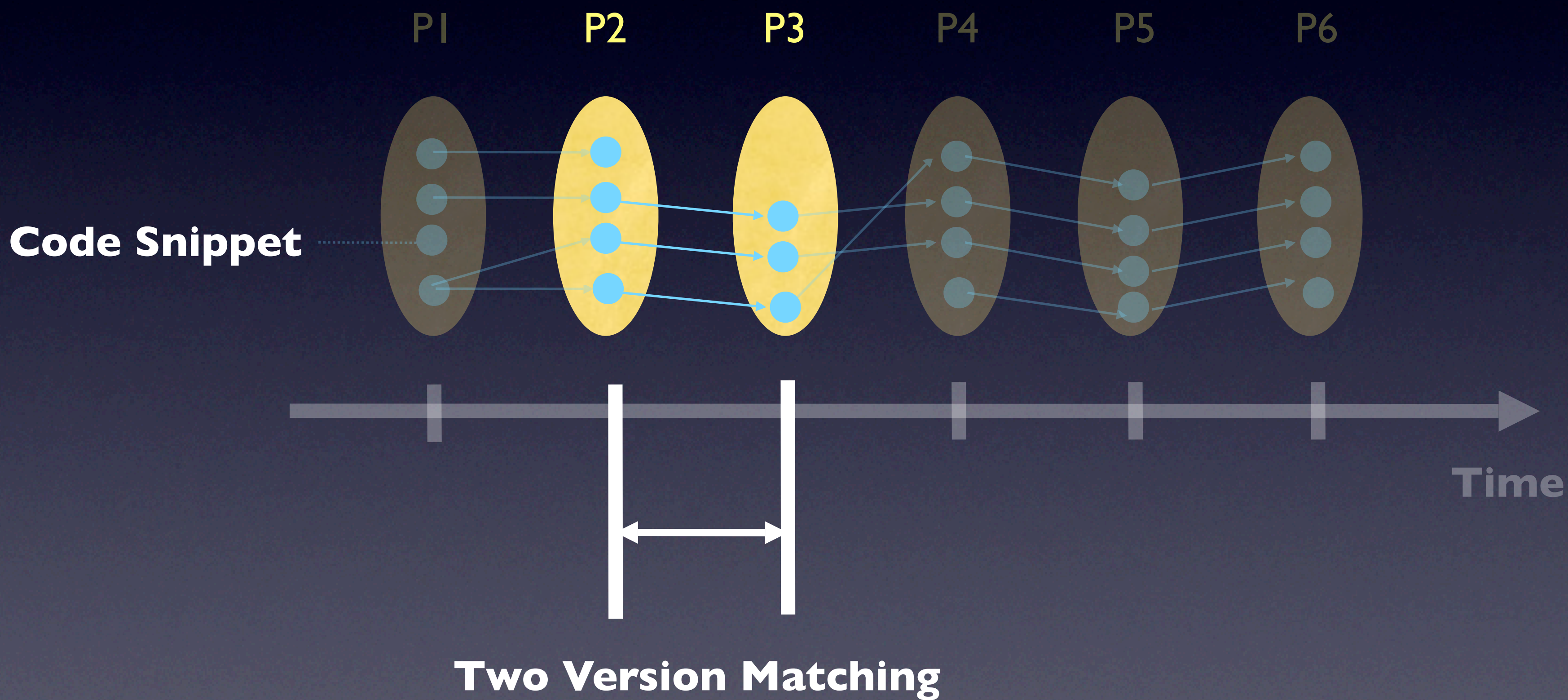
Multi-Version Analysis



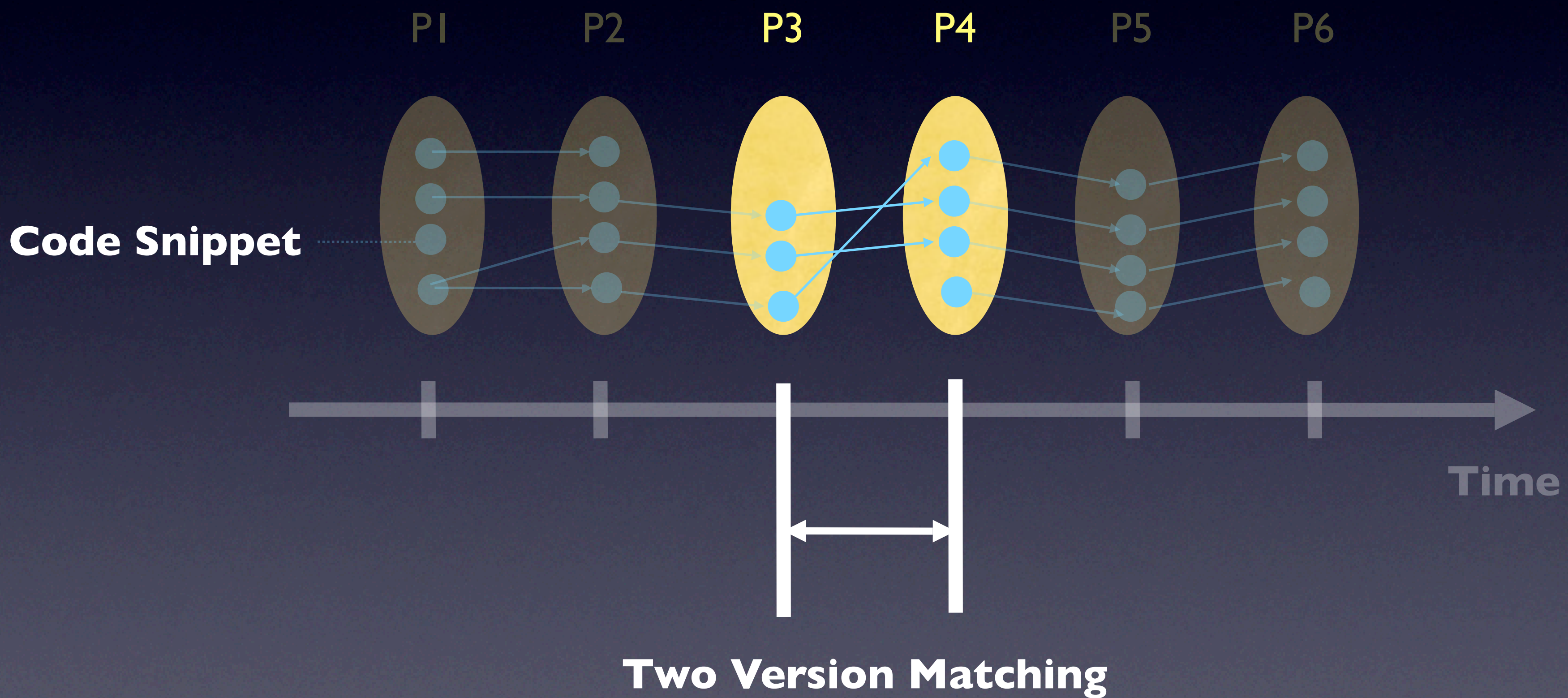
Matching between Two Versions



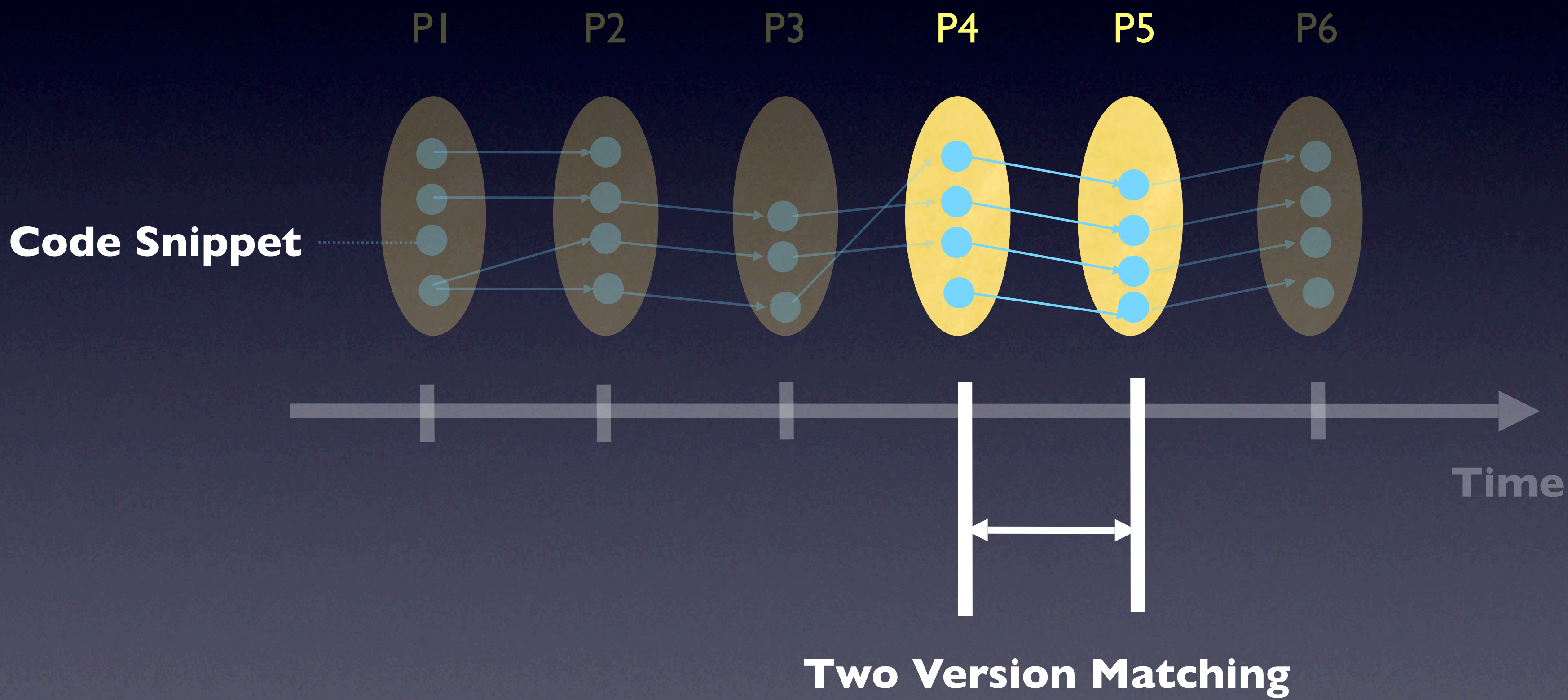
Matching between Two Versions



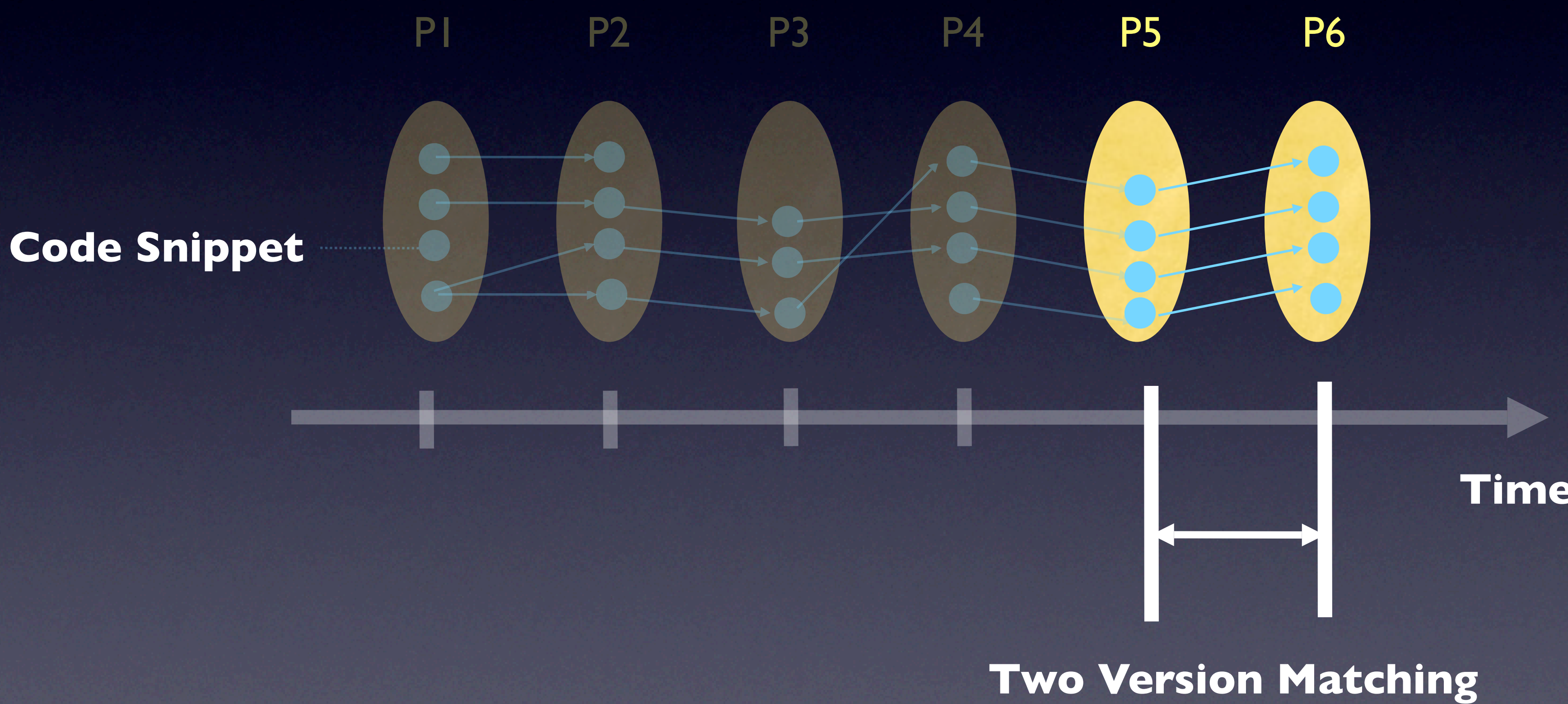
