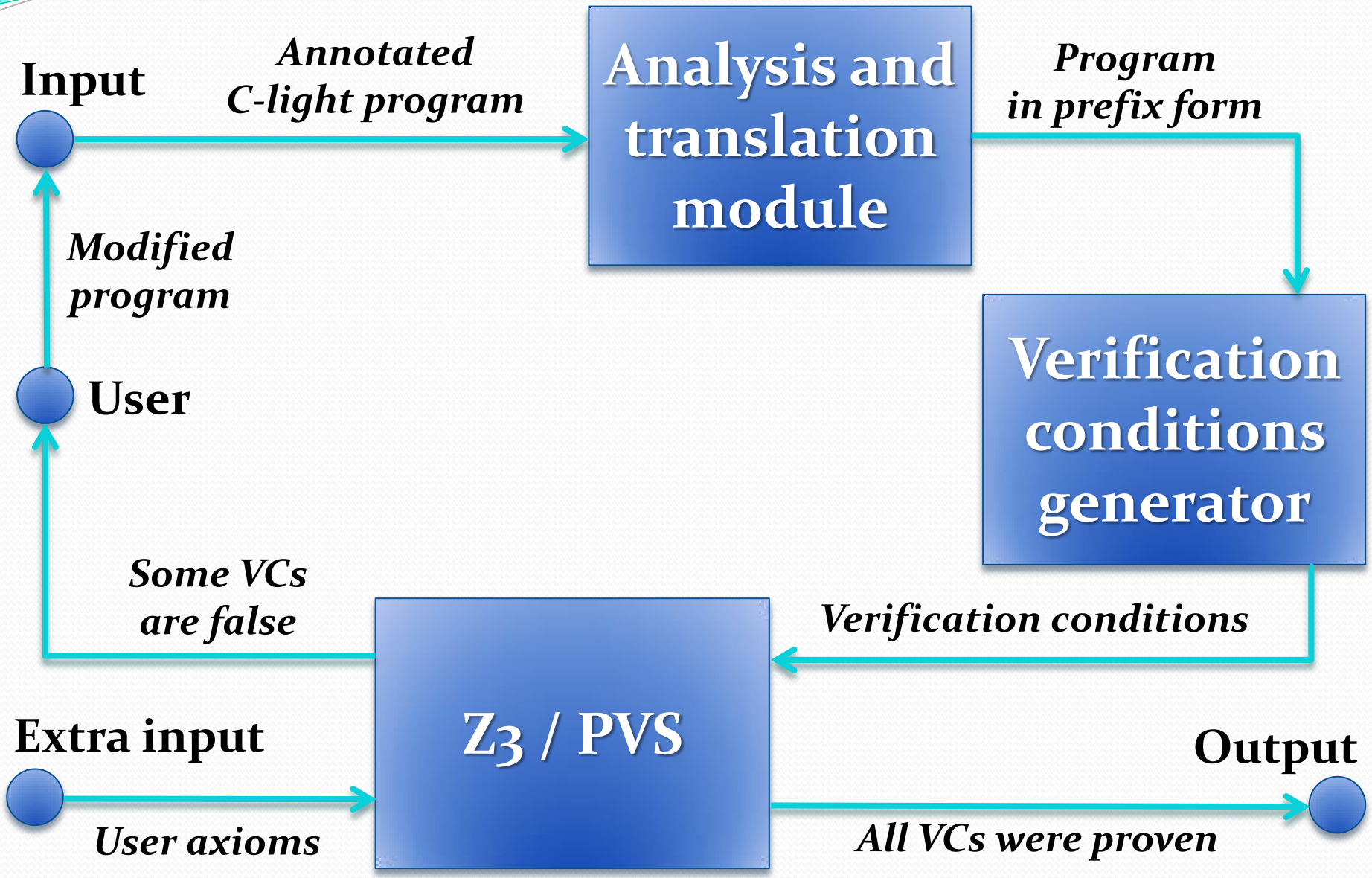
Ilya Maryasov, Valery Nepomniaschy,
and Dmitry Kondratyev

Verification of Definite Iteration over Arrays with a Loop Exit in C Programs

A. P. Ershov Institute of Informatics Systems,
Siberian Branch of the Russian Academy of Sciences,
Novosibirsk, Russia

Program Semantics, Specification and Verification:
Theory and Applications
Moscow, 2017

Design of the C-light Verification System



Definite Iteration

1. Definite iteration over unchangeable data structures without loop exits.
2. Definite iteration over unchangeable data structures with a loop exit.
3. Definite iteration over changeable data structures with a loop exit.
4. Definite iteration over hierarchical data structures with a loop exit.

Definite Iteration over Unchangeable Data Structures

Let S be the structure:

memb(S) is the multiset of elements of the structure S;

|memb(S)| be the power of the multiset memb(S).

