Flexeme: Untangling Commits Using Lexical Flows

Profir-Petru Pârțachi[†]

Santanu Kumar Dash[‡]

Miltos Allamanis[•]

Earl T. Barr[†]

† Department of Computer Science, University College London, London, United Kingdom
‡ Department of Computer Science, University of Surrey, Guildford, Surrey, United Kingdom
• Microsoft Research, Cambridge, Cambridgeshire, United Kingdom

Tangled Commits

The features implemented by these two (atomic) patches are unrelated.

Tangled patches introduce bias.

```
@@ -127,31 +137,32 @@ namespace Terminal {
 {
  this.barItems = barItems;
  this.host = host;

+ ColorScheme = Colors.Menu;                                    (1a)
  CanFocus = true;
 }
 public override void Redraw(Rect region)
 {

- Driver.SetAttribute(Colors.Menu.Normal);                      (1b)
- DrawFrame(region, true);
+ Driver.SetAttribute(ColorScheme.Normal);
+ DrawFrame(region, padding: 0, fill: true);
  for (int i = 0;
       i < barItems.Children.Length;
       i++)
  {
   var item = barItems.Children [i];
   Move(1, i+1);
-  Driver.SetAttribute(

-   item == null ? Colors.Base.Focus :                           (1c)
-   i == current ? Colors.Menu.Focus :
-   Colors.Menu.Normal
-  );
+  Driver.SetAttribute(
+   item == null ? Colors.Base.Focus :
+   i == current ? ColorScheme.Focus :
+   ColorScheme.Normal
+  );
  for (int p = 0; p < Frame.Width-2; p++)
   if (item == null)

-   Driver.AddSpecial(SpecialChar.HLine);                        (2a)
+   Driver.AddRune(Driver.HLine);
   else

-   Driver.AddCh(' ');                                           (2b)
+   Driver.AddRune(' ');
   if (item == null)
    continue;
   Move(2, i + 1);
   DrawHotString(item.Title,

-   i == current? Colors.Menu.HotFocus :                         (1d)
-                 Colors.Menu.HotNormal,
-   i == current ? Colors.Menu.Focus :
-                  Colors.Menu.Normal);
+   i == current? ColorScheme.HotFocus :
+                 ColorScheme.HotNormal,
+   i == current ? ColorScheme.Focus :
+                  ColorScheme.Normal);
```

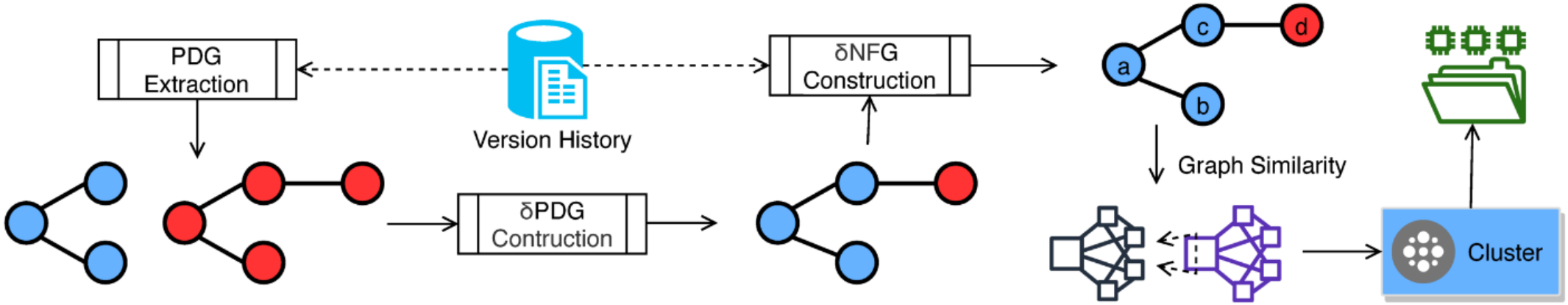
Why care about tangled commits?