Matching between Two Versions



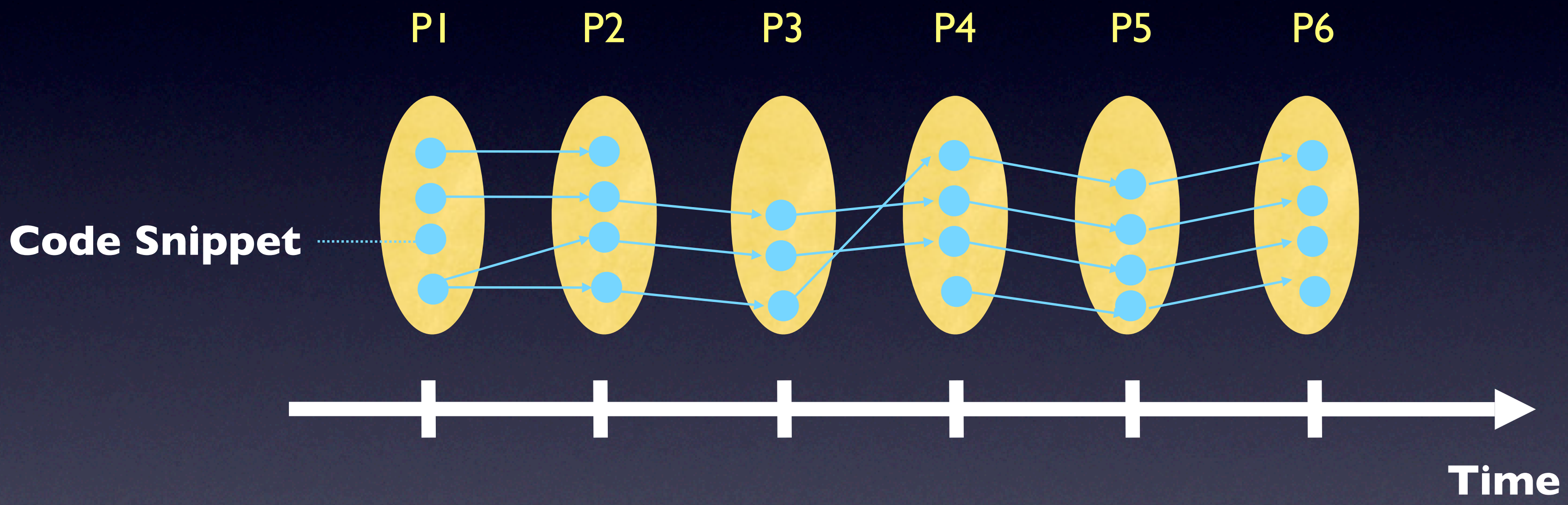
Matching between Two Versions



Matching between Two Versions



Composing Two-Version Matching Results



Program Element Matching Problem