1. **empty(S) = true iff |memb(S)| = 0.**
2. **choo(S) returns an element of memb(S) iff ¬empty(S).**
3. **rest(S) = S', where S' is a structure of the type of S and**
   memb(S') = memb(S) \ {choo(S)}
   iff ¬empty(S).

Definite Iteration over Unchangeable Data Structures

Let ¬empty(S):

vec(S) = $[s_1, s_2, \ldots, s_n]$

where memb(S) = $\{s_1, s_2, \ldots, s_n\}$,

s_i = choo($rest^{i-1}$(S))

for i = 1, 2, ..., n.

Definite Iteration over Unchangeable Data Structures

for x in S do v := body(v; x) end

S is a structure,
x is the variable of the type
"an element of S",
v is a vector of loop variables and
$x \notin v$,
body does not modify x and S.

The Replacement Operation

$rep(v, S, body) = v_n$,

where $v_0 = v$ if $empty(S)$,

$v_i = body(v_{i-1}, s_i)$

for all $i = 1, 2, \ldots, n$

if $\neg empty(S)$.

Inference Rule

$$\frac{\{\exists v'\ P(v \leftarrow v') \wedge v = \mathrm{rep}(v', S, \mathrm{body})\}\ A;\ \{Q\}}{\{P\}\ \textbf{for}\ x\ \textbf{in}\ S\ \textbf{do}\ v := \mathrm{body}(v, x)\ \textbf{end}\ A;\ \{Q\}}$$

Definite Iteration over Unchangeable Arrays

```
for (i = 0; i < n; i++) v := body(v, i, S) end
```

S is a one-dimensional array,
i, n are the variables of the integer type,
v is a vector of loop variables and
$i \notin v$,
body does not modify i and S, can contain assignment, if and break statements.

Algorithm for rep Function Determination

Let the body has the form:

$$\{x_1 = expr_1(x_1, x_2, \ldots, x_k);$$
$$x_2 = expr_2(x_1, x_2, \ldots, x_k);$$
$$\ldots$$
$$x_k = expr_k(x_1, x_2, \ldots, x_k);\}$$

$$v = (x_1, x_2, \ldots, x_k)$$
$$rep(v, S, body, 0) = (x_{1_0}, x_{2_0}, \ldots, x_{k_0})$$

Algorithm for rep Function Determination

$x_1 = expr_1(x_1, x_2, \ldots, x_k);$

$x_2 = expr_2(expr_1(x_1, x_2, \ldots, x_k), x_2, \ldots, x_k);$

...

$x_j \leftarrow rep((x_1, x_2, \ldots, x_k), S, body, i - 1)$

Algorithm for rep Function Determination

if ($e(i, x_1, x_2, \ldots, x_k)$) {A;} else {B;}

A, B are compound statements

$$\forall x_1 \forall x_2 \ldots \forall x_k \; e(i, x_1, x_2, \ldots, x_k) \Rightarrow A^*$$
$$\forall x_1 \forall x_2 \ldots \forall x_k \; \neg e(i, x_1, x_2, \ldots, x_k) \Rightarrow B^*$$

Algorithm for rep Function Determination

break statement:

1. Top level — rep is defined for i = 0, 1

2. In the if statement:

$$\exists j\ 0 < j \leq n\ \forall i\ j < i \leq n$$

$$rep((x_1, x_2, \ldots, x_k), S, body, i) =$$
$$rep((x_1, x_2, \ldots, x_k), S, body, j)$$

$$\forall x_1 \forall x_2 \ldots \forall x_k\ e(i, x_1, x_2, \ldots, x_k) \Rightarrow$$
$$(A^* \wedge (\forall l\ i < l \Rightarrow A^*))$$

Example

```
/* (length > 0) */
int search(int* arr, int length)
{
    auto int result = -1;
    for (i = 0; i < length; i++)
     if (key == arr[i])
     {
      result = i;
      break;
     };
    return result;
/* Q */
```

Example: Postcondition

$$
\begin{aligned}
Q \equiv & (\forall i\ (0 \le i) \wedge (i < length) \Rightarrow \\
& \qquad (key \neq arr[i]) \wedge (result = -1)) \\
\vee & \\
& (\exists r\ (0 \le r) \wedge (r < length) \Rightarrow \\
& \qquad (\forall i\ (-1 < i) \wedge (i < r) \Rightarrow \\
& \qquad\qquad (key \neq arr[i]) \wedge \\
& \qquad\qquad (key = arr[r]) \wedge \\
& \qquad\qquad (result = r)))
\end{aligned}
$$