Because atomic commits

Enable bug localisation techniques, as simple as `git bisect`, or as complex as statistical and neural approaches.

Provide a cleaner interlinking between feature implementations, and their histories, to requirements.

Flexeme at a Glance



Flexeme

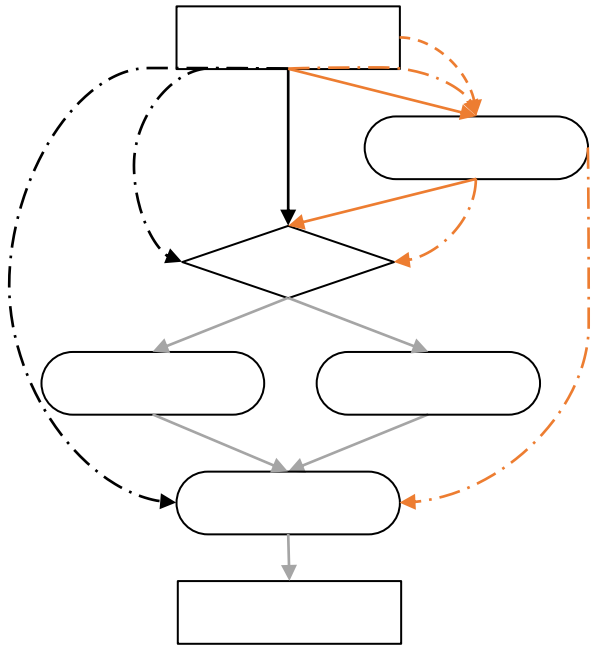
Start from Le and Pattison's MVICFG[1].

Augment with Data-flows to get to the δ-PDG.

Augment with Name-flows to get to the δ-NFG.

Separate the δ-NFG into a collection of graphs.

Re-cluster using graph similarity.



[1] Wei Le and Shannon D. Pattison. 2014. Patch verification via Multiversion Interprocedural Control Flow Graphs. Proc. 36th Int. Conf. Softw. Eng. - ICSE 2014(2014), 1047–1058. https://doi.org/10.1145/2568225.2568304

Weisfeiler-Lehman Graph Kernel

A graph kernel captures the notion of similarity – given a particular kernel, we can embed graphs into a vector space where the dot product captures our desired similarity.

Weisfeiler-Lehman (WL) is a meta-kernel based on the graph isomorphism test under the same name.

Employing the rooted sub-trees kernel under the WL framework captures the intuition, in our setting, of similar downstream “behaviours”.

Commit Untangling Baselines

State-of-the-art at the time of our writing:

- DU-chains based approach (Barnett et al.[2])
- Confidence Voting + Agglomerative Clustering (Herzig et al.[3,4])

Almost all information needed for both approaches is contained in the δ-NFG.

[2] Mike Barnett, Christian Bird, João Brunet, and Shuvendu K. Lahiri. 2015. Helping developers help themselves: Automatic decomposition of code review changesets. Proc. - Int. Conf. Softw. Eng.1, August 2014 (2015), 134–144. https://doi.org/10.1109/ICSE.2015.35
[3] Kim Herzig, Sascha Just, and Andreas Zeller. 2016. The impact of tangled code changes on defect prediction models. Empir. Softw. Eng.21, 2 (2016), 303–336. https://doi.org/10.1007/s10664-015-9376-6
[4] Kim Herzig and Andreas Zeller. 2013. The impact of tangled code changes. IEEE Int. Work. Conf. Min. Softw. Repos.(2013), 121–130. https://doi.org/10.1109/MSR.2013.6624018

Reproducing Barnett et al.

- Starting from the δ-PDG keep only dataflow.
- Separate flow by “kill” statements to obtain DU-chains.
- This is naturally intersected with diff-hunks by the δ-PDG construction.
- The result is DU-chains with some nodes annotated as changed.

Changes are related if:

1. They are both changed uses of the same definition.
2. One is a changed use and the other is its changed definition.

Reproducing Herzig et al.

