

Cours MALG & MOVEX

Vérification d'une annotation

Dominique Méry

Telecom Nancy, Université de Lorraine

(23 mars 2025 at 10:08 P.M.)

Année universitaire 2024-2025

- ① WP calculus in Frama-c
- ② First annotation
- ③ Second annotation

① WP calculus in Frama-c

② First annotation

③ Second annotation

Listing 1 – an1.c

```
//@ assert I1:  $P(x)$ ;  
  x = e(x);  
//@ assert I2:  $Q(x)$ ;
```

Listing 2 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

► $P(x) \Rightarrow WP(x := e(x))(Q(x))$

Listing 3 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$

Listing 4 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$
- ▶ $P(x1) \Rightarrow Q[x \mapsto e(x1)]$ (renaming of free occurrences of x by $x1$)

Listing 5 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$
- ▶ $P(x1) \Rightarrow Q[x \mapsto e(x1)]$ (renaming of free occurrences of x by $x1$)
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$

Listing 6 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$
- ▶ $P(x1) \Rightarrow Q[x \mapsto e(x1)]$ (renaming of free occurrences of x by $x1$)
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$

Listing 7 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$
- ▶ $P(x1) \Rightarrow Q[x \mapsto e(x1)]$ (renaming of free occurrences of x by $x1$)
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $\vdash P(x1) \wedge x = e(x1) \Rightarrow Q(x)$

Listing 8 – an1.c

```
//@ assert l1: P(x);  
  x = e(x);  
//@ assert l2: Q(x);
```

- ▶ $P(x) \Rightarrow WP(x := e(x))(Q(x))$
- ▶ $P(x) \Rightarrow Q[x \mapsto e(x)]$
- ▶ $P(x1) \Rightarrow Q[x \mapsto e(x1)]$ (renaming of free occurrences of x by $x1$)
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $\vdash P(x1) \wedge x = e(x1) \Rightarrow Q(x)$
- ▶ $P(x1) \wedge x = e(x1) \vdash Q(x)$

Listing 10 – an1.c

```
void ex(void) {  
    int x=12,y=24;  
    //@ assert l1:  $2*x == y$ ;  
    x = x+1;  
    //@ assert l2:  $y == 2*(x-1)$ ;  
}
```

```
[kernel] Parsing an1.c (with preprocessing)
[wp] Running WP plugin...
[wp] Warning: Missing RTE guards
[wp] 2 goals scheduled
[wp] Proved goals:      4 / 4
    Terminating:      1
    Unreachable:        1
    Qed:                 2
```

Goal Assertion 'l1' (file an1.c, line 3):

```
Assume {  
  Type: is_sint32(x) /\ is_sint32(y).  
  (* Initializer *)  
  Init: x = 12.  
  (* Initializer *)  
  Init: y = 24.  
}
```

Prove: $(2 * x) = y$.

Prover Qed returns Valid

Goal Assertion 'l2' (file an1.c, line 5):

```
Assume {  
  Type: is_sint32(x) /\ is_sint32(x_1) /\ is_sint32(y).  
  (* Initializer *)  
  Init: x_1 = 12.  
  (* Initializer *)  
  Init: y = 24.  
  (* Assertion 'l1' *)  
  Have: (2 * x_1) = y.  
  Have: (1 + x_1) = x.  
}  
Prove: (2 + y) = (2 * x).  
Prover Qed returns Valid
```



```
[kernel] Parsing an1.c (with preprocessing)
[wp] Running WP plugin...
[wp] Warning: Missing RTE guards
[wp] 2 goals scheduled
[wp] Proved goals:      4 / 4
    Terminating:      1
    Unreachable:        1
    Qed:                 2
```

Listing 11 – an2.c

```
void ex(void) {  
    int x=12,y=24;  
    //@ assert l1: 2*x == y;  
    x = x+1;  
    //@ assert l2:  y == 2*(x-1);  
    x = x+2;  
    //@ assert l3:  y+6 == 2*x;  
}
```

```
[kernel] Parsing an2.c (with preprocessing)
[wp] Running WP plugin...
[wp] Warning: Missing RTE guards
[wp] 3 goals scheduled
[wp] Proved goals:      5 / 5
    Terminating:      1
    Unreachable:       1
    Qed:                3
```

Goal Assertion 'l1' (file an2.c, line 3):

```
Assume {  
  Type: is_sint32(x) /\ is_sint32(y).  
  (* Initializer *)  
  Init: x = 12.  
  (* Initializer *)  
  Init: y = 24.  
}
```

Prove: $(2 * x) = y$.

Prover Qed returns Valid

Goal Assertion 'l2' (file an2.c, line 5):

```
Assume {  
  Type: is_sint32(x) /\ is_sint32(x_1) /\ is_sint32(y).  
  (* Initializer *)  
  Init: x_1 = 12.  
  (* Initializer *)  
  Init: y = 24.  
  (* Assertion 'l1' *)  
  Have: (2 * x_1) = y.  
  Have: (1 + x_1) = x.  
}  
Prove: (2 + y) = (2 * x).  
Prover Qed returns Valid
```

Annotation simple (ii)

Goal Assertion 'l3' (file an2.c, line 7):

Assume {

Type: $\text{is_sint32}(x) \wedge \text{is_sint32}(x_1) \wedge \text{is_sint32}(x_2) \wedge \text{is_sint32}(x_3)$
(* Initializer *)

Init: $x_2 = 12$.

(* Initializer *)

Init: $y = 24$.

(* Assertion 'l1' *)

Have: $(2 * x_2) = y$.

Have: $(1 + x_2) = x_1$.

(* Assertion 'l2' *)

Have: $(2 + y) = (2 * x_1)$.

Have: $(2 + x_1) = x$.

}

Prove: $(6 + y) = (2 * x)$.

Prover Qed returns Valid

Listing 12 – an2bis.c

```
void ex(void) {  
    int x=12,y=24;  
    //@ assert l1:  $2*x == y$ ;  
    x = x+1;  
    //@ assert l2:  $y == 2*(x-1)$ ;  
    x = x+2;  
    //@ assert l3:  $y+6 == 2*x$ ;  
}
```

