Knowledge Transfer based Many-Objective Approach for Finding Bugs in Multi-path Loops

Han Huang, Stuart D. Semujju, Fangqing Liu, Yi Xiang, Shuling Yang, and Zhifeng Hao

1 MODIFIED PROGRAMS

This section presents the modified versions of the case study subjects in Section 6.2 of the main article. The presented modified programs are written in C language.

1.1 loop-acceleration

Fig. 1 depicts the modified program diamond_1-1.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-acceleration/diamond_1-1.c.

Fig. 2 depicts the modified program diamond_2-2.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-acceleration/diamond_2-2.c.

```
1 void diamond1 (int y ){
2 int x = 0;
3 while (x < 10) {
4 if (y % 2 == 0) {
5   x++;
6  }
7 else {
8       x += 2;
9     }
10}
11 if(((x % 2 ) != (y % 2)));//bug 1
12}
```

Fig. 1: diamond_1-1.c

1.2 loops

Fig. 3 depicts the modified program insertion_sort-1. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/insertion_sort-1.c.

Fig. 4 depicts the modified program string-1.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/string-1.c.

- *H. Huang, S. D. Semujju, F. Liu, Yi. Xiang and S. Yang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, P.R.China.*
  E-mail: hhan@scut.edu.cn, stuartsemujju@gmail.com, 564376030@qq.com, gzhuxiang_yi@163.com, and 694712017@qq.com.
- *Z. Hao is with the Department of Mathematics, College of Science Shantou University, Guangdong 515063, P.R.China. E-mail: zfhao@fosu.edu.cn*


```
1 void diamond2 (int y ){
2 int x = 0;
3 while (x < 3) {
4 if (y % 2 == 0) x += 2;
5 else x++;
6 if (y % 2 == 0) x += 2;
7 else x -= 2;
8 if (y % 2 == 0) x += 2;
9 else x += 2;
10 if (y % 2 == 0) x += 2;
11 else x -= 2;
12 if (y % 2 == 0) x += 2;
13 else x += 2;
14 if (y % 2 == 0) x += 2;
15 else x -= 4;
16 if (y % 2 == 0) x += 2;
17 else x += 4;
18 if (y % 2 == 0) x += 2;
19 else x += 2;
20 if (y % 2 == 0) x += 2;
21 else x -= 4;
22 if (y % 2 == 0) x += 2;
22 else x -= 4;
23 }
24 if((x % 2) != (y % 2)); // bug 1
25 }
```

Fig. 2: diamond_2.2.c

```
1 void insertion1 (int v [] ){
2 int i = 0;
3 int j = 0;
4 int k = 0;
5 int key = 0;
6 int SIZE = 10;
7 for (j = 1; j< SIZE ;j++) {
8    key = v[j];
9    i = j - 1;
10    while((i>=0) && (v[i]>key)) {
11    if (i<2){
12    v[i+1] = v[i];}
13    i = i - 1;
14    }
15   v[i+1] = key;
16 }
17 for (k=1;k<SIZE;k++){
18    if((v[k-1] > v[k]));// bug 1
19 }
20 }
```

Fig. 3: insertion_sort-1

```
1 void string_1(char [] string_A, char [] string_B){
2 int i = 0;
3 int j = 0;
4 int nc_A = 0;
5 int nc_B  = 0;
6 int found=0;
7 nc_A = 0;
8 while(string_A[nc_A]!='\0'){
9         nc_A++;
10 }
11 nc_B = 0;
12 while(string_B[nc_B]!='\0'){
13    nc_B++;
14 }
15 i=j=0;
16 while((i<nc_A) && (j<nc_B)){
17 if(string_A[i] == string_B[j]){
18             i++;
19             j++;
20 }
21 else{
22    i = i-j+1;
23    j = 0;
24    }
25}
26if (j>nc_B-1){
27    found = -1;
28}
29 else {found = 0;}
30 if((found != 0));//bug 1
31}
```

Fig.4: string-1.c

Fig. 5 depicts the modified program veris.c_OpenSER_cases1_stripFullBoth_arr.c. The original program can be found at: https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/veris.c_OpenSER_cases1_stripFullBoth_arr.c.