- A fundamental building block for multi-version analyses.
  - co-change [ZWDZ04, YMNC04], instability [BW03], signature change [KWB05], type change [NFH05], code clone change [KSNM05].
- Also used for software version merging, regression testing, and profile propagation.

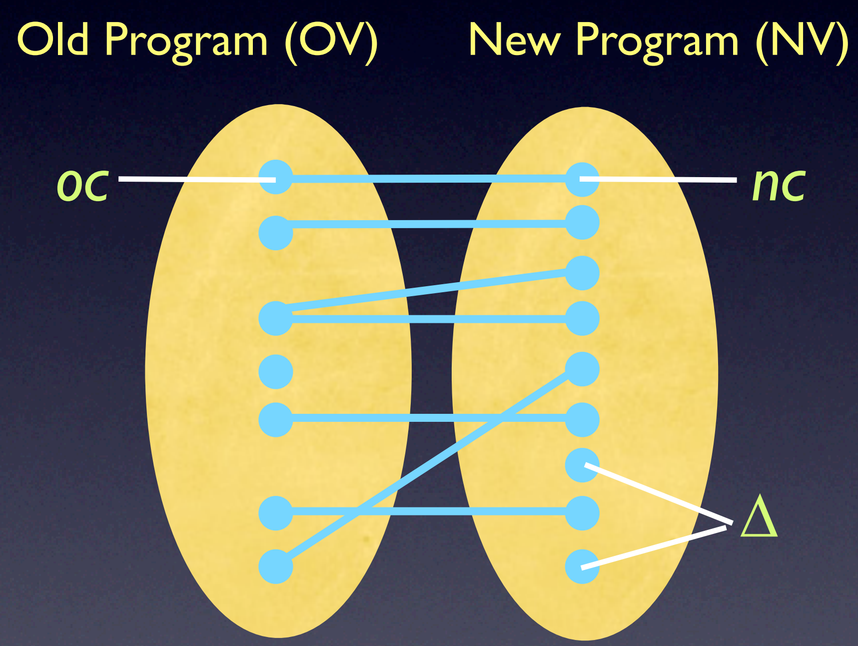
Matching Problem

Determine the differences Δ between OV and NV.

For a code fragment nc ∈ NV,

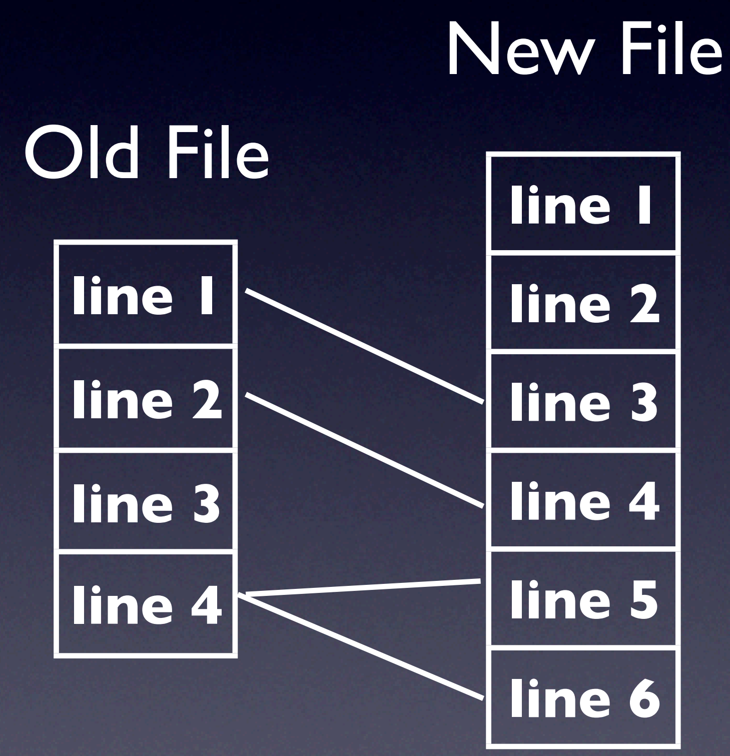
determine whether nc ∈ Δ.

