

Fine-grained Provenance Collection over Scripts Through Program Slicing

Fine-grained Provenance Collection over Scripts Through Program Slicing

<https://github.com/gems-uff/noworkflow>

João Felipe Pimentel (UFF), Juliana Freire (NYU), Leonardo Murta (UFF), Vanessa Braganholo (UFF)



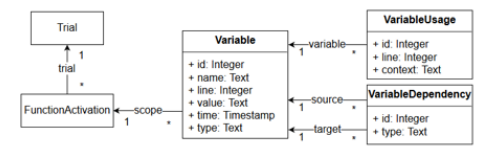
Motivation

- Most existing approaches for capturing provenance of scripts either require annotations or capture provenance at coarse-grain
- Files, functions
- False-positives: "Should show depend on process?"

Solution

- Dynamic program slicing, with configurable depth
- Combine Python's tracer and profiler
- Capture variable dependencies and values

Model



noWorkflow

- Transparently captures provenance of Python scripts: no changes required!
- Allows users to analyze provenance data: SQL queries, Prolog queries, graph visualizations, Jupyter Notebook.
- Install: `$ pip install noworkflow[all]`
- Run: `$ now run -e Tracker happy.py`

```
1| DRY_RUN = ...
2|
3| def process(number):
4|     while number >= 10:
5|         new_number, str_number = 0, str(number)
6|         for char in str_number:
7|             new_number += int(char) ** 2
8|         number = new_number
9|     return number
10|
11| def show(number):
12|     if number not in (1, 7):
13|         return "unhappy number"
14|     return "happy number"
15|
16| n = 2 ** 4000
17| final = process(n)
18| if DRY_RUN:
19|     final = 7
20| print(show(final))
```

Queries

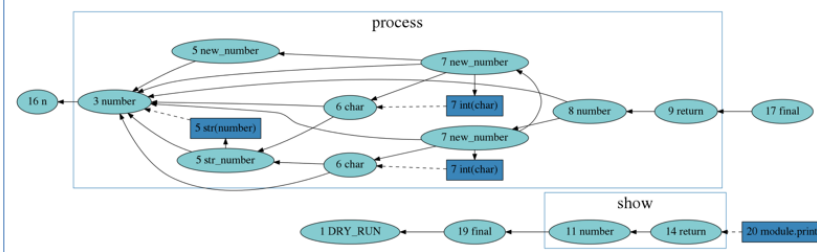
```
SELECT v2.name, v2.line
FROM slicing_variable v, slicing_dependency d,
slicing_variable v2
WHERE d.dependent = v.id AND d.trial_id = v.trial_id
AND d.supplier = v2.id AND d.trial_id = v2.trial_id
AND v2.name = "process" AND v2.line = 17
AND d.trial_id = 1;
```

SQL query: which variables depend on "process" in line 17.
The result is "final" in line 17, without transitive dependencies.

```
1 7= variable(1, _, Id, 'final', 17, _, '_')
variable_dependencies(1, Id, X).
Id = 24,
X = [variable(1, 1, 24, final, 17, '1', 1455903187.908933),
variable(1, 1, 29, 'call process', 17, 'now(n/a)',
1455903187.908886),
variable(1, 2, 22, return, 9, '1', 1455903187.908853),
... variable(1, 1, 4, n, 16, '10', 1455903187.908002)]
```

Prolog query: "final" in line 17 depends on which variables?
The result is a list of transitive dependencies.

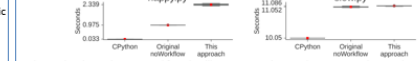
Dataflow: `$ now dataflow -m simulation --rank-line | dot -Tpng fig.png`



Limitations

- Do not track dependencies on complex data structures and some syntactic constructs, such as list, objects (e.g. file handlers), exceptions, and generators
- Python sub-set: one line per statement and one statement per line
- Visualization may not be well suited for huge dependency graphs

