Accelerating Array Constraints in Symbolic Execution

David Perry*1, Andrea Mattavelli*2, Xiangyu Zhang1, Cristian Cadar2

¹Department of Computer Science, Purdue University ²Department of Computing, Imperial College London

*The first two authors contributed equally to this paper

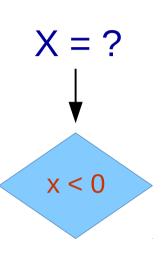


Imperial College London

```
int bad_abs(int x) {
  if(x < 0)
    return -x;
  if(x == 1234)
    return -x;
  return x;
}</pre>
```

```
int bad_abs(int x) {
    if(x < 0)
        return -x;
    if(x == 1234)
        return -x;
    return x;
}</pre>
```

```
int bad_abs(int x) {
  if(x < 0)
    return -x;
  if(x == 1234)
    return -x;
  return x;
}</pre>
```



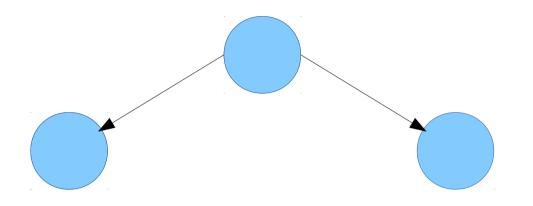
```
int bad abs(int x) {
                                    X = ?
  if(x < 0)
      return -x;
                             True
                                    x < 0
  if(x == 1234)
      return -x;
                        return -x
   return x;
                         x = -2
                       Test1.out
```

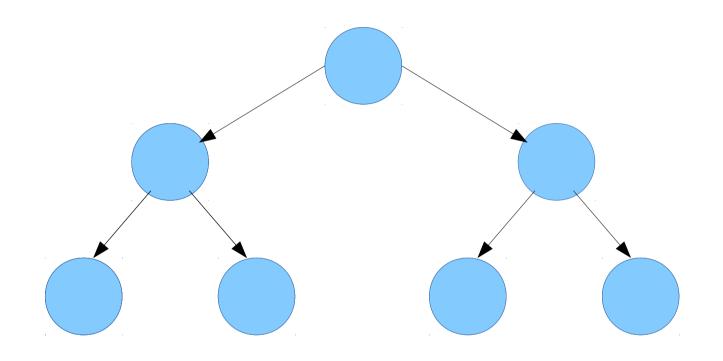
```
int bad abs(int x) {
                                    X = ?
  if(x < 0)
      return -x;
                             True
                                             False
                                     x < 0
   if(x == 1234)
      return -x;
                        return -x
                                              x==1234
   return x;
                         x = -2
                        Test1.out
```

```
int bad abs(int x) {
                                      X = ?
   if(x < 0)
      return -x;
                               True
                                                False
                                       x < 0
   if(x == 1234)
      return -x;
                          return -x
                                          True
                                                x = = 1234
   return x;
                          x = -2
                                      return -x
                         Test1.out
                                     x = 1234
                                     Test2.out
```

```
int bad abs(int x) {
                                       X = ?
   if(x < 0)
      return -x;
                               True
                                                 False
                                        x < 0
   if(x == 1234)
      return -x;
                          return -x
                                           True
                                                            False
                                                 x = = 1234
   return x;
                           x = -2
                                      return -x
                                                            return -x
                         Test1.out
                                      x = 1234
                                                             x = 1
                                                           Test3.out
                                      Test2.out
```

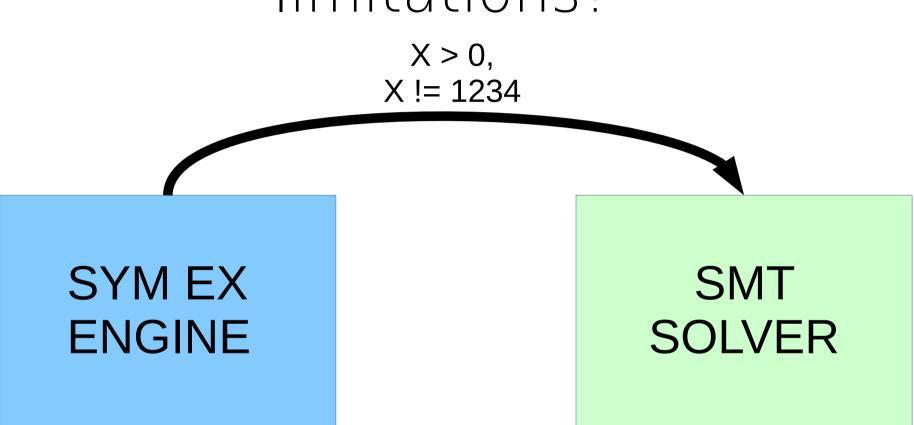


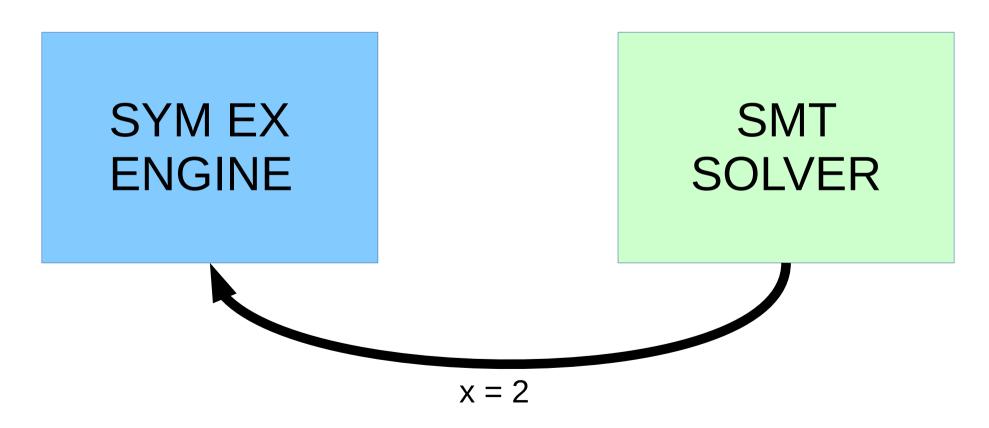






SYM EX ENGINE SMT SOLVER





largeArray[symIdx] != symVar

SYM EX ENGINE SMT SOLVER



A Challenging Example: BC

```
Void tokenMatch(char *input) {
  Unsigned currState = 0;
 Char charPtr = input;
 Do {
   Char currClass = equivClass[*charPtr];
   if(accept[currState]) {
     LastAcceptState = currState;
     LastAcceptPos = charPtr;
   while(check[base[currState] + currClass] != currState) {
     CurrState = def[currState];
     if(currState >= 298) {CurrClass = meta[currClass];}
   CurrState = next[base[currState] + currClass]; charPtr++;
 } while(base[currState] != 526);
```

A Challenging Example: BC

```
Void tokenMatch(char *input) {
  Unsigned currState = 0;
 Char charPtr = input;
 Do {
   Char currClass = equivClass[*charPtr];
   if(accept[currState]) {
     LastAcceptState = currState;
     LastAcceptPos = charPtr;
   while(check[base[currState] + currClass] != currState)
     Currolate – ucijeurrolatej,
     if(currState >= 298) {CurrClass = meta[currClass];}
   CurrState = next[hase[currState] + currClass]; charPtr++;
 } while(base[currState] != 526);
```

```
int small[5] = \{0,3,2,2,2\};
if(small[sym_idx] == 2) \{...\}
```

```
int small[5] = \{0,3,2,2,2\};
if(small[sym idx] == 2) \{...\}
```

Create a variable and assign its value for each offset

```
- small_0 = 0 \land small_1 = 3 \land small_2 = 2 \land small_3 = 3 \land small_4 = 2
```

```
int small[5] = \{0,3,2,2,2\};
if(small[sym_idx] == 2) \{...\}
```

Create a variable and assign its value for each offset

```
- small_0 = 0 \land small_1 = 3 \land small_2 = 2 \land small_3 = 3 \land small_4 = 2
```

Add a constraint for the conditional

```
- val == small[sym_idx] == 2
```

```
int small[5] = \{0,3,2,2,2\};
if(small[sym_idx] == 2) \{...\}
```

- Create a variable and assign its value for each offset
 - $small_0 = 0 \land small_1 = 3 \land small_2 = 2 \land small_3 = 3 \land small_4 = 2$
- Add a constraint for the conditional
 - val == small[sym idx] == 2

Handle the read operation

```
- sym_idx = 0 \rightarrow val = small_0 \wedge sym_idx = 1 \rightarrow val = small_1 \wedge sym_idx = 2 \rightarrow val = small_2 \wedge sym_idx = 3 \rightarrow val = small_3 \wedge sym_idx = 4 \rightarrow val = small_4
```

```
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, -1, -1, -1, -1, -1, -1, 26, 27,
28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
};
unsigned isBase64(unsigned k) {
              if(k > 255)
                             return -1:
              if(b64[k] >= 0)
                             return 1:
              else return 0:
```

```
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, -1, -1, -1, -1, -1, -1, 26, 27,
28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
};
unsigned isBase64(unsigned k) {
                    if(k > 255)
                                        return 1,
                    if(b64[k] >= 0)
                                  return I
                    else return 0;
```

