Homework 5
COSE312, Spring 2023

Hakjoo Oh

Due: 05/31, 23:59

The goal of this assignment is to implement a "static" analyzer for SPY programs. The template code is available at

`https://github.com/kupl-courses/COSE312-2023spring/tree/main/homework/hw5`

Your job is to implement the `analyze` function in `analyzer`:

```
analyze :  Spy.program -> bool
```

which takes an SPY program and returns true iff the input program does not contain any type errors. Submit `analyzer.ml` only. Do not modify any files in the template code except for `analyzer.ml`.

Examples

1. The analyzer should return false for the following buggy program:

   ```
   x = 0
   x = "0"
   x + 1 # TypeError (str + int)
   ```

   while successfully verifying the safety of the fixed version:

   ```
   x = 0
   x = "0"
   int(x) + 1
   ```

2. Buggy:

   ```
   a = [1, 2, 3]
   a[2] + 'world'
   ```

   Safe:

   ```
   a = [1, 2, 3]
   a[2] + 2
   ```

3. Buggy:

   ```
   x = int(input())
   a = input()
   b = input()
   c = int(input())
   ```

```
if isinstance(x, int):
  c = a + b
  a = int(input())
  b = int(input())

d = a + b
e = c + d # TypeError (str + int)
```

Safe:

```
x = int(input())
a = input()
b = input()
c = int(input())

if isinstance(x, int):
  a = int(input())
  b = int(input())
  c = a + b

d = a + b
e = c + d
```

4. Buggy:

```
x = int(input())
y = int(input())

if y == 0:
  while y <= 100:
    y += 1

  if y == 101:
    x = "100"

print(x+y) # TypeError (str + int)
```

Safe:

```
x = int(input())
y = int(input())

if y == 0:
  while y <= 100:
    y += 1

  if y > 101:
    x = "100"

print(x+y)
```

5. Buggy:

```
def fib(x):
  if x < 2:
    return 1
  elif x < 3:
    return 'hello'
  else:
    return fib(x-1) + fib(x-2)    # TypeError (int + str)

fib(10)
```

Safe:

```
def fib(x):
  if x < 2:
    return 1
  else:
    return fib(x-1) + fib(x-2)

fib(10)
```

6. Buggy:

```
def foo(x):
  return x

def goo(x):
  if isinstance(x, int):
    if x < 100:
      return 50
  return 'hello'

t1 = goo(int(input()))       # type(t1) = {int, str}
t2 = goo(int(input()))       # type(t2) = {int, str}
t3 = foo(t1)          # type(t3) = {int, str}
t4 = foo(t2)          # type(t3) = {int, str}
t3 + t4               # TypeError (int + str, str + int)
```

Safe:

```
def foo(x):
  if isinstance(x, str):
    return len(x)
  return x

def goo(x):
  if isinstance(x, int):
    if x < 100:
      return 50
  return 'hello'

t1 = goo(int(input()))       # type(t1) = {int, str}
t2 = goo(int(input()))       # type(t2) = {int, str}
t3 = foo(t1)          # type(t3) = {int}
t4 = foo(t2)          # type(t4) = {int}
t3 + t4               # Safe
```

7. Buggy:

```
def foo(x):
  return [x, x, x]

a = foo(0)
b = foo('hello')
a[0] + b[2]   # Bug -  0 + 'hello'
```

Safe:

```
def foo(x):
  return [x, x, x]

a = foo(0)
b = foo('hello')
a[0] + 3
b[1] + 'world'
```

8. Buggy:

```
def is_one_digit(x) :
  if (x >= 0) and (x < 10) :
    return True
  else :
    return None

keysym = int(input())

if keysym >= 0 :
  if keysym < 16 :
    if int(input()) :
      if keysym >= 10 : pass
      else :
        key = is_one_digit(keysym) - 1
    else :
        key = is_one_digit(keysym) + 1  # TypeError: None + int
```

Safe:

```
def is_one_digit(x) :
  if (x >= 0) and (x < 10) :
    return True
  else :
    return None

keysym = int(input())

if keysym >= 0 :
  if keysym < 16 :
    if int(input()) :
      if keysym >= 10 : pass
      else :
        key = is_one_digit(keysym) - 1
```

```
else :
  if keysym >= 10 : pass
  else :
    key = is_one_digit(keysym) + 1
```