Dataflow Analysis: Part 2

October 6, 2023

Check List

Define semi-lattice

Direction: forward / backward.

Check the order (does it make sense?)

Decide initial values.

Design the transfer functions.

How does each statement affect the dataflow facts?

Prove monotonicity.

Review: Monotone Frameworks

	Available Expressions	Reaching Definitions	Very Busy Expressions	Live Variables
L	$\mathcal{P}(\mathbf{AExp}_\star)$	$\mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}^?_\star)$	$\mathcal{P}(\mathbf{AExp}_\star)$	$\mathcal{P}(\mathbf{Var}_\star)$
$\sqsubseteq$	$\supseteq$	$\subseteq$	$\supseteq$	$\subseteq$
$\bigsqcup$	$\bigcap$	$\bigcup$	$\bigcap$	$\bigcup$
$\bot$	$\mathbf{AExp}_\star$	$\emptyset$	$\mathbf{AExp}_\star$	$\emptyset$
ι	$\emptyset$	$\{(x,?) \mid x \in FV(S_\star)\}$	$\emptyset$	$\emptyset$
E	$\{init(S_\star)\}$	$\{init(S_\star)\}$	$final(S_\star)$	$final(S_\star)$
F	$flow(S_\star)$	$flow(S_\star)$	$flow^R(S_\star)$	$flow^R(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$			
f_ℓ	$f_\ell(l) = (l \setminus kill([B]^\ell)) \cup gen([B]^\ell)$ where $[B]^\ell \in blocks(S_\star)$			

Implementing Analyses in Tundra

Create new file in `analysis/` folder:

- Define `AnalysisType`
- Create a subclass of abstract `Analysis` class (`analysis/analysis.py`)
- Define methods: `initial_in`, `initial_out`, and `get_new_values`
- Useful functions in `lang/utils.py`

Example: Live Variables Analysis

A variable is **live** at the exit from a label if there exists a path from the label to a use of the variable that does not re-define the variable.

Check List: Live Variables Analysis

Define semi-lattice

Direction: forward / backward.

Check the order (does it make sense?)

Decide initial values.

Design the transfer functions.

How does each statement affect the dataflow facts?

Prove monotonicity.

Example: Live Variables Analysis

	Live Variables
L	$\mathcal{P}(\mathbf{Var}_\star)$
$\sqsubseteq$	$\subseteq$
$\sqcup$	$\cup$
$\bot$	$\emptyset$
ι	$\emptyset$
E	$\mathit{final}(S_\star)$
F	$\mathit{flow}^R(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$
f_ℓ	$f_\ell(l) = (l \setminus \mathit{kill}([B]^\ell)) \cup \mathit{gen}([B]^\ell)$ **where** $[B]^\ell \in \mathit{blocks}(S_\star)$

```
AnalysisType = Set[Variable]


14 usages   Avery Laird *
class LiveVariables(Analysis[AnalysisType]):
    T = AnalysisType

    Avery Laird *
    def initial_in(self, node: Node) -> T:
        return set()

    Avery Laird *
    def initial_out(self, node: Node) -> T:
        return set()
```

Tundra Demo

Example: Reaching Definitions

For each program point, which assignments may have been made and not overwritten, when program execution reaches this point along some path.

```
l1: x = 0;
l2: y = 10;
```

Check List: Reaching Definitions

Define semi-lattice

Direction: forward / backward.

Check the order (does it make sense?)

Decide initial values.

Design the transfer functions.

How does each statement affect the dataflow facts?

Prove monotonicity.

Example: Reaching Definitions

	Reaching Definitions
L	$\mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star^?)$
$\sqsubseteq$	$\subseteq$
$\bigsqcup$	$\bigcup$
$\bot$	$\emptyset$
ι	$\{(x,?) \mid x \in FV(S_\star)\}$
E	$\{init(S_\star)\}$
F	$flow(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$
f_ℓ	$f_\ell(l) = (l \setminus kill([B]^\ell)) \cup gen([B]^\ell)$ where $[B]^\ell \in blocks(S_\star)$

```
AnalysisType = Set[Tuple[Variable, Optional[Node]]]


6 usages   Avery Laird *
class ReachingDefinitions(Analysis[AnalysisType]):
    Avery Laird
    def initial_in(self, node: Node) -> AnalysisType:
        return set()

    Avery Laird
    def initial_out(self, node: Node) -> AnalysisType:
        return set()
```

Tundra Demo

Example: Very Busy Expressions

An expression is very busy at the exit from a label if, no matter what path is taken from the label, the expression must always be used before any of the variables occurring in it are redefined.

Example: Very Busy Expressions

An expression is very busy at the exit from a label if, no matter what path is taken from the label, the expression must always be used before any of the variables occurring in it are redefined.

```
if (a > b) {
  x = b - a;
  y = a - b;
} else {
  y = b - a;
  x = a - b;
}
```

Check List: Very Busy Expressions

Define semi-lattice

Direction: forward / backward.

Check the order (does it make sense?)

Decide initial values.

Design the transfer functions.

How does each statement affect the dataflow facts?

Prove monotonicity.

Example: Very Busy Expressions

	Very Busy Expressions
L	$\mathcal{P}(\mathbf{AExp}_\star)$
$\sqsubseteq$	$\supseteq$
$\bigsqcup$	$\bigcap$
$\bot$	$\mathbf{AExp}_\star$
ι	$\emptyset$
E	$final(S_\star)$
F	$flow^R(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$
f_ℓ	$f_\ell(l) = (l \setminus kill([B]^\ell)) \cup gen([B]^\ell)$ where $[B]^\ell \in blocks(S_\star)$

```
AnalysisType = Set[Expression]


2 usages  Avery Laird
class VeryBusyExpressions(Analysis[AnalysisType]):
    T = AnalysisType

    Avery Laird
    def initial_in(self, node: Node) -> T:
        return set()

    Avery Laird
    def initial_out(self, node: Node) -> T:
        return set()
```

Tundra Demo

Example: Available Expressions

For each program point, which expressions must have already been computed, and not later modified, on all paths to the program point.

Example: Available Expressions

For each program point, which expressions must have already been computed, and not later modified, on all paths to the program point.

```
    x = a + b;
l1: y = a + b;
```

Check List: Available Expressions

Define semi-lattice

Direction: forward / backward.

Check the order (does it make sense?)

Decide initial values.

Design the transfer functions.

How does each statement affect the dataflow facts?

Prove monotonicity.

Example: Available Expressions

	Available Expressions
L	$\mathcal{P}(\mathbf{AExp}_\star)$
$\sqsubseteq$	$\supseteq$
$\bigsqcup$	$\bigcap$
$\bot$	$\mathbf{AExp}_\star$
ι	$\emptyset$
E	$\{init(S_\star)\}$
F	$flow(S_\star)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g : f(l) = (l \setminus l_k) \cup l_g\}$
f_ℓ	$f_\ell(l) = (l \setminus kill([B]^\ell)) \cup gen([B]^\ell)$ where $[B]^\ell \in blocks(S_\star)$

Tundra Demo