The approach by Herzig et al. makes use of the following confidence voters: file distance, package distance, call graph, change couplings, data dependency.

We adapt package distance to namespace distance as our projects are in C#.

We precompute co-occurrence matrices for files for change couplings.

Experimental Set-up

We focus on comparing the untangling method and not the construction of auxiliary structures:

we only consider untangling time for runtime results.

We measure accuracy as:

$$A = \frac{\#\text{Correctly labeled nodes}}{\#\text{Nodes in graph}}.$$

We always apply the Hungarian Algorithm to find a maximal accuracy permutation of labels.

Corpus Generation

We consider commits that:

1. Have been committed by the same developer within 14 days of each other with no other commit by the same developer in between them.
2. Change namespaces whose names have a large prefix match.
3. Contain files that are frequently changed together.
4. Do not contain certain keywords (such as ‘fix’, ‘bug’, ‘feature’, ‘implement’) multiple times.

We then attempt to `git cherry-pick` the selected candidates on the parent of the chain.

Corpus Statistics

Table 1: Project statistics. The last revision indicates the commit at which we performed the 'git clone'.

Project	LOC	# of Commits	Last revision
Commandline	11602	1556	67f77e1
CommonMark	14613	418	f3d5453
Hangfire	40263	2889	175207c
Humanizer	56357	1647	604ebcc
Lean	242974	7086	71bc0fa
Nancy	79192	5497	dbdbe94
Newtonsoft.Json	71704	299	4f8832a
Ninject	13656	784	6a7ed2b
RestSharp	16233	1440	b52b9be

Table 2: Successfully tangled commits.

Project	Concerns		
	2	3	Overall
Commandline	308	32	340
CommonMark	52	0	52
Hangfire	229	87	316
Humanizer	85	4	89
Lean	154	24	178
Nancy	284	67	351
Newtonsoft.Json	84	7	91
Ninject	82	0	82
RestSharp	95	18	113
Overall	1373	239	1612

Results: Accuracy

Project Name	Barnett *et al.* [2]			Herzig *et al.* [6]		
	2	3	Overall	2	3	Overall
Commandline	0.18	0.21	0.19	0.67	0.48	0.64
CommonMark	0.20	*	0.20	0.65	*	0.65
Hangfire	0.16	0.13	0.15	0.70	0.54	0.64
Humanizer	0.18	0.31	0.18	0.64	0.42	0.62
Lean	0.19	0.12	0.18	0.69	0.62	0.69
Nancy	0.09	0.08	0.09	0.70	0.56	0.67
Newtonsoft.Json	0.15	0.11	0.15	0.71	0.56	0.71
Ninject	0.14	*	0.14	0.57	*	0.57
RestSharp	0.12	0.14	0.12	0.71	0.69	0.70
Overall	0.14	0.11	0.13	0.69	0.62	0.67

Results: Runtime

Project Name	Barnett *et al.* [2]			Herzig *et al.* [6]		
	2	3	Overall	2	3	Overall
Commandline	**0.10**	**8.51**	**0.12**	10.51	8.56	9.26
CommonMark	**2.56**	*	**2.56**	1.96×10^{3}	*	1.96×10^{3}
Hangfire	**1.15**	**4.97**	**1.95**	1.30×10^{4}	5.57×10^{4}	1.72×10^{4}
Humanizer	**0.44**	**0.23**	**0.41**	40.62	49.53	44.44
Lean	**1.00**	**1.59**	**1.28**	345.05	173.58	288.35
Nancy	**2.06**	**5.63**	**2.42**	570.57	1.29×10^{3}	600.16
Newtonsoft.Json	**2.14**	**6.42**	**2.35**	225.62	510.77	230.49
Ninject	**1.25**	*	**1.25**	81.53	*	81.53
RestSharp	**0.74**	1.25	**0.78**	46.22	222.98	72.09
Overall	**1.02**	**5.02**	**1.41**	81.53	647.99	117.35

