Refresher
Data Flow Analysis

Given a program’s control flow graph, we can compute some facts about the program at different program locations.

E.g. Live Variable Analysis

```
i = 0;
j = 1;
while (i < 10)
{
i += j;
j = i;
}
j = 0;
```
Data Flow Analysis

- A family of algorithms
  - Different types of DFA (forward/backward, may/might)
  - All have the same general shape:
    - CFG nodes have values
    - Values are updated using transfer functions
    - Fixed point: values updated until they stop changing
- Approximates program behaviour
- Connection to lattices
  - Domain of node values = lattice domain
  - Transfer functions are monotonic (preserves order of lattice elements)
  - Termination of DFA due to finite height of lattice
Starter Code
Starter Code

Requirements:

Python (preferably >=3.9)

Download starter code here:

tinyurl.com/muuz9mjk

Check if it works:

pip3 install -r requirements.txt
pytest test/test_cfg.py
Grammar of Tundra, a toy language

program := statement*
statement := label? (assignment | condition | while_loop)
label := identifier "::
assignment := identifier "=" expression ";
condition := "if" "(" expression ")" 
               "{" statement* "}"
               "else" "{" statement* "}"
while_loop := "while" "(" expression ")" 
               "{" statement* "}"
...etc

For more info, see the file lang/tundra.lark
Grammar of Tundra, a toy language

```plaintext
i = 0;
j = 1;
l1: while (i < 10)
    {
        i += j;
j = i;
    }
j = 0;
```

```plaintext
l1: x = 10;
l2: y = 10;
    if (x == 10) {
        y = 10;
    } else {
    }
x = ((y + 0) * 20) % z
```
Tundra Analysis Framework

Program Text

Parser → Parse Tree → Transformer → AST → To CFG

CFG → Fixed Point → Instance of Analysis Class

Domain (AnalysisType) + Initial States + Transfer Functions

Map from CFG nodes to objects of type AnalysisType
A CFG is a directed graph with 3 types of Node: Entry, InnerNode, Exit.

Example CFG.

```plaintext
l1: x = 10;
l2: y = 10;
    if (x == 10) {
        y = 3;
    } else {
    }
```
How to create a new analysis?

- Create a new file in folder `analysis/
- In the new file:
  - Define AnalysisType
  - Create a subclass of generic abstract class Analysis (defined in `analysis/analysis.py`) parameterized on AnalysisType
  - Define methods `initial_in, initial_out` and `get_new_values`
  - Note: you may want to use utility functions `getStatementUsedVars` or `getExpressionUsedVars` (defined in `lang/utils.py`)
How to use the new analysis?

- Create CFG using `cfg = fileToCFG(sys.argv[1])` or `cfg = stringToCFG(s)`
- Apply fixed point algorithm, e.g. `In, Out = fixed_point(cfg, LiveVariables(cfg))`
- Note: whether you want to use `In` or `Out` depends on the goal of the analysis you are using.
- Extract information from the result:
  - For labelled statements, e.g. `getValueByLabel("label", In)`
  - Get values at Entry and Exit nodes: `getEntryValue(In/Out), getExitValue(In/Out)`
Live Variables Analysis

A backward “may” analysis
Live Variables Analysis

“Is it possible for this variable be used before being redefined?”
Is it possible for $x$, $y$, and $z$ to be used before being redefined?
Live Variables Analysis

Domain (AnalysisType):

Powerset(Variables)

Initial Values:

Empty set

Transfer Function:

\[
LV_{entry}(l) = LV_{exit}(l) \setminus \text{write}(l) \cup \text{read}(l)
\]

\[
LV_{exit}(l) = \bigcup_{l \rightarrow l' \in E} LV_{entry}(l')
\]

(Note: we call \text{write}(l) the \textbf{Kill} set and \text{read}(l) the \textbf{Gen} set.)
Live Variables Analysis

- Edit the file `analysis/live_variables_incomplete.py`
- Replace every “...” with Python code
  - Sometimes, “…” can just be “pass” – there’s nothing to be done there
  - Useful code snippets:
    - Get used variables with `getStatementUsedVars(node.statement)`
    - Union all sets in a list of sets: `set().union(*list_of_sets)`
    - `s1.union(s2), s1.difference(s2)`
- When you are done, try it out!
  - Rename `analysis/live_variables_incomplete.py` to `analysis/live_variables.py`
  - Look at `examples/lv1.tundra`
  - Think about this: before and after label `l1`, which variables can be used before being redefined?
  - Run `python3 ./main.py examples/lv1.tundra`
  - Compare your output with your colleagues
  - Once this is working, run `pytest -k "live_variables" -v`
Reaching Definitions Analysis

A forward “may” analysis
Reaching Definitions Analysis

“Which definitions might define the current variable values?”
Reaching Definitions Analysis

```java
if (x == 0) {
    y = x;
    y = 0;
} else {
}
y = y + 1;
x = y;
```

(y, “y = 0;”)
(y, “y = y + 1;”)
Reaching Definitions Analysis

```javascript
x = undefined;
y = undefined;

if (x == 0) {
  y = x;
  y = 0;
} else {
}
y = y + 1;

x = y;
```

reachability analysis:
- (x, "x = undefined;")
- (y, "y = 0;")
- (y, "y = undefined;")
- (x, "x = undefined;")
- (y, "y = y + 1;")

Reaching Definitions Analysis

**Domain (AnalysisType):**

Variables X Nodes

**Initial Values:**

Empty set

(A variant of this analysis assigns Variables X {⊥} as the out value of the Entry node, both initially and subsequently. Hint to generate this set: {(var, None) for var in self.program.variables})

**Transfer Function:**

\[
\begin{align*}
RD_{entry}(l) &= \bigcup_{l' \rightarrow l \in E} RD_{exit}(l') \\
RD_{exit}(l) &= RD_{entry}(l) \setminus Kill(l) \cup Gen(l)
\end{align*}
\]

If \( l \) is an assignment to \( v \), \( Kill(l) \) are the other assignments to \( v \) and \( Gen(l) \) is \((v, l)\)
What is the point of this analysis?

E.g.

1. A def of var v does not reach a use of var v: that def is dead code
2. If var v is used at location l, but \((v, \bot)\) is a reaching definition at location l, then we have detected the possible use of an undefined variable
Using what you learned in the section on Live Variables Analysis,

- Implement the Reaching Definitions analysis
- Check your implementation:
  - Look at examples/rd1.tundra and think about what the In/Out for label 11 should be
  - Update main.py to use your new analysis class
  - Run it: python3 ./main.py examples/rd1.tundra
- Write tests to check other scenarios
- Please discuss freely with your peers
What is the point of this analysis?

E.g.

1. A def reaches nowhere: dead code
2. If var v is used at location l, but \((v, \perp)\) is a reaching definition at location l, then we have detected the possible use of an undefined variable

Sanity check your implementation by writing some example files and see if you can detect 1. and 2.