Fig. 6 depicts the modified program vogal-1.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/vogal-1.c.

Fig. 7 depicts the modified program vogal-2.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/vogal-2.c.

Fig. 8 depicts the modified program invert_string-1.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/invert_string-1.c.

Fig. 9 depicts the modified program sum_array-1. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loops/sum_array-1.c.

1.3 loop-zilu

Fig. 10 depicts the modified program benchmark05_conjuctive.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark05_conjunctive.c.

Fig. 11 depicts the modified program benchmark06_conjuctive.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark06_conjunctive.c.

Fig. 12 depicts the modified program benchmark40_polynomial.c. The original program can be found at:

```
1 #define BASE_SZ  2;
2 #define  EOS  0;
3 #define  EOF -1;
4 #define  ERR  -1;
5 #define  NEEDLE_SZ  2;
6 #define  EXPRESSION_LENGTH   BASE_SZ;
7 #define  LINE_LENGTH EXPRESSION_LENGTH + NEEDLE_SZ + 4;
8 void verisec1(char [] str ){
9 int start=0;
10 int i=-1;
11 int j=-1;
12 do {
13    i++;
14 switch( str[i]) {
15    case EOS:
16       while ((str[start] == ' ') || (str[start] == '\t')){
17              start++;
18       }
19      if (str[start] == '"') {
20            start++;
21       }
22        j = i-1;
23       while ((0 < j) && ((str[j] == ' ') || (str[j] == '\t')))
     {
24          j--;
25      }
26      if ((0 < j) && (str[j] == '"')) {
27           j--;
28      }
29      if (start<=j) {
30          if(j - start + 1 < 2); // bug 1
31        }
32     start = i+1;
33  }
34 } while (str[i] != EOS);
35}
```

Fig. 5: veris.c_OpenSER_cases1_stripFullBoth_arr.c

```
1 void Vogal1(char [] input_string ){
2 int MAX = 5;
3 char[] vogal_array = {'a','A','e','E','i','I','o','O','u','U','
    \0'};
4 int  i = 0;
5 int j = 0;
6 int cont = 0;
7 int n_caracter = 0;
8 while(input_string[n_caracter]!='\0'){
9         n_caracter++;
10 }
11 for(i=0;i<n_caracter;i++){
12    for(j=0;j<MAX/2;j++)
13      if(input_string[i] == vogal_array[j])
14                cont++;
15 }
16  i = 0;
17 int cont_aux = 0;
18 while(input_string[i]!='\0'){
19   for(j=0;j<MAX/2;j++){
20        if(input_string[i] == vogal_array[j])
21                cont_aux++;
22    }
23  i++;
24 }
25if((cont_aux!=cont)); //bug 1
26}
```

Fig. 6: vogal-1.c

```
1 void Vogal2(char [] input_string ){
2 int MAX = 10;
3 input_string[MAX-1] = '\0';
4 char [] vetor_vogais={'a','A','e','E','i','I','o','O','u','U','
    \0'};
5 int  i = 0;
6 int j = 0;
7 int cont = 0;
8 int n_caracter = 0;
9 while(input_string[n_caracter]!='\0'){
10          n_caracter++;
11 }
12 cont = 0;
13 for(i=0;i<n_caracter;i++){
14           for(j=0;j<8;j++){
15              if(input_string[i] == vetor_vogais[j])
16                 cont++;
17    }
18 }
19 i=0;
20 int cont_aux = 0;
21 while(input_string[i]!='\0'){
22    for(j=0;j<10;j++){
23      if(input_string[i] == vetor_vogais[j])
24                 cont_aux++;
25    }
26          i++;
27}
28 if((cont_aux!=cont));//bug 1
29}
```

Fig. 7: vogal-2

```
1 #define MAX 10
2 void invert(char [] str1, char [] str2) {
3 int cont = 0;
4 int i = 0;
5 int j = 0;
6 str1[MAX-1]= '\0';
7 j = 0;
8 for (i = MAX - 1; i >= 0; i--) {
9     str2[j] = str1[0];
10      j++;
11 }
12 j = MAX-1;
13 for (i=0; i<MAX; i++) {
14 if (str1[i] != str2[j]);//bug 1
15j--;
16 }
17}
```