If not, find nc's corresponding origin oc in OV.



Characterization of Matching Problem

	e.g. *diff*
Program Representation	string (a sequence of lines)
Matching Granularity	line
Matching Multiplicity	1:1
Matching Criteria / Heuristics	Two lines are equal.



Comparison

Matching Technique	Program Representation	Granularity	Multiplicity	Heuristics		
				Name	Posi-tion	Similarity
name matching	Entity	Procedure/ File	1:1	✔		
diff [HS77]	String	Line	1:1			✔
bdiff [Tic84]	String	Line	1:n			✔
cdiff [Yang91]	AST	AST node	1:1			✔
Neamtiu et al.	AST	Type, Variable	1:1	✔		
jdiff [AOH04]	CFG	CFG node	1:1	✔		✔
BMAT [WPM00]	Binary code	Code block	1:1, n:1	✔	✔	✔
Clone detectors	Various	Various	n:n			✔
Zou, Godfrey	Hybrid	Procedure	1:1, n:1, 1:n	✔		✔
S. Kim et al.	Hybrid	Procedure	1:1	✔		✔

Comparison

Matching Technique	Program Representation	Granularity	Multiplicity	Heuristics		
				Name	Posi-tion	Similar ity
name matching	Entity	Procedure/ File	1:1	✔		
diff [HS77]	String	Line	1:1			✔
bdiff [Tic84]	String	Line				
cdiff [Yang91]	AST	AST node				
Neamtiu et al.	AST	Type, Variable				
jdiff [AOH04]	CFG	CFG node	1:1	✔		✔
BMAT [WPM00]	Binary code	Code block	1:1, n:1	✔	✔	✔
Clone detectors	Various	Various	n:n			✔
Zou, Godfrey	Hybrid	Procedure	1:1, n:1, 1:n	✔		✔
S. Kim et al.	Hybrid	Procedure	1:1			✔

Many techniques produce mappings at a fixed granularity.

Comparison

Matching Technique	Program Representation	Granularity	Multiplicity	Heuristics		
				Name	Posi-tion	Similar ity
name matching	Entity	Procedure/ File	1:1	✔		
diff [HS77]	String	Line	1:1			✔
bdiff [Tic84]	String	Line				
cdiff [Yang91]	AST	AST node				
Neamtiu et al.	AST	Type, Variable				
jdiff [AOH04]	CFG	CFG node	1:1	✔		✔
BMAT [WPM00]	Binary code	Code block	1:1, n:1	✔	✔	✔
Clone detectors	Various	Various	n:n			✔
Zou, Godfrey	Hybrid	Procedure	1:1, n:1, 1:n	✔		✔
S. Kim et al.	Hybrid	Procedure	1:1	✔		✔

Many fine-grained techniques require mappings at a higher level.

Comparison

Matching Technique	Program Representation	Granularity	Multiplicity	Heuristics		
				Name	Posi-tion	Similar ity
name matching	Entity	Procedure/ File	1:1	✔		
diff [HS77]	String	Line	1:1			✔
bdiff [Tic84]	[illegible]	[illegible]	1:n			✔
cdiff [Yang91]	[illegible]	[illegible]	1:1			✔
Neamtiu et al.	[illegible]	[illegible]le	1:1	✔		
jdiff [AOH04]	CF[illegible]	[illegible]FG no[illegible]	1:1	✔		✔
BMAT [WPM00]	Binary code	Code block	1:1, n:1	✔	✔	✔
Clone detectors	Various	Various	n:n			✔
Zou, Godfrey	Hybrid	Procedure	1:1, n:1, 1:n	✔		✔
S. Kim et al.	Hybrid	Procedure	1:1	✔		✔

Many techniques assume 1:1 mappings.

Comparison

Matching Technique	Program Representation	Granularity	Multiplicity	Heuristics		
				Name	Posi-tion	Similar ity
name matching	Entity	Procedure/ File	1:1	✔		
diff [HS77]	String	Line	1:1			✔
bdiff [Tic84]	String	[illegible]	[illegible]			✔
cdiff [Yang91]	AST	[illegible]	[illegible]			✔
Neamtiu et al.	AST	[illegible]	[illegible]	✔		
jdiff [AOH04]	CFG	[illegible]	[illegible]	✔		✔
BMAT [WPM00]	Binary code	Code block	1:1, n:1	✔	✔	✔
Clone detectors	Various	Various	n:n			✔
Zou, Godfrey	Hybrid	Procedure	1:1, n:1, 1:n	✔		✔
S. Kim et al.	Hybrid	Procedure	1:1	✔		✔

Many techniques heavily rely on heuristics to reduce a matching scope.