b64[0]=-1 Λ b64[1]=-1 Λ b64[2]=-1 Λ b64[3]=-1 Λ b64[4]=-1 Λ b64[5]=-1 Λ b64[6]=-1 Λ b64[7]=-1 Λ b64[8]=-1 Λ b64[9]=-1 Λ b64[10]=-1 Λ b64[11]=-1 Λ b64[12]=-1 Λ b64[13]=-1 Λ b64[14]=-1 Λ b64[15]=-1 Λ b64[16]=-1 Λ b64[17]=-1 Λ b64[18]=-1 Λ b64[19]=-1 Λ b64[20]=-1 Λ b64[21]=-1 Λ b64[22]=-1 Λ b64[23]=-1 Λ b64[24]=-1 Λ b64[25]=-1 Λ b64[26]=-1 Λ b64[27]=-1 Λ b64[28]=-1 Λ b64[29]=-1 Λ b64[30]=-1 Λ b64[31]=-1 Λ b64[32]=-1 \(\text{b64[33]=-1 \(\text{b64[34]=-1 \(\text{b64[35]=-1 \(\text{b64[36]=-1 \(\text{b64[37]=-1 \(\text{b64[38]=-1 \(\text{b64[39]=-1 \(\text{b64[40]=-1 \(\text{b64[41]=-1 \(\text{b64[42]=-1 \(\text{b64[38]=-1 \(\text{b64[39]=-1 \(\text{b64[39 1 Λ b64[43]=62 Λ b64[44]=-1 Λ b64[45]=-1 Λ b64[46]=-1 Λ b64[47]=63 Λ b64[48]=52 Λ b64[49]=53 Λ b64[50]=54 Λ b64[51]=55 Λ b64[52]=56 Λ b64[53]=57 Λ b64[54]=58 Λ b64[55]=59 Λ b64[56]=60 Λ b64[57]=61 Λ b64[58]=-1 Λ b64[59]=-1 Λ b64[60]=-1 Λ b64[61]=-1 Λ b64[62]=-1 Λ b64[63]=-1 \(\text{b64[64]=-1} \(\text{b64[65]=0} \(\text{b64[66]=1} \(\text{b64[67]=2} \(\text{b64[68]=3} \(\text{b64[69]=4} \(\text{b64[70]=5} \(\text{b64[71]=6} \(\text{b64[72]=7} \(\text{b64[73]=8} \(\text{A} \) b64[74]=9 Λ b64[75]=10 Λ b64[76]=11 Λ b64[77]=12 Λ b64[78]=13 Λ b64[79]=14 Λ b64[80]=15 Λ b64[81]=16 Λ b64[82]=17 Λ b64[83]=18 Λ b64[84]=19 Λ b64[85]=20 Λ b64[86]=21 Λ b64[87]=22 Λ b64[88]=23 Λ b64[89]=24 Λ b64[90]=25 Λ b64[91]=-1 Λ b64[92]=-1 Λ b64[93]=-1 Λ b64[94]=-1 Λ b64[95]=-1 Λ b64[96]=-1 Λ b64[97]=26 Λ b64[98]=27 Λ b64[99]=28 Λ b64[100]=29 Λ b64[101]=30 Λ b64[102]=31 Λ b64[103]=32 Λ b64[104]=33 Λ b64[105]=34 Λ b64[106]=35 Λ b64[107]=36 Λ b64[108]=37 Λ b64[109]=38 Λ b64[110]=39 Λ b64[111]=40 Λ b64[112]=41 Λ b64[113]=42 Λ b64[114]=43 Λ b64[115]=44 Λ b64[116]=45 Λ b64[117]=46 Λ b64[118]=47 Λ b64[119]=48 Λ b64[120]=49 Λ b64[121]=50 Λ b64[122]=51 Λ b64[123]=-1 Λ b64[124]=-1 Λ b64[125]=-1 Λ b64[126]=-1 Λ b64[127]=-1 Λ b64[128]=-1 Λ b64[129]=-1 Λ b64[130]=-1 Λ b64[131]=- $1 \land b641321=-1 \land b641331=-1 \land b641341=-1 \land b641351=-1 \land b641361=-1 \land b641371=-1 \land b641381=-1 \land b641391=-1 \land$ $1 \land b64[151] = -1 \land b64[152] = -1 \land b64[153] = -1 \land b64[154] = -1 \land b64[155] = -1 \land b64[156] = -1 \land b64[157] = -1 \land b64[157] = -1 \land b64[158] = -1 \land b64[158]$ b64[160]=-1 Λ b64[161]=-1 Λ b64[162]=-1 Λ b64[163]=-1 Λ b64[164]=-1 Λ b64[165]=-1 Λ b64[166]=-1 Λ b64[167]=-1 Λ b64[168]=-1 Λ b64[169]=-1 Λ b64[170]=-1 Λ b64[171]=-1 Λ b64[172]=-1 Λ b64[173]=-1 Λ b64[174]=-1 Λ b64[175]=-1 Λ b64[176]=-1 Λ b64[177]=-1 Λ b64[178]=-1 Λ b64[179]=-1 \(\text{b64[180]}=-1 \(\text{b64[181]}=-1 \(\text{b64[182]}=-1 \(\text{b64[183]}=-1 \(\text{b64[184]}=-1 \(\text{b64[185]}=-1 \(\text{b64[186]}=-1 \(\text{b64[187]}=-1 \(\text{b64[187]}=-1 \(\text{b64[188]}=-1 \(\text{b64[187]}=-1 \(\text{b 1 Λ b64[189]=-1 Λ b64[190]=-1 Λ b64[191]=-1 Λ b64[192]=-1 Λ b64[193]=-1 Λ b64[194]=-1 Λ b64[195]=-1 Λ b64[196]=-1 Λ b64[197]=-1 Λ b64[198]=-1 \(\text{b64[199]}=-1 \(\text{b64[200]}=-1 \(\text{b64[201]}=-1 \(\text{b64[202]}=-1 \(\text{b64[203]}=-1 \(\text{b64[204]}=-1 \(\text{b64[205]}=-1 \(\text{b64[206]}=-1 \(\text{b64[207]}=-1 \(\text{b 1 Λ b64[208]=-1 Λ b64[209]=-1 Λ b64[210]=-1 Λ b64[211]=-1 Λ b64[212]=-1 Λ b64[213]=-1 Λ b64[214]=-1 Λ b64[215]=-1 Λ b64[216]=-1 Λ b64[217]=-1 \(\lambda\) b64[218]=-1 \(\lambda\) b64[219]=-1 \(\lambda\) b64[220]=-1 \(\lambda\) b64[221]=-1 \(\lambda\) b64[223]=-1 \(\lambda\) b64[223]=-1 \(\lambda\) b64[224]=-1 \(\lambda\) b64[225]=-1 \(\lambda\) b64[226]=-1 $1 \land b64[227]=-1 \land b64[228]=-1 \land b64[229]=-1 \land b64[230]=-1 \land b64[231]=-1 \land b64[232]=-1 \land b64[233]=-1 \land b64[234]=-1 \land b64[235]=-1 \land b64[235]=$ b64[236]=-1 \(\text{b64[237]=-1} \(\text{b64[238]=-1} \(\text{b64[239]=-1} \(\text{b64[240]=-1} \(\text{b64[241]=-1} \(\text{b64[242]=-1} \(\text{b64[243]=-1} \(\text{b64[243]=-1} \(\text{b64[244]=-1} \(\text{b64[245]=-1} \) $1 \land b64[246]=-1 \land b64[247]=-1 \land b64[248]=-1 \land b64[249]=-1 \land b64[250]=-1 \land b64[251]=-1 \land b64[252]=-1 \land b64[253]=-1 \land b64[254]=-1 \land b64[254]=$ b64[255]=-1 A