Results: Accuracy

Project Name	Barnett *et al.* [2]			Heddle (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	0.18	0.21	0.19	**0.82**	**0.92**	**0.82**
CommonMark	0.20	*	0.20	0.70	*	0.70
Hangfire	0.16	0.13	0.15	**0.86**	0.68	0.79
Humanizer	0.18	0.31	0.18	**0.83**	**0.57**	**0.81**
Lean	0.19	0.12	0.18	0.77	**0.82**	0.80
Nancy	0.09	0.08	0.09	0.81	**0.92**	0.84
Newtonsoft.Json	0.15	0.11	0.15	0.71	0.52	0.71
Ninject	0.14	*	0.14	0.80	*	0.80
RestSharp	0.12	0.14	0.12	**0.82**	**0.89**	**0.82**
Overall	0.14	0.11	0.13	0.81	**0.84**	0.81

Results: Accuracy

Project Name	Herzig *et al.* [6]			HEDDLE (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	0.67	0.48	0.64	**0.82**	**0.92**	**0.82**
CommonMark	0.65	*	0.65	0.70	*	0.70
Hangfire	0.70	0.54	0.64	**0.86**	0.68	0.79
Humanizer	0.64	0.42	0.62	**0.83**	**0.57**	**0.81**
Lean	0.69	0.62	0.69	0.77	**0.82**	0.80
Nancy	0.70	0.56	0.67	0.81	**0.92**	0.84
Newtonsoft.Json	0.71	0.56	0.71	0.71	0.52	0.71
Ninject	0.57	*	0.57	0.80	*	0.80
RestSharp	0.71	0.69	0.70	**0.82**	**0.89**	**0.82**
Overall	0.69	0.62	0.67	0.81	**0.84**	0.81

Results: Runtime

Project Name	Barnett *et al.* [2]			Heddle (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	**0.10**	**8.51**	**0.12**	0.85	182.62	1.04
CommonMark	**2.56**	*	**2.56**	14.95	*	14.95
Hangfire	**1.15**	**4.97**	**1.95**	8.06	45.29	11.64
Humanizer	**0.44**	**0.23**	**0.41**	4.86	2.56	4.58
Lean	**1.00**	**1.59**	**1.28**	18.07	24.07	18.23
Nancy	**2.06**	**5.63**	**2.42**	18.38	85.55	21.78
Newtonsoft.Json	**2.14**	**6.42**	**2.35**	8.01	11.98	8.58
Ninject	**1.25**	*	**1.25**	14.99	*	14.99
RestSharp	**0.74**	1.25	**0.78**	9.86	26.25	10.11
Overall	**1.02**	**5.02**	**1.41**	7.99	43.29	9.56

Results: Runtime

Project Name	Herzig *et al.* [6]			HEDDLE (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	10.51	8.56	9.26	0.85	182.62	1.04
CommonMark	1.96×10^3	*	1.96×10^3	14.95	*	14.95
Hangfire	1.30×10^4	5.57×10^4	1.72×10^4	8.06	45.29	11.64
Humanizer	40.62	49.53	44.44	4.86	2.56	4.58
Lean	345.05	173.58	288.35	18.07	24.07	18.23
Nancy	570.57	1.29×10^3	600.16	18.38	85.55	21.78
Newtonsoft.Json	225.62	510.77	230.49	8.01	11.98	8.58
Ninject	81.53	*	81.53	14.99	*	14.99
RestSharp	46.22	222.98	72.09	9.86	26.25	10.11
Overall	81.53	647.99	117.35	7.99	43.29	9.56