Fig. 8: invert_string-1

```
1 void sumarray(int [] A, int [] B , int [] C) {
2 for(int i =0;i< M;i++) {
3    C[i]=A[i]+B[i];
4 }
5 for(int i=0;i<M;i++) {
6 if(C[i]!=A[i]-B[i]) // bug 1
7 }
8}
```

Fig. 9: sum_array-1

```
1 void benchmark05(int x, int y) {
2 int SIZE = 10;
3 if (!(x>=0 && x<=y && y<SIZE)) {
3 while (x< SIZE) {
4    x++;
5 if (x>y) {
6    y++;
7}
8}
9 if(y!=SIZE); // bug 1
10 }}
```

Fig. 10: benchmark05_conjunctive.c

```
1 void benchmark06 (int j, int i , int x, int y, int k){
2 int w = 0;
3 while ( w < SIZE ) {
4 if(j==i) {
5 x++;
6 y--;
7 }
8 else {
9    y++;
10    x--;
11 }
12 j++;
13 w++;
14 }
15if(x+y!=k); //bug 1
16}
```

Fig. 11: benchmark06_conjuctive.c

```
1 void benchmark40 (int x, int y){
2 int i = 0;
3 int SIZE = 5;
4 while (i < SIZE) {
5 if (x == 0) {
6 if (y > 0) {
7  x++;
8 }
9 else {
10   x--;
11 }
12 }
13 if (x > 0) {
14   y++;
15 }
16 else {
17  x--;
18 }
19  i++;
20}
21if ((x * y) < 0); // bug 1
22}
```

Fig. 12: benchmark40_polynomial.c

https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark40_polynomial.c.

Fig. 13 depicts the modified program benchmark44_disjunctive.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark44_disjunctive.c.

Fig. 14 depicts the modified program benchmark47_linear.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark47_linear.c.

Fig. 15 depicts the modified program benchmark53_polynomial.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/loop-zilu/benchmark53_polynomial.c.

1.4 nla-digbench-scaling

Fig. 16 depicts the modified program bresenham-ll_unwindbound10.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/bresenham-ll_unwindbound10.c.

Fig. 17 depicts the modified program dijkstra-u_unwindbound10.c The original program can be found at: https://gitlab.com/sosy-lab/benchmarking/sv-

```
1 void benchmark44 (int x, int y){
2 int counter = 0;
3 while ( x < y ){
4 if(counter == 5){
5   break;
6 }
7 if ((x<0 && y<0)) {
8   x=x+7; y=y-10;
9 }
10 if ((x<0 && y>=0)) {
11    x=x+7; y=y+3;
12 }
13 else {
14  x=x+10; y=y+3;
15 }
16  counter++;
17 }
18 if(!(x >= y && x <= y + 16)); //bug 1
19}
```

Fig. 13: benchmark44_disjunctive.c

```
1 void benchmark47 (int x, int y){
2 int counter = 0;
3 while (x < y){
4 if (counter == 5){
5    break;
6 }
7 if (x < 0) {
8    x = x + 7;
9 }
10 else {
11  x = x + 10;
12 }
13 if (y < 0){
14  y = y - 10;
15 }
16 else {
17 y = y + 3;}
18  counter++;
19}
20 if(!(x >= y && x <= y + 16));// bug 1
21}
```

Fig. 14: benchmark47_linear.c

```
1 void benchmark53 (int x , int y){
2 int i = 0;
3 while (i < 5) {
4  if(x == 0) {
5 if (y>0){
6        x++;
7    }
10  else {
11     x--;
12  }
13 }
14 if(x > 0) {
15    y++;
16 }
17 else {
18  x--;
19 }
20  i++;
21}
22if ((x*y) < 0); // bug found
23}
```