 $B64k \ge 0 \land k \le 255$

```
k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow b64k=b64[1] \land k=2 \rightarrow b64k=b64[2] \land k=3 \rightarrow b64k=b64[3] \land k=4 \rightarrow b64k=b64[4] \land k=5 \rightarrow b64k=b64[5] \land k=6 \rightarrow b64k=b64[6] \land k=7 \rightarrow b64k=b64[7] \land k=8 \rightarrow b64k=b64[7] \land k=8 \rightarrow b64k=b64[7] \land k=8 \rightarrow b64k=b64[7] \land k=8 \rightarrow b64k=b64[8] \land k=8 
b64k = b64[8] \land k = 9 \rightarrow b64k = b64[19] \land k = 10 \rightarrow b64k = b64[10] \land k = 11 \rightarrow b64k = b64[11] \land k = 12 \rightarrow b64k = b64[12] \land k = 13 \rightarrow b64k = b64[13] \land k = 14 \rightarrow b64k = b64[14] \land k = 15 \rightarrow b64k = b64[15] \land k = 16 \rightarrow b64k = b64[16] \land k
   b64k=b64[16] \land k=17 \rightarrow b64k=b64[17] \land k=18 \rightarrow b64k=b64[18] \land k=19 \rightarrow b64k=b64[19] \land k=20 \rightarrow b64k=b64[20] \land k=21 \rightarrow b64k=b64[21] \land k=22 \rightarrow b64k=b64[22] \land k=23 \rightarrow b64k=b64[23] \land k=20 
k=24 \rightarrow b64k=b64[24] \land k=25 \rightarrow b64k=b64[25] \land k=26 \rightarrow b64k=b64[26] \land k=27 \rightarrow b64k=b64[27] \land k=28 \rightarrow b64k=b64[28] \land k=29 \rightarrow b64k=b64[29] \land k=30 \rightarrow b64k=b64[30] \land k=31 \rightarrow b64k=b64[30] 
   b64k = b64[31] \land k = 32 \rightarrow b64k = b64[32] \land k = 33 \rightarrow b64k = b64[33] \land k = 34 \rightarrow b64k = b64[34] \land k = 35 \rightarrow b64k = b64[35] \land k = 36 \rightarrow b64k = b64[36] \land k = 37 \rightarrow b64k = b64[37] \land k = 38 \rightarrow b64k = b64[38] \land k = b64[38]
   k=39 \rightarrow b64k=b64[39] \land k=40 \rightarrow b64k=b64[40] \land k=41 \rightarrow b64k=b64[41] \land k=42 \rightarrow b64k=b64[42] \land k=43 \rightarrow b64k=b64[43] \land k=44 \rightarrow b64k=b64[44] \land k=45 \rightarrow b64k=b64[45] \land k=46 \rightarrow b64k=b64[45] 
   b64k=b64[46] \land k=47 \rightarrow b64k=b64[47] \land k=48 \rightarrow b64k=b64[48] \land k=49 \rightarrow b64k=b64[49] \land k=50 \rightarrow b64k=b64[50] \land k=51 \rightarrow b64k=b64[51] \land k=52 \rightarrow b64k=b64[52] \land k=53 \rightarrow b64k=b64[53] \land k=50 
   k=54 \rightarrow b64k=b64[54] \land k=55 \rightarrow b64k=b64[55] \land k=56 \rightarrow b64k=b64[56] \land k=57 \rightarrow b64k=b64[57] \land k=58 \rightarrow b64k=b64[58] \land k=59 \rightarrow b64k=b64[59] \land k=60 \rightarrow b64k=b64[60] \land k=61 \rightarrow b64k=b64[58] \land k=59 \rightarrow b64k=b64[59] \land k=60 \rightarrow b64k=b64[60] \land k=61 \rightarrow b64k=b64[58] \land k=50 \rightarrow b64k=b64[58] 
   b64k=b64[61] \land k=62 \rightarrow b64k=b64[62] \land k=63 \rightarrow b64k=b64[63] \land k=64 \rightarrow b64k=b64[64] \land k=65 \rightarrow b64k=b64[65] \land k=66 \rightarrow b64k=b64[66] \land k=67 \rightarrow b64k=b64[67] \land k=68 \rightarrow b64k=b64[68] \land k=68 
k=69 \rightarrow b64k=b64[69] \land k=70 \rightarrow b64k=b64[70] \land k=71 \rightarrow b64k=b64[71] \land k=72 \rightarrow b64k=b64[72] \land k=73 \rightarrow b64k=b64[73] \land k=74 \rightarrow b64k=b64[74] \land k=75 \rightarrow b64k=b64[75] \land k=76 \rightarrow b64k=b64[75] \land k=76 \rightarrow b64k=b64[75] \land k=78 \rightarrow b64k=b64[75] 
   b64k=b64[76] \land k=77 \rightarrow b64k=b64[77] \land k=78 \rightarrow b64k=b64[78] \land k=79 \rightarrow b64k=b64[79] \land k=80 \rightarrow b64k=b64[80] \land k=81 \rightarrow b64k=b64[81] \land k=82 \rightarrow b64k=b64[82] \land k=83 \rightarrow b64k=b64[83] \land k=81 \rightarrow b64k=b64[81] \land k=82 \rightarrow b64k=b64[82] \land k=81 
   k=84 \rightarrow b64k=b64[84] \ \land \ k=85 \rightarrow b64k=b64[85] \ \land \ k=86 \rightarrow b64k=b64[86] \ \land \ k=87 \rightarrow b64k=b64[87] \ \land \ k=88 \rightarrow b64k=b64[88] \ \land \ k=89 \rightarrow b64k=b64[89] \ \land \ k=90 \rightarrow b64k=b64[90] \ \land \ k=91 \rightarrow b64k=b64[
   b64k=b64[91] \land k=92 \rightarrow b64k=b64[92] \land k=93 \rightarrow b64k=b64[93] \land k=94 \rightarrow b64k=b64[94] \land k=95 \rightarrow b64k=b64[95] \land k=96 \rightarrow b64k=b64[96] \land k=97 \rightarrow b64k=b64[97] \land k=98 \rightarrow b64k=b64[98] \land k=98 
k=99 \rightarrow b64k=b64[99] \land k=100 \rightarrow b64k=b64[100] \land k=101 \rightarrow b64k=b64[101] \land k=102 \rightarrow b64k=b64[102] \land k=103 \rightarrow b64k=b64[103] \land k=104 \rightarrow b64k=b64[104] \land k=105 \rightarrow b64k=b64[105] \land k=106 
k=106 \rightarrow b64k=b64[106] \ \land \ k=107 \rightarrow b64k=b64[107] \ \land \ k=108 \rightarrow b64k=b64[108] \ \land \ k=109 \rightarrow b64k=b64[109] \ \land \ k=110 \rightarrow b64k=b64[110] \ \land \ k=111 \rightarrow b64k=b64[111] \ \land \ k=112 \rightarrow b64k=b64[112] \ \land \ k=110 \rightarrow b64k=b64[110] \ \land \ k=110 \rightarrow b64k=b64[110] \ \land \ k=110 \rightarrow b64k=b64[110] \ \land \ k=111 \rightarrow b64k=b64[110] \ \land \ 
k=113 \rightarrow b64k=b64[113] \land k=114 \rightarrow b64k=b64[114] \land k=115 \rightarrow b64k=b64[115] \land k=116 \rightarrow b64k=b64[116] \land k=117 \rightarrow b64k=b64[117] \land k=118 \rightarrow b64k=b64[118] \land k=119 \rightarrow b64k=b64[119] \land k=110 \rightarrow b64k=b64[110] \land k=11
k=120 \rightarrow b64k=b64[120] \land k=121 \rightarrow b64k=b64[121] \land k=122 \rightarrow b64k=b64[122] \land k=123 \rightarrow b64k=b64[123] \land k=124 \rightarrow b64k=b64[124] \land k=125 \rightarrow b64k=b64[125] \land k=126 \rightarrow b64k=b64[126] \land k=120 \rightarrow b64k=b64[126] \land k=12
k=127 \rightarrow b64k=b64[127] \ \land \ k=128 \rightarrow b64k=b64[128] \ \land \ k=129 \rightarrow b64k=b64[129] \ \land \ k=130 \rightarrow b64k=b64[130] \ \land \ k=131 \rightarrow b64k=b64[131] \ \land \ k=132 \rightarrow b64k=b64[132] \ \land \ k=133 \rightarrow b64k=b64[133] \ \land \ k=130 \rightarrow b64k=b64[130] \ \land \ 
k=134 \rightarrow b64k=b64[134] \land k=135 \rightarrow b64k=b64[135] \land k=136 \rightarrow b64k=b64[136] \land k=137 \rightarrow b64k=b64[137] \land k=138 \rightarrow b64k=b64[138] \land k=139 \rightarrow b64k=b64[139] \land k=140 \rightarrow b64k=b64[140] \land k=14
k=141 \rightarrow b64k=b64[141] \land k=142 \rightarrow b64k=b64[142] \land k=143 \rightarrow b64k=b64[143] \land k=144 \rightarrow b64k=b64[144] \land k=145 \rightarrow b64k=b64[145] \land k=146 \rightarrow b64k=b64[146] \land k=147 \rightarrow b64k=b64[147] \land k=148 \rightarrow b64k=b64[147] \land k=148 \rightarrow b64k=b64[148] \land k=14
k=148 \rightarrow b64k=b64[148] \ \land \ k=149 \rightarrow b64k=b64[149] \ \land \ k=150 \rightarrow b64k=b64[150] \ \land \ k=151 \rightarrow b64k=b64[151] \ \land \ k=152 \rightarrow b64k=b64[152] \ \land \ k=153 \rightarrow b64k=b64[153] \ \land \ k=154 \rightarrow b64k=b64[154] \ \land \ k=150 \rightarrow b64k=b64[154] \ \land \ 
k=155 \rightarrow b64k=b64[155] \land k=156 \rightarrow b64k=b64[156] \land k=157 \rightarrow b64k=b64[157] \land k=158 \rightarrow b64k=b64[158] \land k=159 \rightarrow b64k=b64[159] \land k=160 \rightarrow b64k=b64[160] \land k=161 \rightarrow b64k=b64[161] \land k=160 \rightarrow b64k=b64[160] \land k=16
k=162 \rightarrow b64k=b64[162] \land k=163 \rightarrow b64k=b64[163] \land k=164 \rightarrow b64k=b64[164] \land k=165 \rightarrow b64k=b64[165] \land k=166 \rightarrow b64k=b64[166] \land k=167 \rightarrow b64k=b64[167] \land k=168 \rightarrow b64k=b64[168] \land k=16
k=169 \rightarrow b64k=b64[169] \ \land \ k=170 \rightarrow b64k=b64[170] \ \land \ k=171 \rightarrow b64k=b64[171] \ \land \ k=172 \rightarrow b64k=b64[172] \ \land \ k=173 \rightarrow b64k=b64[173] \ \land \ k=174 \rightarrow b64k=b64[174] \ \land \ k=175 \rightarrow b64k=b64[175] \ \land \ k=170 \rightarrow b64k=b64[170] \ \land \ 
k=176 \rightarrow b64k=b64[176] \land k=177 \rightarrow b64k=b64[177] \land k=178 \rightarrow b64k=b64[178] \land k=179 \rightarrow b64k=b64[179] \land k=180 \rightarrow b64k=b64[180] \land k=181 \rightarrow b64k=b64[181] \land k=182 \rightarrow b64k=b64[182] \land k=180 \rightarrow b64k=b64[181] \land k=181 \rightarrow b64k=b64[181] \land k=18
k=183 \rightarrow b64k=b64[183] \ \land \ k=184 \rightarrow b64k=b64[184] \ \land \ k=185 \rightarrow b64k=b64[185] \ \land \ k=186 \rightarrow b64k=b64[186] \ \land \ k=187 \rightarrow b64k=b64[187] \ \land \ k=188 \rightarrow b64k=b64[188] \ \land \ k=189 \rightarrow b64k=b64[189] \ \land \ k=180 \rightarrow b64k=b64[180] \ \land \ 
k=190 \to b64k=b64[190] \land k=191 \to b64k=b64[191] \land k=192 \to b64k=b64[192] \land k=193 \to b64k=b64[193] \land k=194 \to b64k=b64[194] \land k=195 \to b64k=b64[195] \land k=196 \to b64k=b64[196] \land k=198 \to b64k=b64[196] \land k=19
k=197 \rightarrow b64k=b64[197] \land k=198 \rightarrow b64k=b64[198] \land k=199 \rightarrow b64k=b64[199] \land k=200 \rightarrow b64k=b64[200] \land k=201 \rightarrow b64k=b64[201] \land k=202 \rightarrow b64k=b64[202] \land k=203 \rightarrow b64k=b64[203] \land k=200 \rightarrow b64k=b64[203] \land k=20
k=204 \rightarrow b64k=b64[204] \land k=205 \rightarrow b64k=b64[205] \land k=206 \rightarrow b64k=b64[206] \land k=207 \rightarrow b64k=b64[207] \land k=208 \rightarrow b64k=b64[208] \land k=209 \rightarrow b64k=b64[209] \land k=210 \rightarrow b64k=b64[210] \land k=208 \rightarrow b64k=b64[208] \land k=209 \rightarrow b64k=b64[208] \land k=20
k=211 \rightarrow b64k=b64[211] \ \land \ k=212 \rightarrow b64k=b64[212] \ \land \ k=213 \rightarrow b64k=b64[213] \ \land \ k=214 \rightarrow b64k=b64[214] \ \land \ k=215 \rightarrow b64k=b64[215] \ \land \ k=216 \rightarrow b64k=b64[216] \ \land \ k=217 \rightarrow b64k=b64[217] \ \land \ k=218 \rightarrow b64k=b64[218] \ \land \ 
k=218 \rightarrow b64k=b64[218] \land k=219 \rightarrow b64k=b64[219] \land k=220 \rightarrow b64k=b64[220] \land k=221 \rightarrow b64k=b64[221] \land k=222 \rightarrow b64k=b64[222] \land k=223 \rightarrow b64k=b64[223] \land k=224 \rightarrow b64k=b64[224] \land k=220 \rightarrow b64k=b64[224] \land k=22
k = 225 \rightarrow b64k = b64[225] \land k = 226 \rightarrow b64k = b64[226] \land k = 227 \rightarrow b64k = b64[227] \land k = 228 \rightarrow b64k = b64[228] \land k = 229 \rightarrow b64k = b64[229] \land k = 230 \rightarrow b64k = b64[230] \land k = 231 \rightarrow b64k = b64[231] \land k = 240 \rightarrow b64k = b64[230] \land 
k=232 \rightarrow b64k=b64[232] \ \land \ k=233 \rightarrow b64k=b64[233] \ \land \ k=234 \rightarrow b64k=b64[234] \ \land \ k=235 \rightarrow b64k=b64[235] \ \land \ k=236 \rightarrow b64k=b64[236] \ \land \ k=237 \rightarrow b64k=b64[237] \ \land \ k=238 \rightarrow b64k=b64[238] \ \land \ 
k=239 \rightarrow b64k=b64[239] \land k=240 \rightarrow b64k=b64[240] \land k=241 \rightarrow b64k=b64[241] \land k=242 \rightarrow b64k=b64[242] \land k=243 \rightarrow b64k=b64[243] \land k=244 \rightarrow b64k=b64[240] \land k=245 \rightarrow b64k=b64[245] \land k=248 \rightarrow b64k=b64[240] \land k=240 \rightarrow b64k=b64[240] \land k=24
k=246 \rightarrow b64k=b64[246] \land k=247 \rightarrow b64k=b64[247] \land k=248 \rightarrow b64k=b64[248] \land k=249 \rightarrow b64k=b64[249] \land k=250 \rightarrow b64k=b64[250] \land k=251 \rightarrow b64k=b64[251] \land k=252 \rightarrow b64k=b64[252] \land k=260 \rightarrow b64k=b64[250] \land k=26
k=253 \rightarrow b64k=b64[253] \land k=254 \rightarrow b64k=b64[254] \land k=255 \rightarrow b64k=b64[255]
```

