FramaC

Static analysis of C code

Why study frama C?

ACSL manual

-2^{+31} to $2^{31}-1$

\result

Contracts:

requires, ensures

```
/*@ requires x >= int_min;
    ensures \result >= 0;
    ensures x < 0 ==> \result == -x;
    ensures x >= 0 ==> \result == x;
*/
```

cumbersome

calling with (wp) plugin

frama-c-gui -wp -wp-rte filename.C

↓

wp runtime error → for runtime additional error checks

calling with (val) plugin

frama-c-gui -val filename.C

Some functions to write specification

\valid() \forall \exists

\old() \nothing \result

writing specification

```
/*@
*/ integer
   Real
```

Swap

```
/*@ requires \valid(a)
             &&
             \valid(b);
    ensures (*a == \old(*b)
             *b == \old(*a));
*/
```

```
void swap(int *a, int *b)
{
  int tmp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

frama-C operators

!P
P && Q
P || Q
P ==> Q → if P then Q
P <==> Q → P if and only if Q

Swap 2 elements in array

```
void swap (int n, int a[], int n1, int n2)
```

Specifications:

```
/*@ requires n >= 0 && 0 <= n1 < n && 0 <= n2 < n;
  requires \valid (a+(0..n-1));
  ensures \old(a[n1]) == \old(a[n2]) &&
          a[n2] == \old(a[n2]);
```

find()

```
int find (int n, const int a[], int v)
```

Specification:

```
/*@ requires n >= 0 && \valid(a+[0..n-1]);
  assigns \nothing;   ✱ → note this.
  ensures \result == -1
  ==> (\forall i; 0 <= i < n ==> a[i] != v)

  ensures 0 <= \result < n
  ==> a[\result] == v;

  ensures -1 <= \result < n;

*/
```

Loops

loop invariant, loop assigns, loop variant

```
/*@ loop invariant    0 <= i < n;
    loop invariant   \forall integer j;
                       0 <= j < i ==> a[j] != v;
    loop assigns     i;
    loop variant     n - i;
*/
```

loop invariant → should be true @ start of loop
@ each iteration flow
@ end of loop

In (WP) more specifications than code, it checks for all possible values, (exhaustive test)

Behaviors: Used to write clean contracts (assumes)*

↳ give name for each behavior.
↳ write assumes, requires, ensures for each behavior

↳ complete behaviors & disjoint behaviors
↳ all possible input ↳ different cases

example with find

```
/*@ requires   n >= 0 && \valid(a+(0..n-1));
    assigns    \nothing;

    behavior found:
    assumes    \exist integer i;   0 <= i < n && a[i] == v;
    ensures    a[\result] == v;

    behavior notfound:
    assumes    \forall integer i;  0 <= i < n && a[i] != v;
    ensures    \result == -1;
```