Fig. 15: benchmark53_polynomial.c

```
1 void bresenham(int X, int Y ){
2 long x = 0;
3 long y = 0;
4 long v = 0;
5 long xy = 0;
6 long yx = 0;
7 v = ((long)2 * Y) - X; // cast required to avoid int overflow
8 y = 0;
9 x = 0;
10 int counter = 0;
11 while(counter < 10) {
12  yx = (long)Y * x;
13  xy = (long)X * y;
14  if( 2*yx - 2*xy - X + (long long) 2*Y - v != 0) //bug
15  if(!(x <= X)) break;
16  if(v < 0) {
17    v = v + (long)2 * Y;
18  }
19   else {
20    v = v + 2 * ((long)Y - X);
21    y++;
22  }
23  x++;
24  counter++;
25}
26 xy = x * y;
27  yx = (long)Y * x;
28  if( 2 * yx - 2 * xy - X + (long)2 * Y - v + 2 * y != 0 ); //
     bug
29}
```

Fig. 16: bresenham-ll_unwindbound10.c

benchmarks/-/blob/main/c/nla-digbench-scaling/dijkstra-u_unwindbound10.c.

Fig. 18 depicts the modified program cohendiv-ll_unwindbound2.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/cohendiv-ll_unwindbound2.c.

Fig. 19 depicts the modified program divbin_unwindbound5.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/divbin_unwindbound5.c.

Fig. 20 depicts the modified program egcd-ll_unwindbound2.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/egcd-ll_unwindbound2.c.

Fig. 21 depicts the modified program hard2_unwindbound10.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/hard-ll_unwindbound10.c.

Fig. 22 depicts the modified program mannadiv_unwindbound10.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/mannadiv_unwindbound10.c.

Fig. 23 depicts the modified program prod4br-ll_unwindbound5.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/blob/main/c/nla-digbench-scaling/prod4br-ll_unwindbound5.c.

Fig. 24 depicts the modified program prodbin-ll_unwindbound10.c. The original program can be found at:https://gitlab.com/sosy-lab/benchmarking/sv-

```
1 void dijkastra(int n) {
2 if(!(n < 4294967295 / 4)) {abort();}
3 int p = 0;
4 int q = 1;
5 int r = n;
6 int h = 0;
7 int count = 0;
8 while (count++ < 10) {
9 if (!(q <= n))break;
10     q = 4 * q;
11     count++;
12}
13 int i = 0;
14 while (i++ < 10) {
15 if(r >= 2 * p + q);//bug 1
16 if(p*p + r*q != n*q); //bug 2
17 if(h * h * h - 12 * h * n * q + 16 * n * p * q - h * q * q - 4
      * p * q * q + 12 * h * q * r - 16 * p * q * r != 0);//bug
     3
18 if(h * h * n - 4 * h * n * p + 4 * (n * n) * q - n * q * q - h
      * h * r + 4 * h * p * r - 8 * n * q * r + q * q * r + 4 * q
      * r * r 1= 0);//bug
     4
19 if(h * h * p - 4 * h * n * q + 4 * n * p * q - p * q * q + 4 *
      h * q * r - 4 * p * q * r != 0); //bug
     5
20 if(p * p - n * q + q * r != 0); //bug 6
21 if (!(q != 1))break;
22   q = q / 4;
23   h = p + q;
24   p = p / 2;
25  if (r >= h){
26  p = p + q;
27  r = r - h;
28  }
29 i++;
30}
31 if(h*h*h - 12*h*n + 16*n*p + 12*h*r - 16*p*r - h - 4*p != 0);
     //bug 7
32 if(p*p - n + r != 0);//bug 8
33 if(h*h*p - 4*h*n + 4*n*p + 4*h*r - 4*p*r - p != 0);//bug 9
34 }
```

Fig. 17: dijkstra-u_unwindbound10.c

```
1 void cohendiv(int x, int y) {
2 int q = 0;
3 int r = 0;
4 int a = 0;
5 int b = 0;
6 if(!(y >= 1)) { abort();}
7 q = 0;
8 r = x;
9 a = 0;
10  b = 0;
11  int counter = 0;
12  while (counter < 2) {
13    if (!(r >= y)) break;
14    a = 1;
15    b = y;
16    int i = 0;
17    while (i<2) {
18        if(b != y*a);  //bug 1
19          if (x != q*y + r); // bug 2
20        if (r < 0); //bug 3
21        if (!(r < 2 * b))break;
22        if(r < 2 * y * a);//bug 4
23        a = 2 * a;
24        b = 2 * b;
25        i++;
26    }
27    r = r - b;
28    q = q + a;
29    counter++;
30      }
31      if(x != q*y + r);//bug 5
32  }
```