b64[0]=-1 Λ b64[1]=-1 Λ b64[2]=-1 Λ b64[2]=-1 Λ b64[3]=-1 Λ b64[3]=-1 Λ b64[4]=-1 Λ b64[6]=-1 Λ b64[6]=-1 Λ b64[6]=-1 Λ b64[7]=-1 Λ b64[7]=-1 Λ b64[10]=-1 Λ b64 Λ b64[16]=-1 Λ b64[17]=-1 Λ b64[18]=-1 Λ b64[18]=-1 Λ b64[20]=-1 Λ b64[20]=-1 Λ b64[21]=-1 Λ b64[22]=-1 Λ b64[23]=-1 Λ b64[24]=-1 Λ b64[25]=-1 Λ b64[26]=-1 Λ b64[27]=-1 Λ b64[28]=-1 Λ b6 Λ b64[31]=-1 Λ b64[32]=-1 Λ b64[42]=-1 Λ b6 Λ b64[46]=-1 Λ b64[47]=63 Λ b64[48]=52 Λ b64[49]=53 Λ b64[50]=54 Λ b64[51]=55 Λ b64[52]=56 Λ b64[53]=57 Λ b64[54]=58 Λ b64[55]=59 Λ b64[55]=60 Λ b64[57]=61 Λ b64[58]=-1 Λ b64[59]=-1 Λ b64[60]=-1 Λ b64[61]=-1 Λ b64[62]=-1 Λ b64[63]=-1 Λ b64[63]=-1 Λ b64[63]=-1 Λ b64[64]=-1 Λ b64[65]=0 Λ b64[66]=1 Λ b64[66]=2 Λ b64[68]=3 Λ b64[69]=4 Λ b64[70]=5 Λ b64[71]=6 Λ b64[72]=7 Λ b64[73]=8 Λ b64[74]=9 Λ b64[75]=10 \(\) b64[76]=11 \(\) b64[76]=12 \(\) b64[78]=12 \(\) b64[78]=13 \(\) b64[79]=14 \(\) b64[80]=15 \(\) b64[80]=15 \(\) b64[82]=17 \(\) b64[83]=18 \(\) b64[83]=18 \(\) b64[84]=19 \(\) b64[85]=20 \(\) b64[85]=21 \(\) b64[87]=22 \(\) b64[87]=22 \(\) b64[87]=22 \(\) b64[87]=21 \(\) b64[87]=22 \(\) b64[87]=21 \(\) b64[87 b64[89]=24 \(\) b64[90]=25 \(\) b64[91]=-1 \(\) b64[92]=-1 \(\) b64[93]=-1 \(\) b64[93]=-1 \(\) b64[95]=-1 \(\) b64[95]=-1 \(\) b64[96]=-1 \(\) b64[96]=-26 \(\) b64[98]=27 \(\) b64[99]=28 \(\) b64[100]=29 \(\) b64[100]=29 \(\) b64[101]=30 \(\) b64[102]=31 \(\) b64[103]=32 Λ b64[104]=33 Λ b64[105]=34 Λ b64[106]=35 Λ b64[107]=36 Λ b64[108]=37 Λ b64[109]=38 Λ b64[110]=39 Λ b64[111]=40 Λ b64[112]=41 Λ b64[113]=42 Λ b64[114]=43 Λ b64[115]=44 Λ b64[116]=45 Λ b64[117]=46 Λ b64[118]=47 Λ b64[127]=48 Λ b64[120]=49 Λ b64[121]=50 Λ b64[122]=51 Λ b64[123]=-1 Λ b64[124]=-1 Λ b64[125]=-1 Λ b64[126]=-1 Λ b b64[129]=-1 Λ b64[130]=-1 Λ b64[130]=-1 Λ b64[131]=-1 Λ b64[132]=-1 Λ b64[133]=-1 Λ b64[133]=-1 Λ b64[135]=-1 Λ b64[137]=-1 Λ b64[136]=-1 Λ b $1 \land b64[143]=-1 \land b64[144]=-1 \land b64[145]=-1 \land b64[145]=-1 \land b64[145]=-1 \land b64[145]=-1 \land b64[145]=-1 \land b64[145]=-1 \land b64[155]=-1 \land b64[155]=$ b64[156]=-1 \(\lambda \) b64[157]=-1 \(\lambda \) b64[157]=-1 \(\lambda \) b64[157]=-1 \(\lambda \) b64[163]=-1 \(\la $1 \land b64[170]=-1 \land b64[171]=-1 \land b64[172]=-1 \land b64[172]=-1 \land b64[173]=-1 \land b64[173]=$ b64[183]=-1 Λ b64[184]=-1 Λ b64[185]=-1 Λ b64[185]=-1 Λ b64[185]=-1 Λ b64[186]=-1 Λ b64[186]=-1 Λ b64[195]=-1 Λ b64[195]=-1 Λ b64[196]=-1 Λ b $1 \land b64[197]=-1 \land b64[198]=-1 \land b64[209]=-1 \land b64[209]=$ b64[210]=-1 Λ b64[211]=-1 Λ b64[212]=-1 Λ b64[212]=-1 Λ b64[213]=-1 Λ b $1 \land b64[224]=-1 \land b64[225]=-1 \land b64[225]=-1 \land b64[226]=-1 \land b64[227]=-1 \land b64[228]=-1 \land b64[230]=-1 \land b64[230]=$ b64[237]=-1 Λ b64[238]=-1 Λ b64[248]=-1 Λ b64[249]=-1 Λ b 1 Λ b64[251]=-1 Λ b64[252]=-1 Λ b64[253]=-1 Λ b64[254]=-1 Λ b64[255]=-1 Λ

