1 Checkpoint

We have mostly met the target for the checkpoint that we specified in the proposal. Important we decided to limit ourselves to a subset of actual Ocaml code. Our current subset is very basic, but we intend to use it as a launching framework of sorts to expand into more complex things, while allowing us to test the complete analysis framework fairly quickly.

Here's is a list of what we've accomplished so far:

- We figured out how to use the `compiler-libs` library to parse and generate the AST for the OCaml code.
- We constructed a new, simpler AST, and wrote a way to convert Ocaml AST into our new AST.
- Using homework 3 as a reference point, we implemented a control flow graph, which is constructed from our simplified AST.
- Also using homework 3 as a reference, we started to fill out the dataflow analysis (this time for constant propagation)

We haven't tested the given code, thus it will not likely compile.

What we need to do to continue is to implement the flow function, then implement kildall's. This will give us a way to fully test our simplified language. Then we will expand out the language to include more Ocaml constructs.

Below we define an expression in our simplified Ocaml language. We have hard-implemented integers, as well as arithmetic on it. We don't include instructions in this pdf as their are only bindings (at this point)/

$$
\begin{aligned}
E ::= \ & \texttt{Var}\ x \\
| \ & \texttt{Const}\ n \\
| \ & \texttt{App}\ (E_1, E_2) \\
| \ & \texttt{Lam}\ (x, E) \\
| \ & \texttt{Add}\ (E_1, E_2) \\
| \ & \texttt{Sub}\ (E_1, E_2) \\
| \ & \texttt{Mul}\ (E_1, E_2) \\
| \ & \texttt{Div}\ (E_1, E_2)
\end{aligned}
$$