Fig. 18: cohendiv-ll_unwindbound2.c

```
1 void Divbin(int A , int B) {
2 if (!( B < 2147483647) &&( B >=1  )){abort();}
3 int q = 0;
4 int r = 0;
5 int a = 0;
6 int b = 0;
7 r = A;
8 b = B;
9 int i = 0;
10 while (i++<5) {
11     if (!(r >= b)) break;
12      b = 2 * b;
13 }
14 int counter = 0;
15 while (counter++<5) {
16    if(A != q * b + r);//bug1
17    if (!(b != B)) break;
18    q = 2 * q;
19    b = b / 2;
20    if (r >= b) {
21       q = q + 1;
22       r = r - b;
23    }
24 }
25  if(A != q * b + r);//bug2
26 }
```

Fig. 19: divbin_unwindbound5.c

```
#include<stdio.h>
#include<stdlib.h>
1 void egcd(int x, int y) {
2 int q = 0;
3 int b = 0;
4 int p = 0;
7 int r = 0;
8 int s = 0;
9 int a = 0;
10 if ( !(x >= 1) && (y >= 1)) {abort(); }
11 a = x;
12 b = y;
13 p = 1;
14 q = 0;
15 r = 0;
16 s = 1;
17 while (counter < 2) {
18    if(1 != p * s - r * q);//bug 1
19    if(a != y * r + x * p);//bug 2
20    if(b != x * q + y * s);//bug 3
21    if (!(a != b))break;
22     if (a > b) {
23      a = a - b;
24      p = p - q;
25      r = r - s;
26     }
27       else {
28        b = b - a;
29        q = q - p;
30        s = s - r;
31          }
32          counter++;
33 }
34 if(a - b != 0);//bug 4
35 if(p*x + r*y - b != 0);//bug 5
36 if(q*r - p*s + 1 != 0);//bug 6
37 if(q*x + s*y - b != 0);//bug 7
38 }
```

Fig. 20: egcd-ll_unwindbound2.c

```
1 void hard(int A, int B) {
2 int r = A;
3 int k = B;
4 int p = 1;
5   int q = 0;
6   int counter = 0;
7  while (counter < 10) {
8 if(q != 0);//bug 1
9 if(r != A);//bug 2
10    if(d != B * p);//bug 3
11  d = 2 * d;
12  p = 2 * p;
13  counter++;
14 }
15 int i = 0;
16 while ( i < 10) {
17  if(A != q*B + r);//bug4
18  if(d != B*p); //bug 5
19    d = d / 2;
20    p = p / 2;
21    if (r >= d) {
22     r = r - d;
23     q = q + p;
24  }
25     i++;
26}
27 if(A != d*q + r);//bug 6
28 if(B != d);//bug 7

}
```

Fig. 21:hard-ll_unwindbound10.c

```
1 void mannadiv(int x1, int x2 ){
2  int y1 = 0;
3 int y2 = 0;
4 int y3 = x1;
5 int i = 0;
6 while (i < 10) {
7   if(y1*x2 + y2 + y3 != x1);//bug 1
8 if (!(y3 != 0)) break;
9    if (y2 + 1 == x2) {
10     y1 = y1 + 1;
11     y2 = 0;
12     y3 = y3 - 1;
13  }
14  else {
15    y2 = y2 + 1;
16      y3 = y3 - 1;
17  }
18 i++;
19 }
20  if(y1*x2 + y2 != x1); // bug 2
21}
```