b64k ≥ 0 Λ

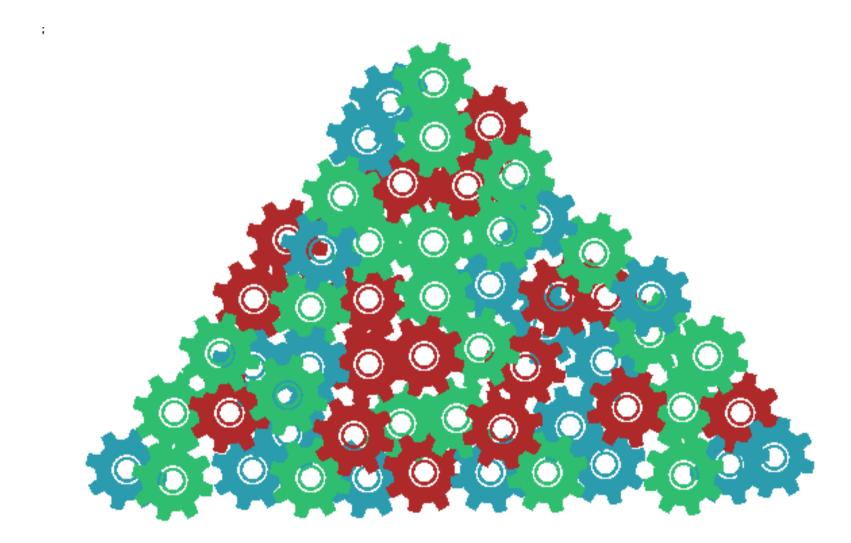
 $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow b64k=b64[1] \land k=2 \rightarrow b64k=b64[2] \land k=3 \rightarrow b64k=b64[3] \land k=4 \rightarrow b64k=b64[4] \land k=5 \rightarrow b64k=b64[5] \land k=6 \rightarrow b64k=b64[6] \land k=7 \rightarrow b64k=b64[7] \land k=8 \rightarrow b64k=b64[8] \land k=1 \rightarrow b64k=b64[8] \land k=1$ $k=9 \rightarrow b64k=b64[19] \land k=10 \rightarrow b64k=b64[10] \land k=11 \rightarrow b64k=b64[11] \land k=12 \rightarrow b64k=b64[12] \land k=13 \rightarrow b64k=b64[13] \land k=14 \rightarrow b64k=b64[14] \land k=15 \rightarrow b64k=b64[15] \land k=16 \rightarrow b64k=b64[16] \land k=17 \rightarrow b64k=b64[16] \land k=18 \rightarrow b64k=b64[16] \land$ → b64k=b64[17] ∧ k=18 → b64k=b64[18] ∧ k=19 → b64k=b64[19] ∧ k=20 → b64k=b64[20] ∧ k=21 → b64k=b64[21] ∧ k=23 → b64k=b64[23] ∧ k=23 → b64k=b64[23] ∧ k=24 → b64k=b64[24] ∧ k=25 → b64k=b64[24] ∧ k=26 → b64k=b64[24] $b64k = b64[25] \land k = 26 \rightarrow b64k = b64[25] \land k = 27 \rightarrow b64k = b64[27] \land k = 28 \rightarrow b64k = b64[28] \land k = 29 \rightarrow b64k = b64[29] \land k = 31 \rightarrow b64k = b64[31] \land k = 32 \rightarrow b64k = b64[32] \land k = 33 \rightarrow b64k = b64[32] \land k = 31 \rightarrow b64k = b64[32] \land k = b64k = b64[32] \land k = 31 \rightarrow b64k = b64[32] \land k =$ $b64k = b64[33] \land k = 34 \rightarrow b64k = b64[34] \land k = 35 \rightarrow b64k = b64[35] \land k = 36 \rightarrow b64k = b64[36] \land k = 37 \rightarrow b64k = b64[37] \land k = 38 \rightarrow b64k = b64[38] \land k = 39 \rightarrow b64k = b64[39] \land k = 40 \rightarrow b64k = b64[40] \land k = 41 \rightarrow b64k = b64[38] \land k = 30 \rightarrow b64k = b64[38] \land$ $b64k = b64\hat{1}41\hat{1} \land k = 42 \rightarrow b64k = b64\hat{1}42\hat{1} \land k = 43 \rightarrow b64k = b64\hat{1}43\hat{1} \land k = 44 \rightarrow b64k = b64\hat{1}43\hat{1} \land k = 45 \rightarrow b64k = b64\hat{1}45\hat{1} \land k = 46 \rightarrow b64k = b64\hat{1}46\hat{1} \land k = 47 \rightarrow b64k = b64\hat{1}47\hat{1} \land k = 48 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 49 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 48 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 49 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 48 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 49 \rightarrow b64k = b64\hat{1}48\hat{1} \land k = 4$ $b64k = b64[54] \land k = 50 \rightarrow b64k = b64[54] \land k = 50 \rightarrow b64k = b64[54] \land k = 50 \rightarrow b64k = b64[55] \land k = 50 \rightarrow b64k = b64[55] \land k = 50 \rightarrow b64k = b64[56] \land$ $b64k = b64[57] \land k = 58 \rightarrow b64k = b64[58] \land k = 59 \rightarrow b64k = b64[59] \land k = 60 \rightarrow b64k = b64[60] \land k = 61 \rightarrow b64k = b64[61] \land k = 62 \rightarrow b64k = b64[62] \land k = 63 \rightarrow b64k = b64[63] \land k = 64 \rightarrow b64k = b64[64] \land k = 65 \rightarrow b64k = b64[64] \land k = 64 \rightarrow b64k = b64[64] \land$ $b64k = b64[65] \land k = 66 \rightarrow b64k = b64[65] \land k = 67 \rightarrow b64k = b64[67] \land k = 68 \rightarrow b64k = b64[68] \land k = 69 \rightarrow b64k = b64[69] \land k = 70 \rightarrow b64k = b64[70] \land k = 71 \rightarrow b64k = b64[71] \land k = 72 \rightarrow b64k = b64[72] \land k = 73 \rightarrow b64k = b64[72] \land$ $b64k = b64[73] \land k = 74 \rightarrow b64k = b64[74] \land k = 75 \rightarrow b64k = b64[75] \land k = 76 \rightarrow b64k = b64[76] \land k = 77 \rightarrow b64k = b64[77] \land k = 79 \rightarrow b64k = b64[79] \land k = 80 \rightarrow b64k = b64[80] \land k = 81 \rightarrow b64k = b64[78] \land k = 70 \rightarrow b64k = b64[78] \land$ $b64k = b64[81] \land k = 82 \rightarrow b64k = b64[82] \land k = 83 \rightarrow b64k = b64[83] \land k = 84 \rightarrow b64k = b64[83] \land k = 85 \rightarrow b64k = b64[85] \land k = 87 \rightarrow b64k = b64[87] \land k = 88 \rightarrow b64k = b64[88] \land k = 89 \rightarrow b64k = b64[88] \land k = 80 \rightarrow b64[88] \land k$ $b64k = b64[99] \land k = 90 \rightarrow b64k = b64[91] \land k = 91 \rightarrow b64k = b64[91] \land k = 92 \rightarrow b64k = b64[92] \land k = 93 \rightarrow b64k = b64[93] \land k = 94 \rightarrow b64k = b64[94] \land k = 95 \rightarrow b64k = b64[95] \land k = 96 \rightarrow b64k = b64[96] \land$ $b64k=b64[97] \land k=98 \rightarrow b64k=b64[98] \land k=99 \rightarrow b64k=b64[99] \land k=100 \rightarrow b64k=b64[100] \land k=101 \rightarrow b64k=b64[101] \land k=102 \rightarrow b64k=b64[102] \land k=103 \rightarrow b64k=b64[103] \land k=104 \rightarrow b64k=b64[104] \land k=104 \rightarrow b64k=b64[$ $k = 105 \rightarrow b64k = b64[105] \ \land \ k = 106 \rightarrow b64k = b64[106] \ \land \ k = 107 \rightarrow b64k = b64[107] \ \land \ k = 108 \rightarrow b64k = b64[108] \ \land \ k = 109 \rightarrow b64k = b64[109] \ \land \ k = 111 \rightarrow b64k = b64[111] \ \land \ k = 112 \rightarrow b64k = b64[108] \ \land \ k = 100 \rightarrow b64k = b64[108] \$ $b64k = b64[112] \land k = 113 \rightarrow b64k = b64[113] \land k = 114 \rightarrow b64k = b64[114] \land k = 115 \rightarrow b64k = b64[115] \land k = 116 \rightarrow b64k = b64[116] \land k = 117 \rightarrow b64k = b64[117] \land k = 118 \rightarrow b64k = b64[118] \land k = 119 \rightarrow b64k = b64[118] \land k = 110 \rightarrow$ $b64k = b64[119] \land k = 120 \rightarrow b64k = b64[120] \land k = 121 \rightarrow b64k = b64[121] \land k = 122 \rightarrow b64k = b64[122] \land k = 123 \rightarrow b64k = b64[123] \land k = 124 \rightarrow b64k = b64[124] \land k = 125 \rightarrow b64k = b64[125] \land k = 126 \rightarrow b64k = b64[126] \land k = 126 \rightarrow$ $b64k = b64[126] \land k = 127 \rightarrow b64k = b64[127] \land k = 128 \rightarrow b64k = b64[128] \land k = 129 \rightarrow b64k = b64[129] \land k = 130 \rightarrow b64k = b64[130] \land k = 131 \rightarrow b64k = b64[131] \land k = 132 \rightarrow b64k = b64[132] \land k = 133 \rightarrow b64k = b64[130] \land k = 130 \rightarrow$ $b64k = b64\tilde{1}33\tilde{1} \land k = 134 \rightarrow b64k = b64\tilde{1}34\tilde{1} \land k = 135 \rightarrow b64k = b64\tilde{1}35\tilde{1} \land k = 136 \rightarrow b64k = b64\tilde{1}36\tilde{1} \land k = 137 \rightarrow b64k = b64\tilde{1}37\tilde{1} \land k = 138 \rightarrow b64k = b64\tilde{1}38\tilde{1} \land k = 139 \rightarrow b64k = b64\tilde{1}39\tilde{1} \land k = 130 \rightarrow b64k = b64\tilde{1}38\tilde{1} \land k = 130$ $b64k = b64[140] \land k = 141 \rightarrow b64k = b64[141] \land k = 142 \rightarrow b64k = b64[142] \land k = 143 \rightarrow b64k = b64[143] \land k = 144 \rightarrow b64k = b64[144] \land k = 145 \rightarrow b64k = b64[145] \land k = 146 \rightarrow b64k = b64[146] \land k = 147 \rightarrow b64k = b64[146] \land k = 148 \rightarrow$ $b64k = b64[147] \land k = 148 \rightarrow b64k = b64[148] \land k = 149 \rightarrow b64k = b64[149] \land k = 150 \rightarrow b64k = b64[150] \land k = 151 \rightarrow b64k = b64[151] \land k = 152 \rightarrow b64k = b64[152] \land k = 153 \rightarrow b64k = b64[153] \land k = 154 \rightarrow b64k = b64[154] \land k = 154 \rightarrow$ $b64k = b64[154] \land k = 155 \rightarrow b64k = b64[155] \land k = 156 \rightarrow b64k = b64[156] \land k = 157 \rightarrow b64k = b64[157] \land k = 158 \rightarrow b64k = b64[158] \land k = 150 \rightarrow b64k = b64[159] \land k = 160 \rightarrow b64k = b64[160] \land k = 160 \rightarrow$ $b64k = b64[161] \land k = 162 \rightarrow b64k = b64[162] \land k = 163 \rightarrow b64k = b64[163] \land k = 164 \rightarrow b64k = b64[164] \land k = 165 \rightarrow b64k = b64[165] \land k = 167 \rightarrow b64k = b64[167] \land k = 168 \rightarrow b64k = b64[167] \land k = 168 \rightarrow b64k = b64[168] \land k = 168 \rightarrow$ $b64k = b64[168] \land k = 169 \rightarrow b64k = b64[170] \land k = 170 \rightarrow b64k = b64[170] \land k = 171 \rightarrow b64k = b64[171] \land k = 172 \rightarrow b64k = b64[172] \land k = 173 \rightarrow b64k = b64[173] \land k = 174 \rightarrow b64k = b64[174] \land k = 175 \rightarrow$ $b64k=b64[175] \land k=176 \rightarrow b64k=b64[176] \land k=177 \rightarrow b64k=b64[177] \land k=178 \rightarrow b64k=b64[178] \land k=179 \rightarrow b64k=b64[179] \land k=180 \rightarrow b64k=b64[180] \land k=181 \rightarrow b64k=b64[181] \land k=182 \rightarrow b64k=b64[180] \land k=181 \rightarrow b64k$ $b64k = b64[182] \land k = 183 \rightarrow b64k = b64[183] \land k = 184 \rightarrow b64k = b64[184] \land k = 185 \rightarrow b64k = b64[185] \land k = 186 \rightarrow b64k = b64[186] \land k = 187 \rightarrow b64k = b64[187] \land k = 188 \rightarrow b64k = b64[188] \land k = 189 \rightarrow b64k = b64[188] \land k = 180 \rightarrow$ $b64k = b64[189] \land k = 190 \rightarrow b64k = b64[190] \land k = 191 \rightarrow b64k = b64[191] \land k = 192 \rightarrow b64k = b64[192] \land k = 193 \rightarrow b64k = b64[193] \land k = 194 \rightarrow b64k = b64[194] \land k = 195 \rightarrow b64k = b64[195] \land k = 196 \rightarrow b64k = b64[196] \land k = 196 \rightarrow$ $b64k = b64[196] \land k = 197 \rightarrow b64k = b64[197] \land k = 198 \rightarrow b64k = b64[198] \land k = 199 \rightarrow b64k = b64[199] \land k = 200 \rightarrow b64k = b64[200] \land k = 201 \rightarrow b64k = b64[201] \land k = 202 \rightarrow b64k = b64[202] \land k = 203 \rightarrow$ $b64k = b64[203] \land k = 204 \rightarrow b64k = b64[204] \land k = 205 \rightarrow b64k = b64[205] \land k = 206 \rightarrow b64k = b64[206] \land k = 207 \rightarrow b64k = b64[207] \land k = 208 \rightarrow b64k = b64[208] \land k = 209 \rightarrow b64k = b64[208] \land k = 209 \rightarrow b64k = b64[208] \land k = 200 \rightarrow$ $b64k = b64[217] \land k = 218 \rightarrow b64k = b64[218] \land k = 219 \rightarrow b64k = b64[219] \land k = 220 \rightarrow b64k = b64[220] \land k = 221 \rightarrow b64k = b64[221] \land k = 222 \rightarrow b64k = b64[222] \land k = 223 \rightarrow b64k = b64[223] \land k = 224 \rightarrow$ $b64k = b64[224] \land k = 225 \rightarrow b64k = b64[225] \land k = 226 \rightarrow b64k = b64[226] \land k = 227 \rightarrow b64k = b64[227] \land k = 228 \rightarrow b64k = b64[228] \land k = 229 \rightarrow b64k = b64[229] \land k = 230 \rightarrow b64k = b64[220] \land k = 231 \rightarrow b64k = b64[220] \land k = 230 \rightarrow$ $b64k = b64[238] \land k = 239 \rightarrow b64k = b64[239] \land k = 240 \rightarrow b64k = b64[240] \land k = 241 \rightarrow b64k = b64[241] \land k = 242 \rightarrow b64k = b64[242] \land k = 243 \rightarrow b64k = b64[243] \land k = 244 \rightarrow b64k = b64[244] \land k = 245 \rightarrow b64k = b64[248] \land k = 246 \rightarrow b64k = b64[249] \land k = 246 \rightarrow$ $b64k=b64[245] \land k=246 \rightarrow b64k=b64[246] \land k=247 \rightarrow b64k=b64[247] \land k=248 \rightarrow b64k=b64[248] \land k=249 \rightarrow b64k=b64[249] \land k=250 \rightarrow b64k=b64[250] \land k=251 \rightarrow b64k=b64[251] \land k=252 \rightarrow b64k=b64[248] \land k=248 \rightarrow b64k$ $h64k = h64\hat{1}252\hat{1}$ A $k = 253 \rightarrow h64k = h64\hat{1}253\hat{1}$ A $k = 254 \rightarrow h64k = h64\hat{1}254\hat{1}$ A $k = 255 \rightarrow h64k = h64\hat{1}253\hat{1}$

