# CS 3EA3 Tutorial Exercises 1.0

McMaster University
Department of Computing And Software
Curtis D'Alves

January $9^{th}$, 2017

## Exercise 1.1

- Install Frama-C `https://frama-c.com/install-aluminium-20160501.html`. Fair warning, this will be far easier on a *unix-like* system, those using windows system may want to consider installing a Ubuntu partition (even if you're not installing Frama-C you may want to consider doing this).
- Then read chapters 2-4 of ACSL-By-Example. Focus mainly on learning just syntax for now (by example!!! the best way to learn) `https://bts.frama-c.com/dokuwiki/lib/exe/fetch.php?media=mantis:frama-c:tutorial:acsl-by-example-neon.pdf`.

## Exercise 1.2

```
/*@
    requires x >= 0;
    requires y >= 0;

    assigns \nothing;

    ensures \result >= 0;
*/
int add_nats(int x, int y) {
    int z = 0;
    //@ assert 0 <= z == 0;
    z = x + y;
    //@ assert 0 <= z == x+y;
    return z;
}
```

- Save the above code in a file **nat_arith.c**, and load it into Frama-C with the command

  ```
  $$ frama-c-gui -wp nat_arith.c
  ```

  (**Note:** The flag **-wp** tells Frama-C to use the **Weakest-Precondition** plugin, the main feature we will be exploiting in this course.) Click on you're source code on the left and start clicking around, notice the green dots (green is good). Click on **WP Goals** on the bottom, your screen should like look Figure 1
- **Delete the assert annotations** from the body, does the post-condition still pass?

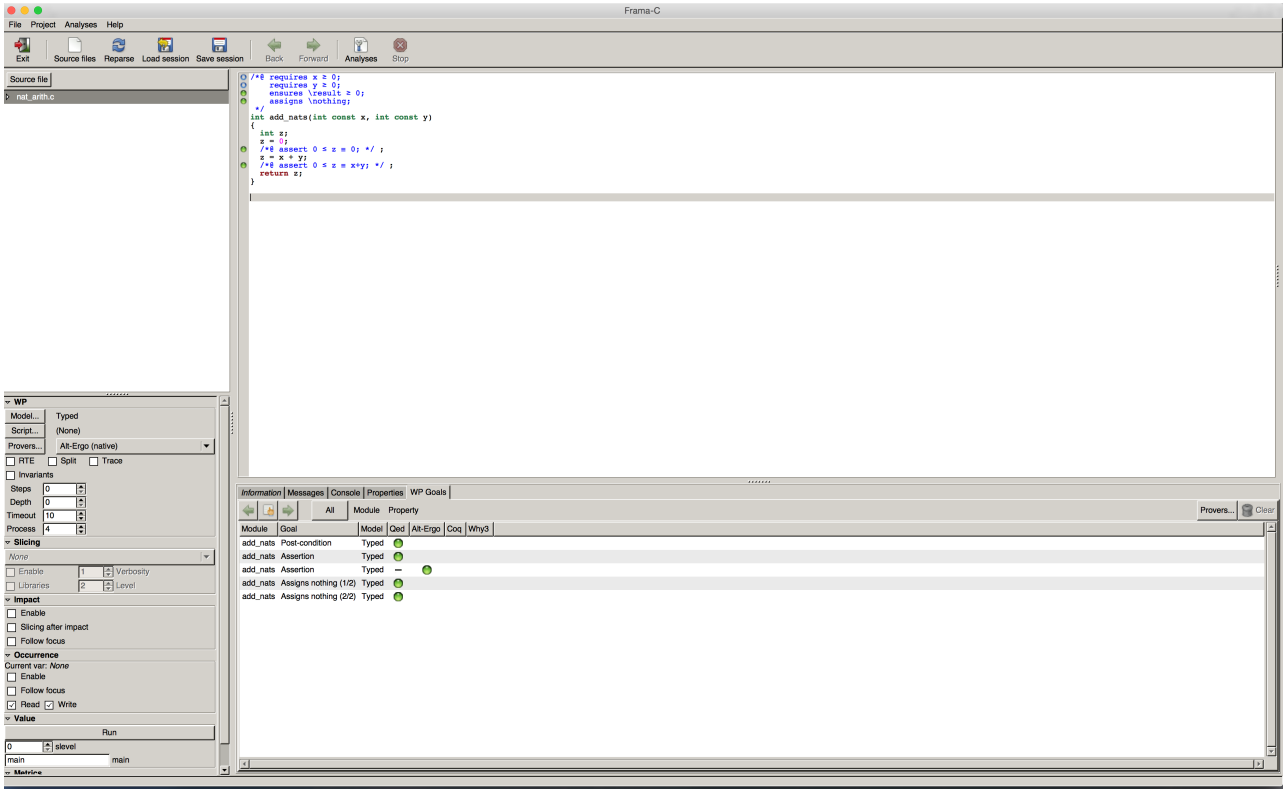


Figure 1: Frama-C Screenshot

# Exercise 1.3

```
/*@
    requires x >= 0;
    requires y >= 0;

    assigns \nothing;

    ensures \result >= 0;
*/
int sub_nats(int x, int y) {
    int z = 0;
    //@ assert 0 <= z == 0;
    z = x - y;
    //@ assert 0 <= z == x - y;
    return z;
}
```

- Add the above function to your file and reload Frama-C
- **Append the pre-condition** so that WP is able to verify the post-condition

## Exercise 1.4

```
/*@
    requires x >= 0;
    requires y >= 0;

    assigns \nothing;

    ensures \result == x*y;
*/
int mult_nats(int x, int y)
{
    int z = 0;
    /*@
        loop invariant 0 <= i <= y;
        loop assigns i,z;
        loop variant y-i;
    */
    for (int i = 0; i < y; i++) {
        z = z + x;
    }
    return z;
}
```

- Add the above function to your file and reload Frama-C
- **Append the loop-invariant** so that WP is able to verify the post-condition

## Exercise 1.4

```
/*@
    requires n >= 0;
    requires y >= 0;

    assigns \nothing;
*/
int is_sqrt(int y, int n)
{
    if (n == y*y) {
        return 1;
    }
    else {
        return 0;
    }
}
```

- Add the above function to your file and reload Frama-C
- **Add behavior annotations** to complete the specification