# Optimising Compilers Exercise Sheet 2

The purpose of this exercise sheet is to practise *register allocation*, *strength reduction*, *static single assignment*, *abstract interpretation* and *strictness analysis*.

## Questions

1. (a) Briefly describe the concept of *abstract interpretation*, making particular mention of safety.

   Consider a form of abstract interpretation in which our abstraction function captures the possible intervals of integer arithmetic. For example given a variable $x$ the abstraction function $\alpha$ returns the interval $[l_x, h_x]$ where $l_x$ is the lowest possible value of $x$ and $h_x$ is the highest possible value of $x$. For variables $x$ and $y$ the following properties hold for our abstraction function:

   $$\begin{aligned} f(x+y) &= [l_x + l_y, h_x + h_y] \\ f(x-y) &= [x_x - h_y, h_x - l_y] \end{aligned}$$

   (b) Given the following function calculate the interval of its return value in terms of the intervals of x and y.

   ```
   int g(x, y) {
       int a = x-y;
       int b = x+x;
       return b+a;
   }
   ```

   (c) An abstract interpretation of a program containing the function g ascertains the interval of the parameters to g as $\alpha(x) = [5, 10]$ and $\alpha(y) = [0, 2]$. Given this information can g return 0? Give the interval of g.

2. (a) Explain the notion of a 3-argument function being strict in its second parameter, considering both mathematical view of functions (an extra value $\perp$ representing nontermination), and the operational view of looping behaviour.

   (b) Do the functions $f(x, y) = f(x, y)$ and $g(x, y) = x + y$ have different strictness? Do they allow different strictness optimisations? Explain any differences between 'strict' in a parameter, and needing to evaluate it.

   Note: $f(x, y) = f(x, y)$ is to be understood as a recursive function.

   (c) Give the strictness function for

   ```
   f(x, y, z) = if x = 0 then y else y + z
   ```

   and justify it.

   (d) Consider a weaker form of strictness analysis where the abstract value of an $n$-argument function is just an $n$-argument bit vector where bit $k$ is 1 if and only if the concrete function is strict in parameter $k$. Why is this weaker? Give a program for which strictness optimisation optimises a parameter to call-by-value but which this weaker analysis fails to optimise.

3. (a) Describe the purpose of register allocation and how graph colouring can help.
   (b) Describe a possible downside of a graph colouring approach in the context of JIT compilers.
   (c) Research an alternative register allocation algorithm (*hint: linear scan*) and briefly contrast it with the graph colouring approach.

## Suggested past exam questions

2002 Paper 7 Question 4

2004 Paper 8 Question 7

2005 Paper 8 Question 7 Part (b)

## Relevant past exam questions

This section contains links to all past exam questions relevant to the topics covered in this supervision sheet. Note that some questions appear under multiple headings and / or on multiple exercise sheets when they cover more than one topic.

### Register allocation and clash graphs

- 2017 Paper 9 Question 10
- 2016 Paper 7 Question 12
- 2009 Paper 7 Question 12
- 2005 Paper 8 Question 7
- 2002 Paper 7 Question 4
- 1996 Paper 8 Question 7

### Code motion

- 2004 Paper 8 Question 7

### Static single assignment, strength reduction

- 2019 Paper 9 Question 11
- 2015 Paper 7 Question 11
- 2013 Paper 9 Question 9
- 2004 Paper 8 Question 7
- 2001 Paper 9 Question 7
- 1999 Paper 7 Question 4

**Abstract interpretation**

- 2018 Paper 9 Question 11
- 2007 Paper 9 Question 16

**Strictness analysis**

- 2016 Paper 9 Question 9
- 2013 Paper 9 Question 9
- 2012 Paper 9 Question 9
- 2010 Paper 7 Question 12
- 2009 Paper 9 Question 10
- 2005 Paper 9 Question 7
- 2003 Paper 9 Question 7
- 2001 Paper 8 Question 7
- 1999 Paper 8 Question 7
- 1996 Paper 9 Question 7
- 1995 Paper 9 Question 7