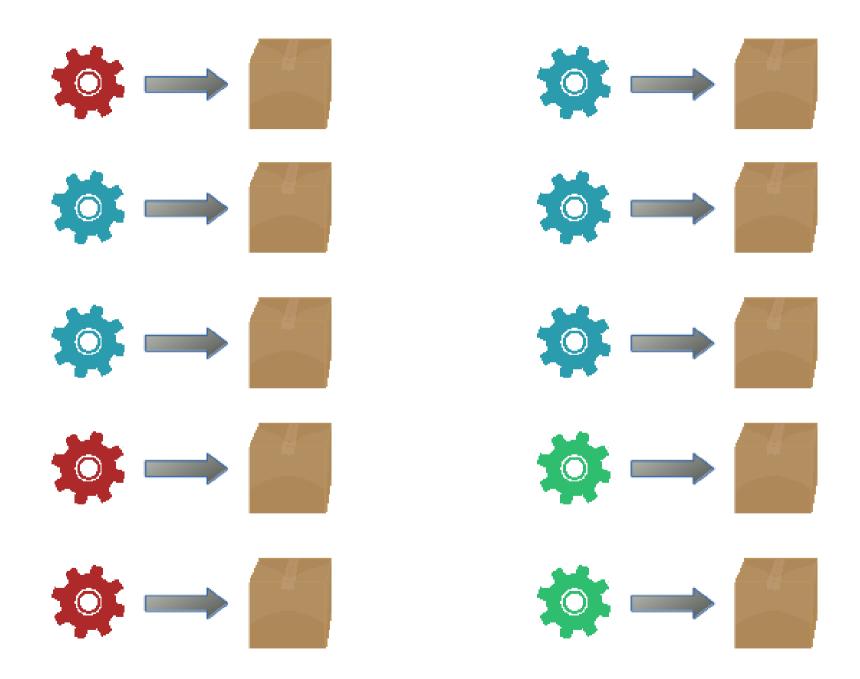
Impact on the Solver

 This explosion of variables creates an incredibly large search space

 Solver wastes time exploring multiple solutions with different indexes that point to the same array value

 Complicated queries make the performance even worse as backtracking results in more redundant exploration







Our Technique

 Takes advantage of statically and dynamically available information about the target array

Applies one of two optimizations depending on how the array read is used

 Yields a simplified query that results in dramatic performance improvement in programs that read from large arrays with symbolic indexes

Index-Based Transformation

- Only applicable to conditionals comparing array reads with statically known values
- Turns the comparison operation into a conjunctive formula that compares index ranges without the explicit array read
- Avoids the overhead of representing values in the array you don't need
- Creates the most concise encoding of either of our transformations

Original Constraints	Opt Constraints

37

Target Program	Original Constraints	Opt Constraints
<pre>isBase64(unsigned k) { if(k > 255) return -1; if(b64[k] >= 0) return 1; else return 0; }</pre>	// Array Variables $b64[0]=-1 \land b64[1]=-1 \land \land$ //Array Conditional $b64k >= 0 \land$ // Read Operation $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ $b64k=b64[1] \land$	

Target Program	Original Constraints	Opt Constraints
<pre>isBase64(unsigned k) { if(k > 255) return -1; if(b64[k] >= 0) return 1; else return 0; }</pre>	// Array Variables $b64[0]=-1 \land b64[1]=-1 \land \land$ //Array Conditional $b64k >= 0 \land$ // Read Operation $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ $b64k=b64[1] \land$	k = 43 v

