Exercise Sheet 5

Compiler Design Lab
Prof. Dr. Sebastian Hack
Fabian Ritter, B.Sc.

Exercise 5.1 Monotonicity and Expansiveness

In the lecture we saw the following two properties of a function $f : A \to A$ over some partially ordered set $(A, \sqsubseteq)$:

- Monotonicity: $\forall x, y \in A.\ x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$
- Expansiveness: $\forall x \in A.\ x \sqsubseteq f(x)$

Show that, in general, neither implies the other.

Exercise 5.2 Kleene Iteration

In the lecture, we have seen the Kleene iteration as a technique for finding fixed-points of a function:

Let $(L, \sqsubseteq)$ be a complete lattice without infinite ascending chains and let $f : L \to L$ be a monotone function. By computing $f^k(\bot)$ for increasing natural numbers k, we will eventually find a fixed-point $x^* = f^i(\bot) = f^{i+1}(\bot)$.

Show that the fixed-point x^* that we obtain from this procedure is the least fixed-point of f.

Exercise 5.3 Constant Propagation Analysis

Consider the following program S.

```
x = 1;
while (y > 0) {
  x = 2 - x;
  y = y - 1;
}
```

- Provide the control flow graph for program S.
- Perform the Constant Propagation Analysis on program S. In particular, provide the corresponding equation system for program S and perform a fixed-point iteration.
- Could the initial state be chosen differently? How will the results of your analysis change if so? Which advantages and disadvantages do you gain?

Project task D Pretty Printing

Up to now your compiler can already lex and parse input. In this project phase you will add a pretty printer to your compiler, i.e., functionality to generate nicely formatted source code from the parse tree. Therefore, you have to construct an abstract syntax tree (AST). Each AST node should be able to dump itself.

For a syntactically correct input program, `c4 --print-ast [file]` must print the output to stdout. This new switch must *augment* the `c4 --parse` functionality, i.e., its acceptance and rejection behavior as well as the respective return codes.

Download the example file from the link on the website. Invoking `c4 --print-ast ex05_pretty.c` should emit a char-exact copy of the source file. You can use the tool `diff` to compare your output to the reference file. Once again: *Your output must exactly match the reference output*.

You can use the pretty printer to debug your compiler. For example, the expression `a * b + d + e` should be dumped as `(((a * b) + d) + e)`. If you see a different output, your compiler handles associativity/operator precedence incorrectly.

In general, follow the reference file for placing newlines, spacing and so forth. The example mostly follows K&R style. Additionally, consider the following guidelines for your pretty printer:

- Print everything in the original order of the source file.

- Use a tab character (`'\t'`) for one level of indentation; do not use spaces.
- Do not wrap long lines.
- Do not print digraphs. Print the regular punctuator instead.
- Each (sub-)*expression* and each (sub-)*declarator*/*abstract-declarator* must be fully parenthesized, except for atoms consisting of at most one token like `0`, `'c'`, `"abc"`, *identifier*s and so forth. In particular, original parentheses are discarded. Do not re-associate any subexpressions.
- Do not hesitate to ask questions in the Forum.
- The soft deadline for this milestone is 2017-12-01.
- Keep it simple!