Evaluation based on Hypothetical Change Scenarios

- Inadequate evaluation for most matching techniques except S. Kim's origin analysis
- We created a set of hypothetical program change scenarios.
  - scenario 1 (small changes):
    - changes in the nested level of a control structure
    - semantics-preserving statement reordering.
  - scenario 2 (large changes):
    - procedure level renaming and splitting
  - renaming, splitting, and merging scenarios.

Evaluation based on Hypothetical Change Scenarios

Matching Technique	Scenario		Transformation				Weaknesses
			Split/Merge		Rename		
	Small	Large	Proc	File	Proc	File	
diff	◒	○	○	○	◒	○	✘ require file level mapping
bdiff	●	○	◒	○	◒	○	✘ require file level mapping
cdiff	○	○	○	○	○	○	✘ require procedure level mapping ✘ sensitive to nested level change
Neamtiu et al.	○	○	○	○	○	○	✘ partial AST matching
jdiff	●	◒	○	○	◒	◒	✘ sensitive control structure change
BMAT	○	●	○	○	●	●	✘ 1:1 mapping only ✘ only applicable to binary code
Zou, Godfrey	○	●	●	●	●	●	✘ semi-automatic analysis
S. Kim et al.	○	●	○	○	●	●	✘ 1:1 mapping only

● good, ◒ mediocre, ○ poor

Evaluation based on Hypothetical Change Scenarios

Matching Technique	Scenario		Transformation				Weaknesses
			Split/Merge		Rename		
	Small	Large	Proc	File	Proc	File	
diff	◒	○	○	○	[illegible]	[illegible]	✘ require file level mapping
bdiff	●	○	◒	[illegible]	[illegible]	[illegible]	… file level mapping
cdiff	○	○	○	[illegible]	[illegible]	[illegible]	… procedure level mapping …ive to nested level change
Neamtiu et al.	○	○	○	○	○	○	✘ partial AST matching
jdiff	●	◒	○	○	◒	◒	✘ sensitive control structure change
BMAT	○	●	○	○	●	●	✘ 1:1 mapping only ✘ only applicable to binary code
Zou, Godfrey	○	●	●	●	●	●	✘ semi-automatic analysis
S. Kim et al.	○	●	○	○	●	●	✘ 1:1 mapping only

● good, ◒ mediocre, ○ poor

Fine-grained matching techniques do not work well in case of large changes.

Evaluation based on Hypothetical Change Scenarios

Matching Technique	Scenario		Transformation				Weaknesses
			Split/Merge		Rename		
	Small	Large	Proc	File	Proc	File	
diff	◒	○	○	○	◒	○	✗ require file level mapping
bdiff	●	○	◒	○	◒		...ng
cdiff	○	○	○	○	○		...pping ...ange
Neamtiu et al.	○	○	○	○	○		
jdiff	●	◒	○	○	◒	◒	✗ sensitive control structure change
BMAT	○	●	○	○	●	●	✗ 1:1 mapping only ✗ only applicable to binary code
Zou, Godfrey	○	●	●	●	●	●	✗ semi-automatic analysis
S. Kim et al.	○	●	○	○	●	●	✗ 1:1 mapping only

● good, ◒ mediocre, ○ poor

Due to 1:1
mapping assumptions, they
perform poorly when
splitting or merging.

Evaluation based on Hypothetical Change Scenarios

Matching Technique	Scenario		Transformation				Weaknesses
			Split/Merge		Rename		
	Small	Large	Proc	File	Proc	File	
diff	○		○	○	◒	○	✘ require file level mapping
bdiff					◒	○	✘ require file level mapping
cdiff				○	○	○	✘ require procedure level mapping ✘ sensitive to nested level change
Neamtiu et al.	○	○	○	○	○	○	✘ partial AST matching
jdiff	●	◒	○	○	◒	◒	✘ sensitive control structure change
BMAT	○	●	○	○	●	●	✘ 1:1 mapping only ✘ only applicable to binary code
Zou, Godfrey	○	●	●	●	●	●	✘ semi-automatic analysis
S. Kim et al.	○	●	○	○	●	●	✘ 1:1 mapping only

● good, ◒ mediocre, ○ poor

Techniques that require higher level correspondences perform poorly in case of renaming.

Evaluation based on Hypothetical Change Scenarios

Matching Technique	Scenario		Transformation				Weaknesses
			Split/Merge		Rename		
	Small	Large	Proc	File	Proc	File	
diff	◒	○	○	○	◒	○	✘ require file level mapping
bdiff	●	○	◒	○	◒		ng
cdiff	○	○	○	○	○		pping ange
Neamtiu et al.	○	○	○	○	○		
jdiff	●	◒	○	○	◒	◒	✘ sensitive control structure change
BMAT	○	●	○	○	●	●	✘ 1:1 mapping only ✘ only applicable to binary code
Zou, Godfrey	○	●	●	●	●	●	✘ semi-automatic analysis
S. Kim et al.	○	●	○	○	●	●	✘ 1:1 mapping only

● good, ◒ mediocre, ○ poor

Zou and Godfrey's origin analysis will work well but is semi-automatic.