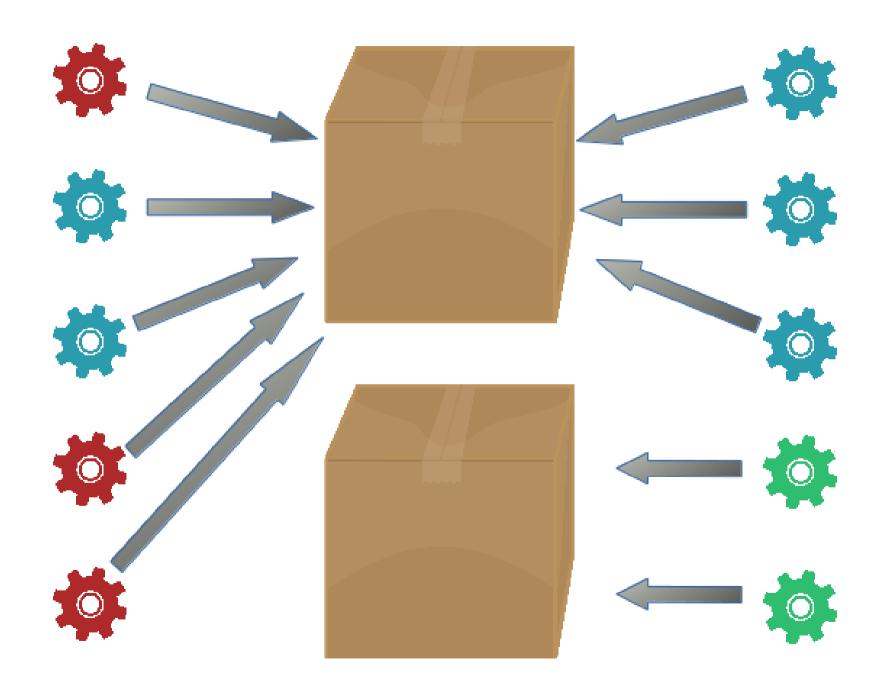
39

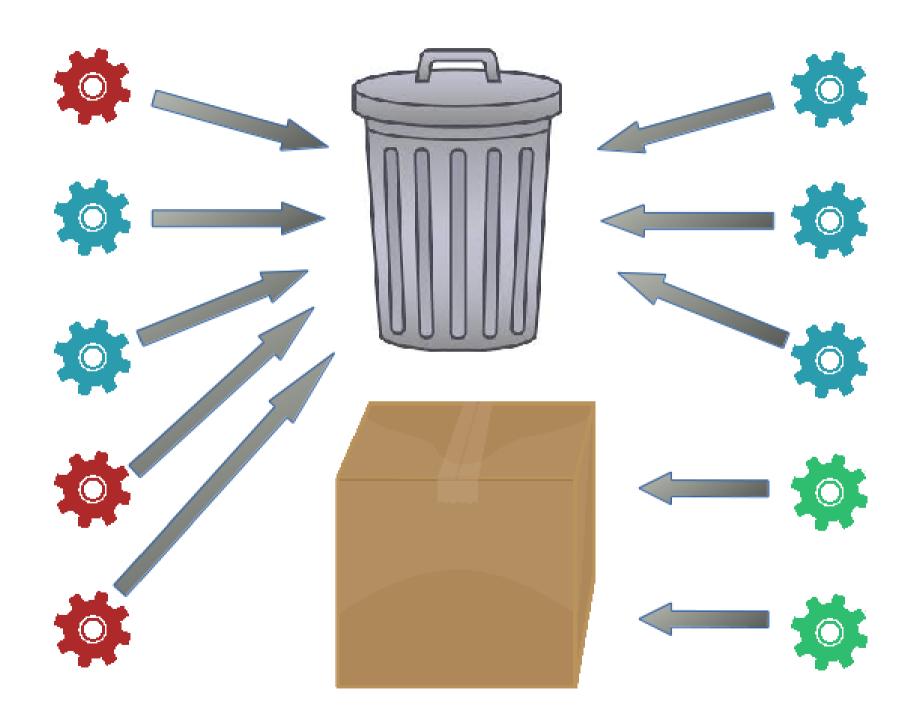
```
Opt Constraints
Target Program
                             Original Constraints
                                                                       k = 43 \text{ V}
isBase64(unsigned k) {
                             // Array Variables
     if(k > 255)
                             b64[0]=-1 Λ b64[1]=-1 Λ ... Λ
          return -1:
                             //Array Conditional
                                                                       47 <= k <= 57 v
     if(b64[k] >= 0)
                             b64k >= 0 A
          return 1:
                             // Read Operation
     else return 0;
                             k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow
                                                                                                 40
                             b64k=b64[1] \( \ldots \) ...
```

```
Opt Constraints
Target Program
                             Original Constraints
                                                                       k = 43 \text{ V}
isBase64(unsigned k) {
                             // Array Variables
     if(k > 255)
                             b64[0]=-1 Λ b64[1]=-1 Λ ... Λ
          return -1:
                             //Array Conditional
                                                                       47 <= k <= 57 v
     if(b64[k] >= 0)
                             b64k >= 0 A
          return 1:
                             // Read Operation
                                                                       65 <= k <= 90
     else return 0;
                             k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow
                             b64k=b64[1] \( \ldots \) ...
```

41

Opt Constraints Target Program **Original Constraints** k = 43 VisBase64(unsigned k) { // Array Variables if(k > 255)b64[0]=-1 Λ b64[1]=-1 Λ ... Λ return -1: //Array Conditional 47 <= k <= 57 v if(b64[k] >= 0)b64k >= 0 Areturn 1: // Read Operation 65 <= k <= 90 else return 0; $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ 42 97 <= k <= 122 b64k=b64[1] \(\dagger \)...





Applicable to all array reads

 Turns the array read into a series of nested ITE's comparing the index to a range of offsets that correspond to the same value

Removes the redundancy created by the traditional Theory of Arrays

Target Program	Original Constraints	Opt Constraints
isBase64(unsigned k) { if($k > 255$)		
return -1;		
if(b64[k] >= sym) return 1;		
else return 0;		
}		

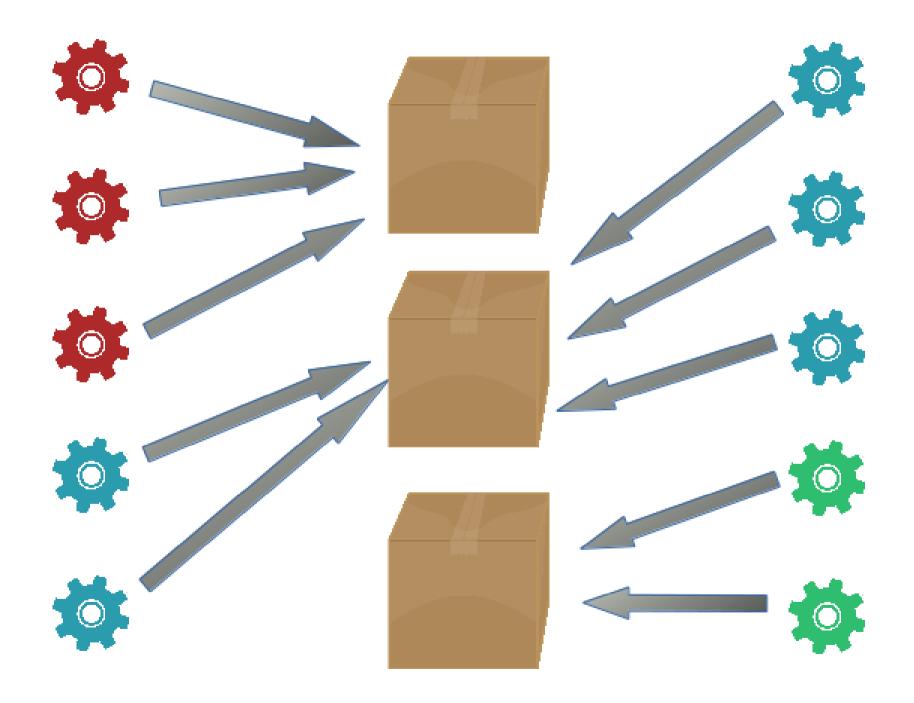
Target Program	Original Constraints	Opt Constraints
<pre>isBase64(unsigned k) { if(k > 255) return -1; if(b64[k] >= sym) return 1; else return 0; }</pre>	// Array Variables $b64[0]=-1 \land b64[1]=-1 \land \land$ //Array Conditional $b64k >= sym \land$ // Read Operation $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ $b64k=b64[1] \land$	

```
-1, 62, -1, -1, -1, 63, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1,
-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, -1, -1, -1, -1, -1, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1, -1, -1, -1, -1, -1, -1, -1,
|-1. -1. -1. -1. -1|
Target Program
               Original Constraints
                                 Opt Constraints
isBase64(unsigned k) {
                                x = Ite(0 \le k \le 42 \mid 44 \le k \le 46 \mid
               // Array Variables
   if(k > 255)
               b64[0]=-1 Λ b64[1]=-1 Λ ... Λ ..., -1,
     return -1:
               //Array Conditional
   if(b64[k] >= sym)
               b64k >= sym \Lambda
     return 1:
               // Read Operation
   else return 0;
               k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow
               b64k=b64[1] Λ ...
                                                 48
```

```
-1, <mark>62, -1, -1, -1</mark>, 63, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1,
<del>-1,</del> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, -1, -1, -1, -1, -1, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1, -1, -1, -1, -1, -1, -1, -1,
|-1. -1. -1. -1. -1|
Target Program
               Original Constraints
                                  Opt Constraints
                                  x = Ite(0 \le k \le 42 \mid 44 \le k \le 46 \mid
isBase64(unsigned k) {
               // Array Variables
   if(k > 255)
               b64[0]=-1 Λ b64[1]=-1 Λ ... Λ
     return -1:
               //Array Conditional
   if(b64[k] >= sym)
                                      ite(k == 43, 62,
               b64k >= sym \Lambda
     return 1:
               // Read Operation
   else return 0;
               k=0 \rightarrow b64k=b64[0] \wedge k=1 \rightarrow
               b64k=b64[1] Λ ...
                                                   49
```

```
-1, <mark>62, -1, -1, -1, 63,</mark> 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1,
<del>-1,</del> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, <del>-1, -1, -1, -1, -1, -1,</del> 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1, -1, -1, -1, -1, -1, -1, -1,
|-1. -1. -1. -1. -1|
Target Program
                Original Constraints
                                   Opt Constraints
                                   x = Ite(0 \le k \le 42 \mid 44 \le k \le 46 \mid
isBase64(unsigned k) {
                // Array Variables
   if(k > 255)
                b64[0]=-1 Λ b64[1]=-1 Λ ... Λ
     return -1:
                //Array Conditional
   if(b64[k] >= sym)
                                       ite(k == 43, 62,
                b64k >= sym \Lambda
     return 1:
                // Read Operation
   else return 0;
                k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow
                                         ite(k=47, 63,
                b64k=b64[1] Λ ...
                                                    50
```

```
-1, <mark>62, -1, -1, -1, 63,</mark> 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, -1, -1, -1, -1, -1, -1,
-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, -1, -1, -1, -1, -1, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1, -1, -1, -1, -1, -1, -1, -1,
|-1. -1. -1. -1. -1|
Target Program
               Original Constraints
                                  Opt Constraints
                                  x = Ite(0 \le k \le 42 \mid 44 \le k \le 46 \mid
isBase64(unsigned k) {
               // Array Variables
   if(k > 255)
               b64[0]=-1 Λ b64[1]=-1 Λ ... Λ
     return -1:
               //Array Conditional
   if(b64[k] >= sym)
               b64k >= sym \Lambda
                                      ite(k == 43, 62,
     return 1:
               // Read Operation
   else return 0;
               k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow
                                        ite(k=47, 63,
               b64k=b64[1] Λ ...
                                                   51
                                  x >= svm
```



What about partially symbolic arrays?

