Assignment serie 5

Jos Bonsink (10172920) & Mustafa Karaalioglu (10217665)

16 maart 2014

Consider the following CiviC code fragment:

```
i = 0;
while (i<n) {
    j = 0;
    while (j<n) {
        if (i<j) {
            A[i,j] = A[i,j] + i;
        }
        else if (j==i) {
            A[i,j] = -A[i,j];
        }
        else {
            A[i,j] = A[i,j] + j;
        }
        j = j + 1;
    }
    i = i + 1;
}
```

Assignment 15: Static Single Assignment Form

Transform the above code into Static Single Assignment Form (SSA).

```
i_0 = 0;
p_0 = i_0 < n;
while (phi(p_0, p_1)) {
    j_0 = 0;
    t_0 = j_0 < n;
    while (phi(t_0, t_1)) {
        if (i_0<j_0) {
            A_1[i_0,j_0] = A_0[i_0,j_0] + i_0;
        }
```

```
        else if (j_0==i_0) {
            A_2[i_0,j_0] = -A_0[i_0,j_0];
        }
        else {
            A_3[i_0,j_0] = A_0[i_0,j_0] + j_0;
        }
        j_1 = j_0 + 1;
        t_1 = j_1 < n;
    }
    i_1 = i_0 + 1;
    p_1 = i_1 < n;
}
```

Assignment 16: Machine-independent optimisation

Apply the loop unswitching optimisation to the (original) code above.

```
i = 0;
while (i<n) {
    j = i + 1;
    while (j<n) {
        A[i,j] = A[i,j] + i;
        j = j + 1;
    }
    j = i;
    while (j == i) {
        A[i,j] = -A[i,j];
        j = j + 1;
    }
    j = 0;
    while (j < i) {
        A[i,j] = A[i,j] + j;
        j = j + 1;
    }
    i = i + 1;
}
```

Assignment 17: Compilation Schemes

Devise a formal compilation scheme that systematically eliminates all occurences of while-loops in the body of a CiviC function definition and replaces them by semantically equivalent control code without while-loops.

$$\mathcal{C}\left[\begin{array}{l}\text{while } (\mathit{condition}) \ \{ \\ \mathit{Body} \\ \} \\ \mathit{Rest}\end{array}\right] \tag{1}$$

$$\Rightarrow \begin{array}{l}\text{if } (\mathit{condition}) \ \{ \\ \text{do } \{ \\ \mathcal{C}\,[\mathit{Body}] \\ \} \text{ while}(\mathit{condition}); \\ \}\end{array} \Bigg| \mathit{always} \tag{2}$$