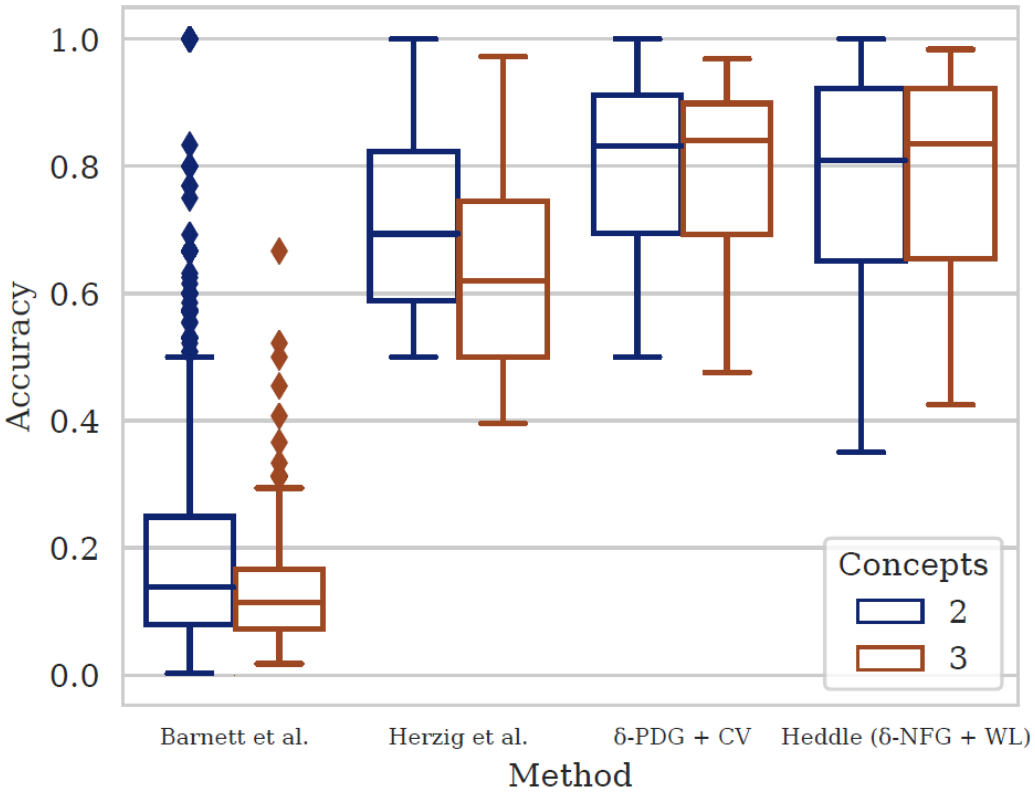
Results: Accuracy

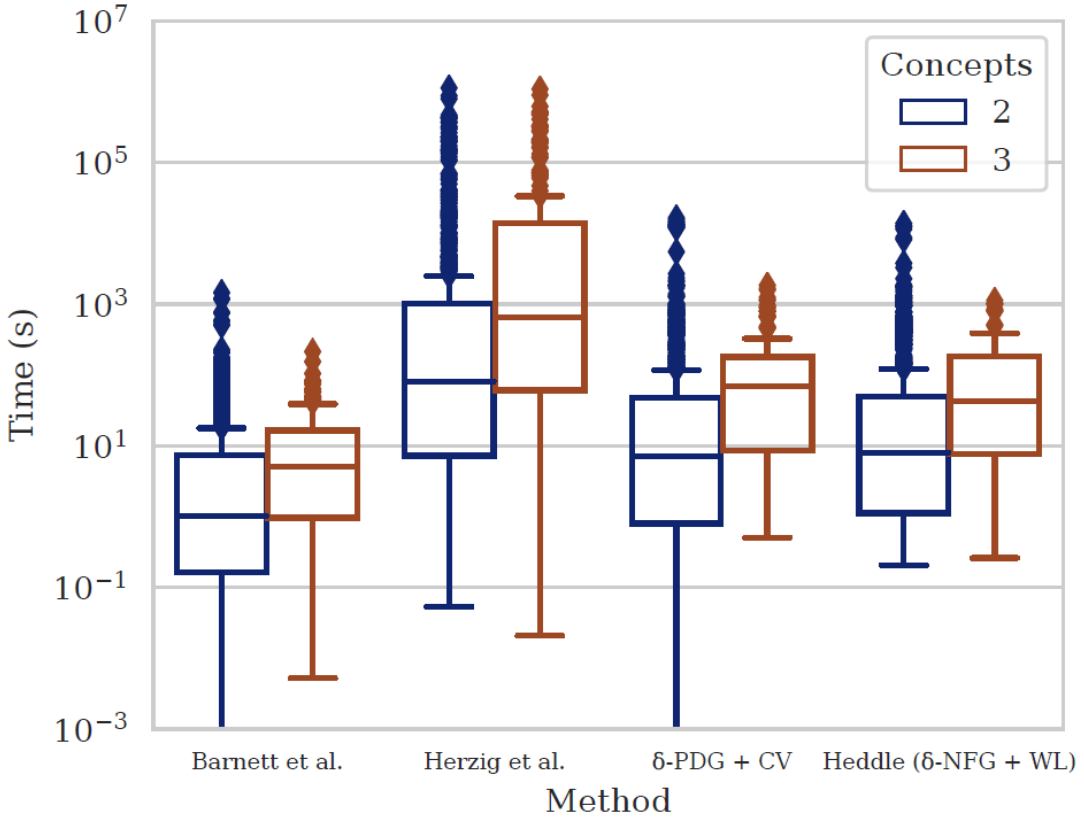
Project Name	δ-PDG+CV			HEDDLE (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	0.77	0.84	0.80	**0.82**	**0.92**	**0.82**
CommonMark	**0.90**	*	**0.90**	0.70	*	0.70
Hangfire	0.84	**0.88**	**0.87**	**0.86**	0.68	0.79
Humanizer	0.69	x	0.69	**0.83**	**0.57**	**0.81**
Lean	**0.84**	0.71	**0.84**	0.77	**0.82**	0.80
Nancy	**0.86**	0.80	**0.86**	0.81	**0.92**	0.84
Newtonsoft.Json	**0.86**	**0.69**	**0.82**	0.71	0.52	0.71
Ninject	**0.94**	*	**0.94**	0.80	*	0.80
RestSharp	0.74	0.53	0.70	**0.82**	**0.89**	**0.82**
Overall	**0.83**	**0.84**	**0.83**	0.81	**0.84**	0.81

Results: Runtime

Project Name	δ-PDG+CV			HEDDLE (δ-NFG + WL)		
	2	3	Overall	2	3	Overall
Commandline	0.42	153.55	0.59	0.85	182.62	1.04
CommonMark	10.38	*	10.38	14.95	*	14.95
Hangfire	10.61	123.99	13.84	8.06	45.29	11.64
Humanizer	9.24	x	9.24	4.86	2.56	4.58
Lean	19.28	17.08	19.28	18.07	24.07	18.23
Nancy	22.04	20.52	21.96	18.38	85.55	21.78
Newtonsoft.Json	20.04	51.54	20.32	8.01	11.98	8.58
Ninject	4.52	*	4.52	14.99	*	14.99
RestSharp	7.80	**1.01**	4.99	9.86	26.25	10.11
Overall	7.17	70.35	10.27	7.99	43.29	9.56

Results: Overview





Future Directions

In Flexeme, we still make use of agglomerative clustering which is common in untangling literature, it is left to future work to explore the design space and consider other clustering approaches.

We employ our multiversion Program Dependency Graph and Name-flow graphs for untangling tasks; however, nothing precludes them from more typical static analysis work on multiple versions.

Tangled Commits

The features implemented by these two (atomic) patches are unrelated.