 Arrays can become partially symbolic due to program behavior or decisions made during analysis

 the Value-based transformation can be extended to handle them

 Add an individual case for each symbolic value in the array

Partially Symbolic Arrays

Target Program	Original Constraints	Opt Constraints	
<pre>isBase64(unsigned k) { b64[0] = sym0; b64[1] = sym1; b64[2] = sym2; b64[3] = sym3; if(b64[k] >= sym) return 1; else return 0;</pre>	// Array Variables $b64[0]=? \land b64[1]=? \land \land$ //Array Conditional $b64k >= sym \land$ // Read Operation $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ $b64k=b64[1] \land$		54
}			

Partially Symbolic Arrays

isBase64(unsigned k) { b64[0] = sym0; b64[1] = sym1; b64[2] = sym2; b64[3] = sym3; if(b64[k] >= sym) return 1; else return 0;

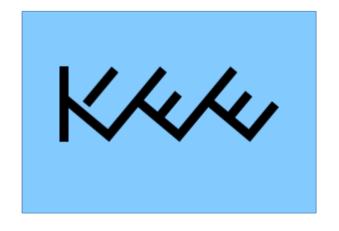
Original Constraints

// Array Variables $b64[0]=? \land b64[1]=? \land ... \land$ //Array Conditional $b64k >= sym \land$ // Read Operation $k=0 \rightarrow b64k=b64[0] \land k=1 \rightarrow$ $b64k=b64[1] \land ..$

Opt Constraints

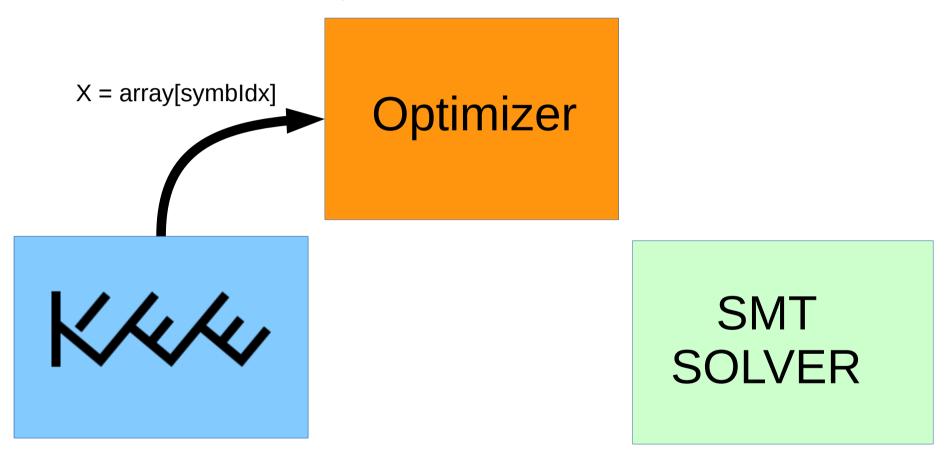
ite(k == 0, b64[0], ite(k==1, b64[1], ite(k==2, b64[2], ...

Optimizer

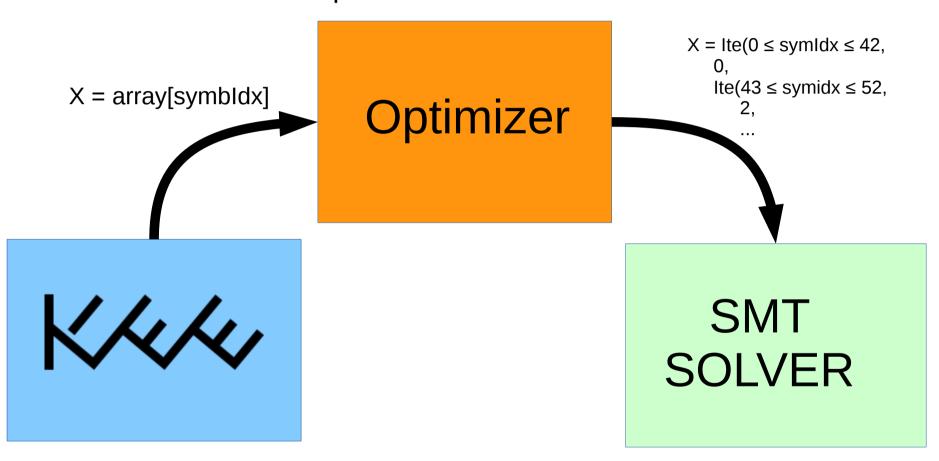


SMT SOLVER

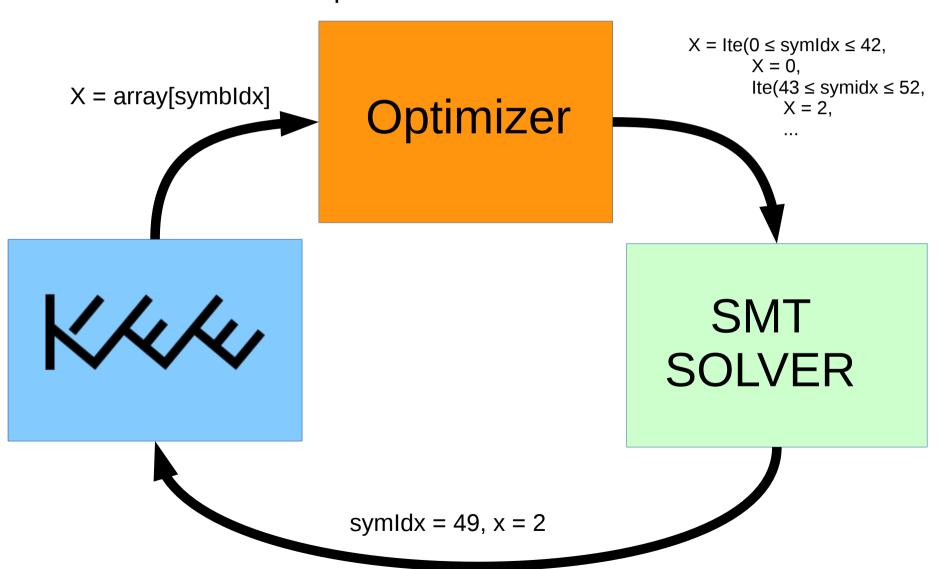
Implementation is available at: https://srg.doc.ic.ac.uk/projects/klee-array/artifact.html



Implementation is available at: https://srg.doc.ic.ac.uk/projects/klee-array/artifact.html



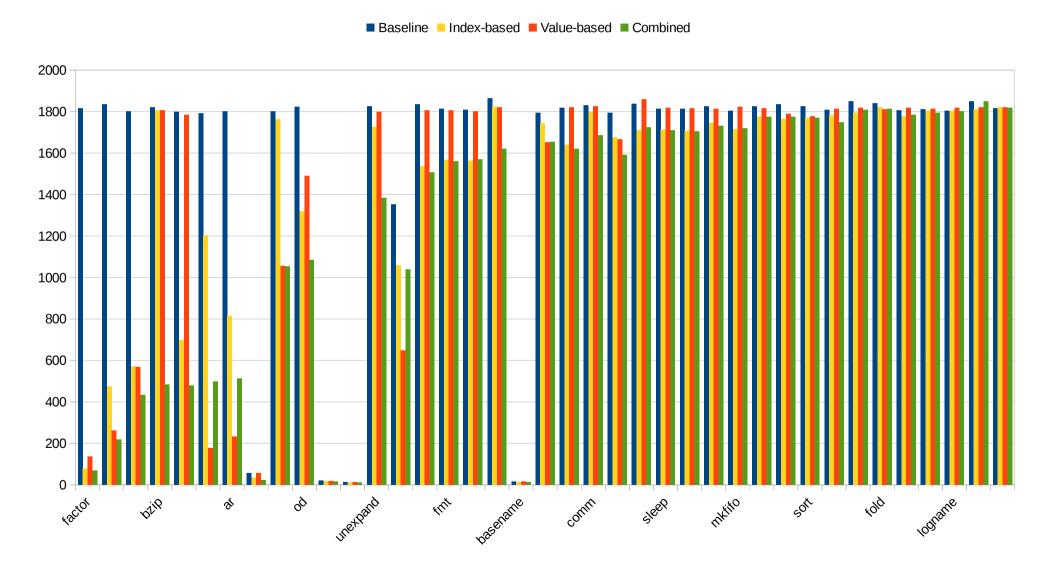
Implementation is available at: https://srg.doc.ic.ac.uk/projects/klee-array/artifact.html



Evaluation: Speed

- Performed on 104 programs from coreutils, binutils, and other open source repositories
- Baseline and Optimized runs execute the same number of instructions
- Instructions executed are logged to ensure fairness
- 7 programs with more than 3x speedup and none with slowdown

Evaluation: Speed



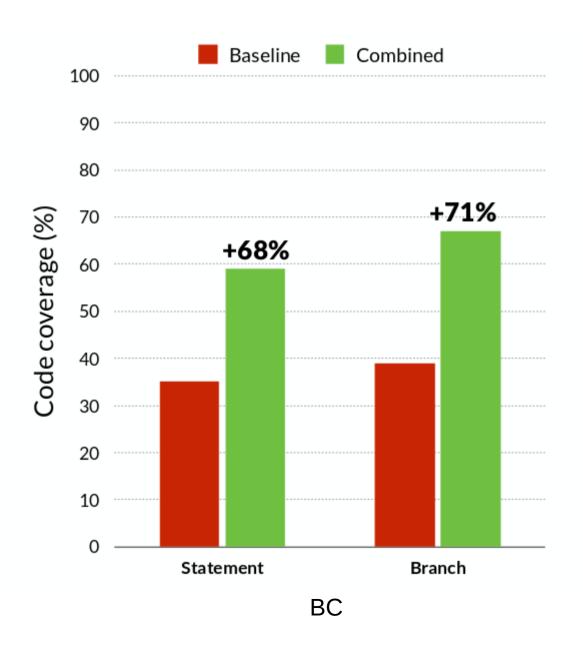
Evaluation: Code Coverage

Performed on the program BC

 Baseline and optimized runs use a heuristic based search strategy designed to explore uncovered code

 Both programs are analyzed for 6 hours creating inputs with the most code coverage possible

Evaluation: Code Coverage



Related Work

Constraint Optimization

- V. Ganesh and D. L. Dill. A decision procedure for bit-vectors and arrays. In Proc. of the 19th International Conference on Computer-Aided Veriication (CAV'07) July 2007.
- Erete and A. Orso. Optimizing constraint solving to better support symbolic execution. In Proc. of the Workshop on Constraints in Software Testing, Verification, and Analysis (CSTVA'11), Mar. 2011.
- S. Dong, O. Olivo, L. Zhang, and S. Khurshid. Studying the influence of standard compiler optimizations on symbolic execution. In Proc. of the 26th International Symposium on Software Reliability Engineering (ISSRE'15), Nov. 2015.

Conclusion

 Our technique uses static and dynamic information to build specific encodings for individual arrays

 This encoding strategy yields considerable improvement in SMT solver performance

 This speedup allows symbolic execution to explore more code at a faster rate

Q&A

Thank You!

David Perry

PhD Student, Purdue University

Contact: perry74@purdue.edu