Fig. 22:mannadiv_unwindbound10.c

```
1 void prod4br(int f, int g){
2  long a = f;
3  long b = g;
4  long p = 1;
5  long q = 0;
6  int i = 0;
7  while (i < 5) {
8    if (q + a * b * p != ( long) f * g);//bug 1
9    if (!(a != 0 && b != 0))break;
10    if (a % 2 == 0 && b % 2 == 0) {
11        a = a / 2;
12        b = b / 2;
13        p = 4 * p;
14  }
15    else if (a % 2 == 1 && b % 2 == 0) {
16        a = a - 1;
17        q = q + b * p;
18  }
19    else if (a % 2 == 0 && b % 2 == 1) {
20        b = b - 1;
21        q = q + a * p;
22  }
23   else {
24      a = a - 1;
25      b = b - 1;
26      q = q + (a + b + 1) * p;
27  }
28         i++;
29 }
30 if (q != ( long long) f * g);//bug 2
31 if (a * b != 0); // bug 3
32 }
```

Fig. 23:prod4br-ll_unwindbound5.c

```
1 void prodbin (int a, int b ){
2  long  x = a;
3  long  y = b ;
4  long  z = 0;
5  int i= 0;
6  while (i < 10) {
7  if(z + x * y != (long long) a * b);//bug 1
8     if (!(y != 0))break;
9     if (y % 2 == 1) {
10          z = z + x;
11          y = y - 1;
12     }
13     x = 2 * x;
14     y = y / 2;
15     i++;
16 }
17  if((z != (long long)a * b)); //bug 2
18}
```

Fig. 24:prodbin-ll_unwindbound10.c

benchmarks/-/blob/main/c/nla-digbench-scaling/prodbin-ll_unwindbound10.c.

2 Supplemental material to Results section (RQ1 and RQ2)

We present programs in which KT-GDE3 finds bugs while DynaMOSA, WTS, KLEE and PEX fail. The corresponding java and C# programs can be found at: https:// github.com/stuartsemujju/ATCGPC.

2.1 Supplemental material to Results section RQ1

Fig. 6 in section 1.2 is a code snippet of loop program `vogal-1.c` . The java version of the program can be found at: https:// github.com/stuartsemujju/ATCGPC. The program has one assert statement in Line 25. A bug is witnessed when the negated condition of the assert statement triggered (i.e., if (cont_ aux!=cont)). DynaMOSA and WTS do not trigger the negated condition in all the 30 independent runs. KT-GDE3 finds the bug in all the 30 independent runs. For example, input input_string = {'$' , 'Y', '-' 'M' , 'A' } triggers the negated condition of the assert statement.

The next example in Fig. 19, is a code snippet of loop program divbin_unwindbound5.c. The java version of the program can be found at: https:// github.com/stuartsemujju/ATCGPC. The program has two assert statements in Line 17 and Line 26. A bug is witnessed when the negated condition of the assert statements in Line 17 and Line 26 are triggered. DynaMOSA and WTS do not trigger the negated condition of the both assert statement in all the 30 independent runs. KT-GDE3 finds the bug in all the 30 independent runs. For example, input `int A = 1546745821 , int B = 1489176611` triggers the negated condition of the assert statement in Line 14 and Line 21.

Additional information on test cases generated by KT-GDE3 to cover bugs can be found at https://github.com/stuartsemujju/ATCGPC.

2.2 Supplemental material to Results section RQ2

KLEE and PEX do not trigger the negated conditions in Line 25 in all the 30 independent runs for program `vogal-1.c` in Fig. 6. For example, KT-GDE3 input input_string = {'$' , 'Y', '-', 'M' , 'A' } triggers the negated condition of the assert statement. Fig. 17 in section 1.2 is a code snippet of loop program `djikastra`. Both KLEE and PEX do not trigger any of the negated conditions of the assert statements in all the 30 independent runs for program. KT-GDE3 is able to trigger one of the negated conditions. For example, KT-GDE3 triggers the negated condition in Line 17 in all the 30 independent runs for program `djikastra` with generated input n = -65357.

Additional information on test cases generated by KT-GDE3 to cover bugs can be found at https://github.com/stuartsemujju/ATCGPC.

3 Loop path enumeration

To enumerate target paths, our method extracts the simple routes from each loop and constructs the possible loop path possibilities from a k-bounded number of iterations. Details of this method can the found in the attached source code. See function enumeratePaths(...) which is called on line 351 and its implementation details can be found on line 891 of the file Ktgde3.java in the folder KT-GD3_sourcecode at https://github.com/stuartsemujju/ATCGPC.