Example: Verification Condition

$$
\begin{aligned}
&(length > 0) \wedge (result = rep(result, key, arr, length)) \Rightarrow \\
&\quad (\forall i\ (0 \leq i) \wedge (i < length) \Rightarrow \\
&\qquad\qquad (key \neq arr[i]) \wedge (result = -1)) \\
&\quad \vee \\
&\quad (\exists r\ (0 \leq r) \wedge (r < length) \Rightarrow \\
&\qquad\qquad (\forall i\ (-1 < i) \wedge (i < r) \Rightarrow \\
&\qquad\qquad\qquad (key \neq arr[i]) \wedge \\
&\qquad\qquad\qquad (key = arr[r]) \wedge \\
&\qquad\qquad\qquad (result = r)))
\end{aligned}
$$

Example: Verification Condition

```
(assert (not (forall ((result Int) (key Int)
 (length Int) (arr (Array Int Int)))
 (implies (and (> length 0)
               (= result (rep result key arr length)))
          (or (forall ((i Int))
               (implies (and (< -1 i) (< i length))
                        (and (not (= key (select arr i)))
                        (= result -1))))
              (exists ((r Int))
               (implies (and (< -1 r) (< r length))
                   (forall ((i Int))
                    (implies (and (< -1 i) (< i r))
                    (and (not (= key (select arr i)))
                              (= key (select arr r))
                              (= result r)))))))))))
```

Example: Axioms for *rep* Function

$$(i < 1) \Rightarrow (rep(result, key, arr, i) = -1)$$

$$(0 < i) \wedge (arr[i - 1] = key) \Rightarrow$$
$$(rep(result, key, arr, i) = i - 1) \wedge$$
$$(\forall j\ (i < j) \Rightarrow$$
$$rep(result, key, arr, j) = i - 1)$$

$$(0 < i) \wedge (arr[i - 1] \neq key) \Rightarrow$$
$$(rep(result, key, arr, i) = rep(result, key, arr, i - 1))$$

Working with Induction in Z3

$$\forall n\ P(n)$$

is rewriten to

$$\forall n\ (\forall k\ k < n \Rightarrow P(k))$$

Induction base: $P(1)$

Induction step: $P(i) \Rightarrow P(i + 1)$

Example: Extra Axiom

$$
\begin{aligned}
&(len > 0) \wedge (length > len) \wedge (result = rep(result, key, arr, len)) \Rightarrow \\
&\quad (\forall i\ (0 \leq i) \wedge (i < len) \Rightarrow \\
&\qquad\qquad (key \neq arr[i]) \wedge (result = -1)) \\
&\quad \vee \\
&\quad (\exists r\ (0 \leq r) \wedge (r < len) \Rightarrow \\
&\qquad\qquad (\forall i\ (0 \leq i) \wedge (i < r) \Rightarrow \\
&\qquad\qquad\qquad (key \neq arr[i]) \wedge \\
&\qquad\qquad\qquad (key = arr[r]) \wedge \\
&\qquad\qquad\qquad (result = r)))
\end{aligned}
$$

Example: Extra Verification Condition

$$
\begin{aligned}
&(length = 1) \wedge (result = rep(result, key, arr, length)) \Rightarrow \\
&\quad (\forall i\ (0 \leq i) \wedge (i < length) \Rightarrow \\
&\qquad\qquad (key \neq arr[i]) \wedge (result = -1)) \\
&\quad \vee \\
&\quad (\exists r\ (0 \leq r) \wedge (r < length) \Rightarrow \\
&\qquad\qquad (\forall i\ (-1 < i) \wedge (i < r) \Rightarrow \\
&\qquad\qquad\qquad (key \neq arr[i]) \wedge \\
&\qquad\qquad\qquad (key = arr[r]) \wedge \\
&\qquad\qquad\qquad (result = r)))
\end{aligned}
$$

Using PVS to Prove Inductive Verification Conditions

```
search: THEORY
BEGIN  rep(result:int, key:int, arr:ARRAY[nat->int],
i:nat) : RECURSIVE int =
     IF (i < 1) THEN result
     ELSE (IF ((0 < i) AND (arr(i-1) = key)) THEN
rep(i-1, key, arr, i-1)
           ELSE rep(result, key, arr, i-1) ENDIF)
ENDIF
  MEASURE i
```

Using PVS to Prove Inductive Verification Conditions

```
vc: LEMMA
      FORALL (result:int, key:int, length:nat,
arr:ARRAY[nat -> int]):
               ((length > 0) AND (result = rep(-1, key,
arr, length)))
      IMPLIES
      ((FORALL (i:int): ((0 <= i) AND (i < length) AND
(not (key = arr(i))))
       IMPLIES
       (result = -1))
      OR
      (EXISTS (r:int): ((0 <= r) AND (r < length))
       IMPLIES
       (FORALL (i:int): ((0 <= i) AND (i < r) AND (not
(key = arr(i)))
                              AND (key = arr(r))))
        IMPLIES (result = r)))
END search
```

Conclusion

1. The work is in progress.
2. Automatic verification of definite iteration over unchangeable arrays with a loop exit.
3. Plans: definite iteration over changeable structures, more complicated examples.

THANK YOU!