Current Work

- Matching representation
  - expressible for various granularity and structure
  - compact
  - composable results for multi-version analysis
- Evaluation metric based on a matching representation

First Order Logic Rule to Represent Matches

old	new
chart:ChartFactory-createPieChart [String, PieDataset, boolean]->JChart	package class method parameter return chart:ChartFactory-createPieChart [String, PieDataset, boolean, boolean, boolean]->JChart
chart:ChartFactory-createGanttChart [String, IntervalSet, boolean]->JChart	chart:ChartFactory-createGanttChart [String, IntervalSet, boolean, boolean, boolean]->JChart
chart:ChartFactory-createLineXYChart [String, XYDataset, boolean]->JChart	chart:ChartFactory-createLineXYChart [String, XYDataset, boolean, boolean, boolean]->JChart
...	...
...	...

First Order Logic Rule to Represent Matches

old	new
chart:ChartFactory-createPieChart [String, PieDataset, boolean]->JChart	package ↓ class ↓ method ↓ parameter ↓ return ↓ chart:ChartFactory-createPieChart [String, PieDataset, boolean, boolean, boolean]->JChart
chart:ChartFactory-createGanttChart [String, IntervalSet, boolean]->JChart	chart:ChartFactory-createGanttChart [String, IntervalSet, boolean, boolean, boolean]->JChart
chart:ChartFactory-createLineXYChart [String, XYDataset, boolean]->JChart	chart:ChartFactory-createLineXYChart [String, XYDataset, boolean, boolean, boolean]->JChart
...	...
...	...

$$\forall x\text{: FullProcedureName, PatternMatch(x.method, “create*”)} \wedge \text{x.class =“ChartFactory”} \rightarrow$$

$$\text{new(x).parameter = concatenate (x.parameter, [boolean, boolean])}$$

Summary

- Matching program elements is a fundamental building block for multi-version program analyses.
- We characterized the code matching problem and compared matching techniques based on several criteria.
- We identified limitations of current matching techniques and proposed future directions.

Acknowledgment: Dagstuhl 05261 participants for ideas and discussions

Back Up Slides

Motivating Scenarios

- fixing a bug in forked projects
- monitoring interface evolution
- other code evolution analyses
  - co-change [ZWDZ04, YMNC04], instability [BW03], signature change [KWB05], type change [NFH05], code clone change [KSNM05].

First Order Logic Rule to Represent Matches

All methods that start with "create" in the class ChartFactory take additional input parameters [boolean, boolean] in the new version.

$$\forall x: \text{FullProcedureName}, \text{PatternMatch}(x.\text{method}, \text{"create*"}) \land x.\text{class} = \text{"ChartFactory"} \rightarrow$$

$$\text{new}(x).\text{parameter} = \text{concatenate}\ (x.\text{parameter}, [\text{boolean}, \text{boolean}])$$

Surveyed Techniques

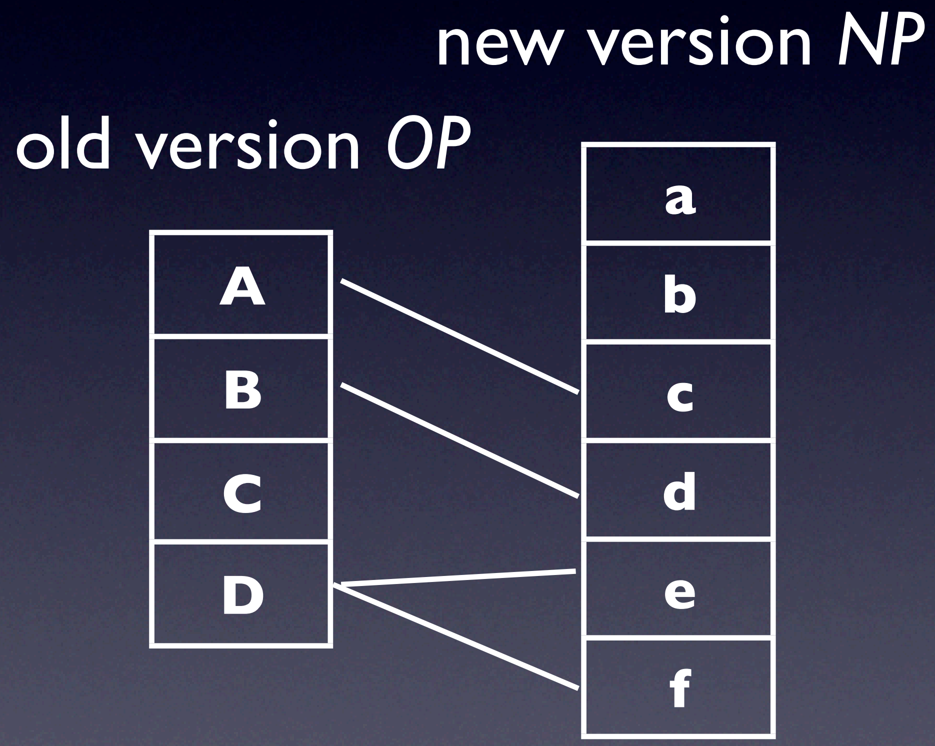
- name matching
- String: *diff* [HS77] and *bdiff* [Tic84]
- AST: *cdiff* [Yang91], Neamtiu et al. [NFH05]
- CFG: *jdiff* [AOH04]
- Binary Code: *BMAT* [WPM00]
- clone detectors
- tools that infer refactoring events [ZG05] [KPW05], etc.