Overhead



- slow.py invokes a slow external function like most experiments do, compensating overhead

jpimentel@ic.uff.br
github.com/gems-uff/noworkflow

Provenance of Scripts

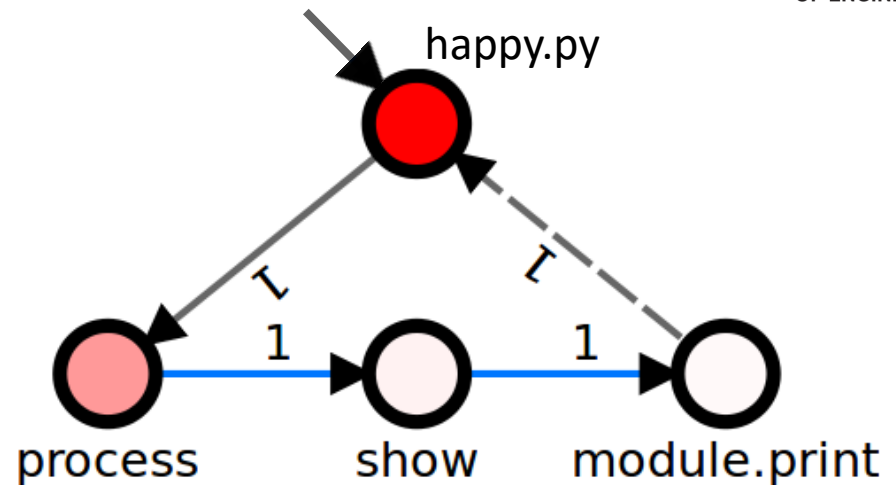
- YesWorkflow
 - McPhillips et al., 2015
- API
 - Bochner; Gude; Schreiber, 2008
- StarFlow
 - Angelino; Yamins; Seltzer, 2010
- RDataTracker
 - Lerner and Boose, 2014
- noWorkflow
 - Murta et al., 2014
- Sumatra
 - Davison, 2012
- LLVM compiler
 - Tariq et al., 2012
- DataTracker
 - Stamatogiannakis et al., 2014
- ...

Provenance of Scripts

- YesWorkflow
 - McPhillips et al., 2015
- noWorkflow
 - Murta et al., 2014
- Coarse-grained Provenance
 - Files
 - Functions
 - ...
- Seltzer, 2010
- RDataTracker
 - Stamatogiannakis et al., 2014
 - Lerner and Boose, 2014
 - ...

```
1 | DRY_RUN = ...
2 |
3 | def process(number):
4 |     ...
10 |
11 | def show(number):
12 |     ...
15 |
16 | n = 10
17 | final = process(n)
18 | if DRY_RUN:
19 |     final = 7
20 | print(show(final))
```

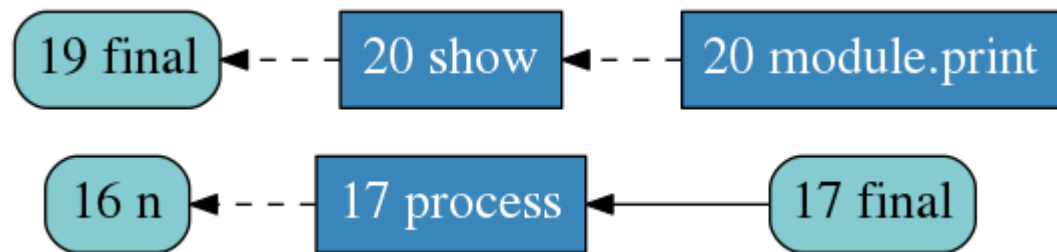
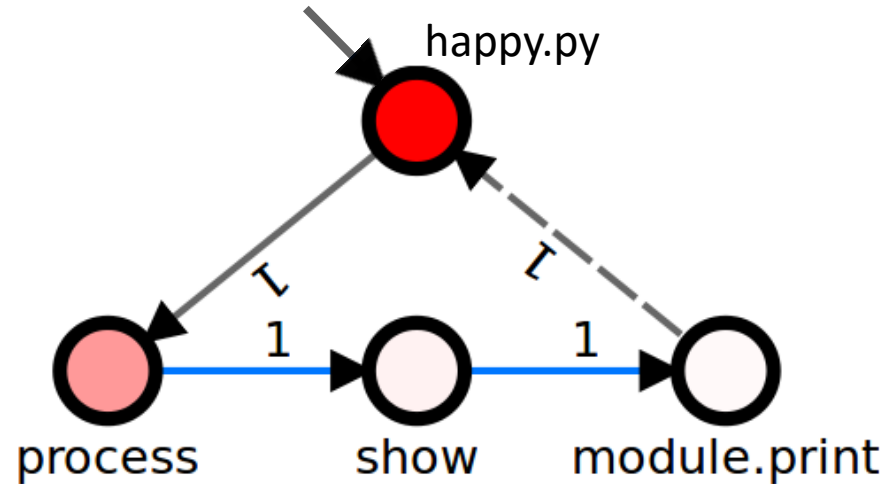
```
1 | DRY_RUN = ...
2 |
3 | def process(number):
4 |     ...
10 |
11 | def show(number):
12 |     ...
15 |
16 | n = 10
17 | final = process(n)
18 | if DRY_RUN:
19 |     final = 7
20 | print(show(final))
```



```

1 | DRY_RUN = True
2 |
3 | def process(number):
4 |     ...
10 |
11 | def show(number):
12 |     ...
15 |
16 | n = 10
17 | final = process(n)
18 | if DRY_RUN:
19 |     final = 7
20 | print(show(final))

