Cobo - Assignment 5

S.R.W. van Kampen en R.A.J. Wacanno

March 15, 2020

5.1 Static Single Assignment Form

```
i_0 = 0;
p_0 = i_0 < n_0
while (phi(p_0, p_1))
{
    j_0 = 0;
    q_0 = j_0 < m_0
    while (phi(q_0, q_1))
    {
        if (phi(i_0, i_1) < phi(j_0, j_1))
        {
            val_0 = phi(val_0, phi(val_1, val_1)) + phi(i_0, i_1) ;
        }
        else if (phi(j_0, j_1) == i_0)
        {
            val_1 = phi(val_0, phi(val_1, val_1)) - 1;
        }
        else
        {
            val_2 = phi(val_0, phi(val_1, val_1)) + phi(j_0, j_1);
        }
        j_1 = j_0 + 1;
        q_1 = j_1 < m_0
    }
    i_1 = phi(i_0, i_1) + 1;
    p_1 = i_1 < n_0
}
```

5.2 Machine-Independent Optimisation

```
i = 0;
j = 0;

if (i < j)
{
    while (i < n)
    {
        while (j < m)
        {
            val = val + i;
            j = j + 1;
        }
        i = i + 1;
    }
}
else if (j == i)
{
    while (i < n)
    {
        while (j < m)
        {
            val = val - 1;
            j = j + 1;
        }
        i = i + 1;
    }
}
else
{
    while (i < n)
    {
        while (j < m)
        {
            val = val + j;
            j = j + 1;
        }
        i = i + 1;
    }
}
```

5.3 Compilation schemes

Our first idea was to turn loops into recursive local functions and we ran into lots of issues, and then we realised that a do-while loop is *technically* not a while loop (and confirmed that, yes, turning the while loop into a do-while loop was an option). Here's our compilation scheme, which takes care of

also traversing into our loop body and traversing through the rest of the code after the loop:

$$\mathcal{C}\left[\!\!\left[\begin{array}{l}\text{while } (\mathit{Cond}) \ \{\ \mathit{Body}\ \} \\ \mathit{Rest}\end{array}\right]\!\!\right] \quad \Rightarrow \quad \begin{array}{l}\text{if } (\mathit{Cond}) \ \{ \\ \quad \text{do } \{ \\ \quad\quad \mathcal{C}[\![\mathit{Body}]\!] \\ \quad \} \text{ while } (\mathit{Cond}); \\ \} \\ \mathcal{C}[\![\mathit{Rest}]\!]\end{array}$$