Two-Version Matching Problem

Determine the differences Δ between OP and NP.
For a code fragment nc ∈ NP,

determine whether nc ∈ Δ.

If not, find nc's corresponding origin oc in OP.



{A,c}, {B,d}, {D,e}, {D,f}
{C, ∅}, {∅, a}, {∅,b}

Challenges

- Absence of benchmarks
- Various granularity support
- Renaming, splitting, merging, and copying
- Scalability (e.g. matching result representation.)

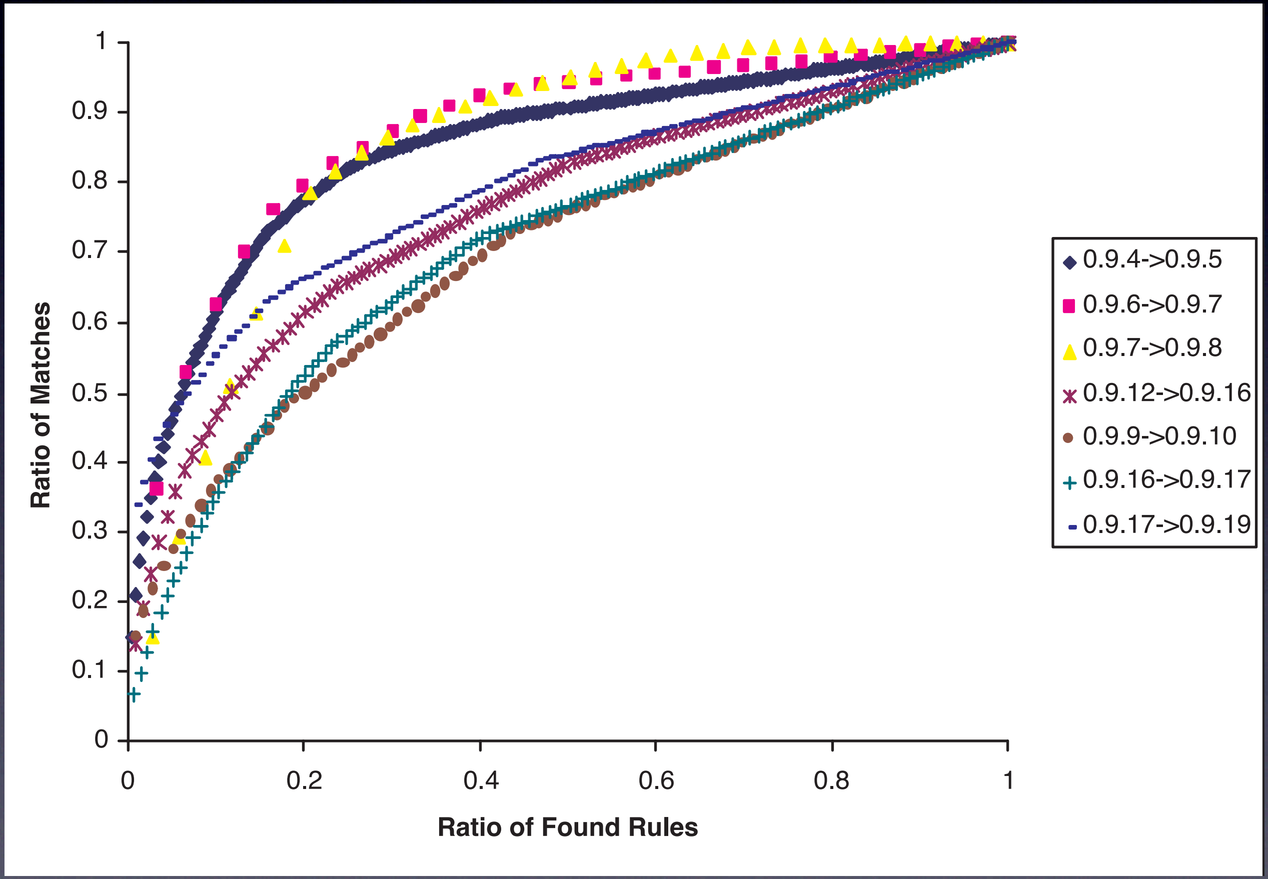
Limitations (1)

- Most matching techniques
  - assume 1:1 mappings,
  - produce mappings at a fixed granularity, and
  - require correspondences at a certain level.
- Fine-grained matching techniques are costly and do not work well when there are many changes at a high level.

Limitations (2)

- Most matching techniques are inadequately evaluated.
  - Matching results are long and not compact.
  - Matching results may not be intuitive and may not be easy to understand.
  - There's no global metric that measures the quality of matching results.
- It may not be straightforward to compose two version matching results for multi-version matching results.

Benefits of Representing Matches as Rules



Average rule match ratio
(# matches/ # rules) = 6.61