```



Fine-grained Provenance Collection over Scripts Through Program Slicing

Fine-grained Provenance Collection over Scripts Through Program Slicing

<https://github.com/gems-uff/noworkflow>

João Felipe Pimentel (UFF), Juliana Freire (NYU), Leonardo Murta (UFF), Vanessa Braganholo (UFF)



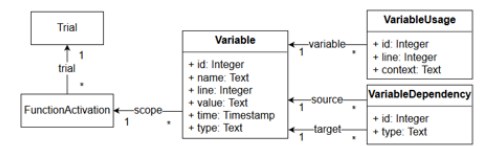
Motivation

- Most existing approaches for capturing provenance of scripts either require annotations or capture provenance at coarse-grain
- Files, functions
- False-positives: "Should show depend on process?"

Solution

- Dynamic program slicing, with configurable depth
- Combine Python's tracer and profiler
- Capture variable dependencies and values

Model



noWorkflow

- Transparently captures provenance of Python scripts: no changes required!
- Allows users to analyze provenance data: SQL queries, Prolog queries, graph visualizations, Jupyter Notebook.
- Install: `$ pip install noworkflow[all]`
- Run: `$ now run -e Tracker happy.py`

```
1| DRY_RUN = ...
2|
3| def process(number):
4|     while number >= 10:
5|         new_number, str_number = 0, str(number)
6|         for char in str_number:
7|             new_number += int(char) ** 2
8|         number = new_number
9|     return number
10|
11| def show(number):
12|     if number not in (1, 7):
13|         return "unhappy number"
14|     return "happy number"
15|
16| n = 2 ** 4000
17| final = process(n)
18| if DRY_RUN:
19|     final = 7
20| print(show(final))
```

Queries

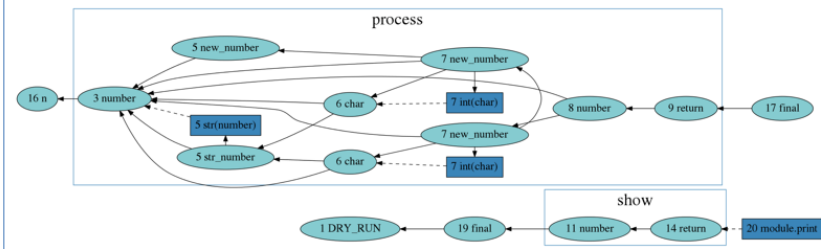
```
SELECT v2.name, v2.line
FROM slicing_variable v, slicing_dependency d,
slicing_variable v2
WHERE d.dependent = v.id AND d.trial_id = v.trial_id
AND d.supplier = v2.id AND d.trial_id = v2.trial_id
AND v2.name = "process" AND v2.line = 17
AND d.trial_id = 1;
```

SQL query: which variables depend on "process" in line 17.
The result is "final" in line 17, without transitive dependencies.

```
1 7= variable(1, _, Id, 'final', 17, _, '_')
variable_dependencies(1, Id, X).
Id = 24,
X = [variable(1, 1, 24, final, 17, '1', 1455903187.908933),
variable(1, 1, 29, 'call process', 17, 'now(n/a)',
1455903187.908886),
variable(1, 2, 22, return, 9, '1', 1455903187.908853),
... variable(1, 1, 4, n, 16, '10', 1455903187.908002)]
```

Prolog query: "final" in line 17 depends on which variables?
The result is a list of transitive dependencies.

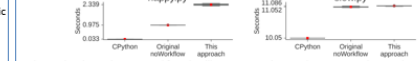
Dataflow: `$ now dataflow -m simulation --rank-line | dot -Tpng fig.png`



Limitations

- Do not track dependencies on complex data structures and some syntactic constructs, such as list, objects (e.g. file handlers), exceptions, and generators
- Python sub-set: one line per statement and one statement per line
- Visualization may not be well suited for huge dependency graphs

Overhead



- slow.py invokes a slow external function like most experiments do, compensating overhead

jpimentel@ic.uff.br
github.com/gems-uff/noworkflow