Tangled patches introduce bias.

```
@@ -127,31 +137,32 @@ namespace Terminal {
 {
  this.barItems = barItems;
  this.host = host;
+ ColorScheme = Colors.Menu;                                   (1a)
  CanFocus = true;
 }
 public override void Redraw(Rect region)
 {
-  Driver.SetAttribute(Colors.Menu.Normal);                    (1b)
-  DrawFrame(region, true);
+  Driver.SetAttribute(ColorScheme.Normal);
+  DrawFrame(region, padding: 0, fill: true);
   for (int i = 0;
        i < barItems.Children.Length;
        i++)
   {
    var item = barItems.Children [i];
    Move(1, i+1);
-   Driver.SetAttribute(
-     item == null ? Colors.Base.Focus :                        (1c)
-     i == current ? Colors.Menu.Focus :
-     Colors.Menu.Normal
-   );
+   Driver.SetAttribute(
+     item == null ? Colors.Base.Focus :
+     i == current ? ColorScheme.Focus :
+     ColorScheme.Normal
+   );
```

```
for (int p = 0; p < Frame.Width-2; p++)
 if (item == null)
-  Driver.AddSpecial(SpecialChar.HLine);                       (2a)
+  Driver.AddRune(Driver.HLine);
 else
-  Driver.AddCh(' ');                                          (2b)
+  Driver.AddRune(' ');
 if (item == null)
  continue;
 Move(2, i + 1);
 DrawHotString(item.Title,
-  i == current? Colors.Menu.HotFocus :                        (1d)
-               Colors.Menu.HotNormal,
-  i == current ? Colors.Menu.Focus :
-                 Colors.Menu.Normal);
+  i == current? ColorScheme.HotFocus :
+               ColorScheme.HotNormal,
+  i == current ? ColorScheme.Focus :
+                 ColorScheme.Normal);
```

Flexeme

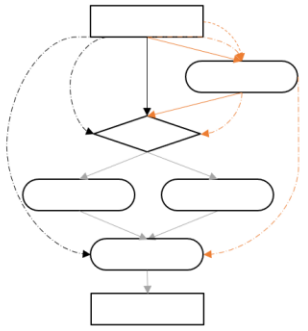
Start from Le and Pattison's MVICFG[1].

Augment with Data-flow to get to the δ-PDG.

Augment with Name-flows to get to the δ-NFG.

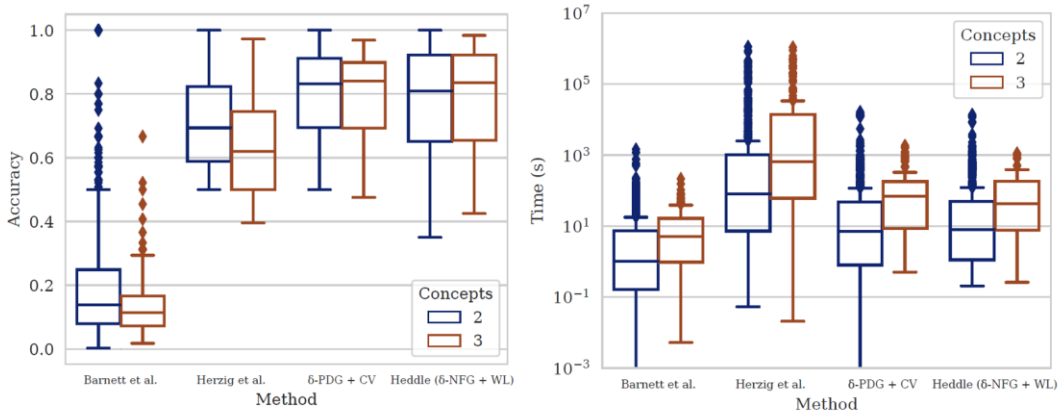
Separate the δ-NFG into a collection of graphs.

Re-cluster using graph similarity.



[1] Wei Le and Shannon D. Pattison. 2014. Patch verification via Multiversion Interprocedural Control Flow Graphs. Proc. 36th Int. Conf. Softw. Eng. - ICSE 2014(2014), 1047–1058. https://doi.org/10.1145/2568225.2568304

Results: Overview



You can find Flexeme at:
https://pppi.github.io/Flexeme