### Functional Program Recognition in Synthesis Michael B. James & Nadia Polikarpova





### Github Copilot is here

```
@@ -43,21 +43,35 @@ def __repr__(self) -> str:
      strToStr = FunType(ListType(CharType()), ListType(CharType()))
      intToInt = FunType(ListType(IntType()), ListType(IntType()))
46 - def antiunify(t1, t2, theta):
         if isinstance(t1, FunType) and isinstance(t2, FunType):
47 –
             targ, theta1 = antiunify(t1.arg, t2.arg, theta)
48 –
             tret, theta2 = antiunify(t1.ret, t2.ret, theta1)
49 –
             return FunType(targ, tret), theta2
50 –
         if isinstance(t1, ListType) and isinstance(t2, ListType):
51 –
52 –
             tcons, theta1 = antiunify(t1.cons, t2.cons, theta)
             return ListType(tcons), theta1
53 –
         if t1 == t2:
             return t1, theta
55 –
         if (t1, t2) in theta:
56 –
             return theta[(t1, t2)], theta
57 –
         a = FreeType()
58 –
         theta[(t1, t2)] = a
59 –
         return a, theta
60 –
```

43	strToStr = FunType(ListType(CharType()), ListType(CharType()))
44	intToInt = FunType(ListType(IntType()), ListType(IntType()))
45	
46	+ ## Anti-unification algorithm of two types t1 and t2, with theta as a substitution
47	+ ## Returns a tuple (t, theta) where t is the type that is the result of applying theta to t1 and t2
48	+ ## and theta is the substitution that makes t1 and t2 equal to t
40	

49 + def antiunify\_types(t1, t2, theta):

50		<b>if</b> t1 == t2:
51	+	return (t1, theta)
52	+	<pre>if isinstance(t1, FunType) and isinstance(t2, FunType):</pre>
53	+	<pre>if t1.arg == t2.arg:</pre>
54	+	<pre>return antiunify_types(t1.ret, t2.ret, theta)</pre>
55	+	<pre>elif t1.arg == t2.ret:</pre>
56	+	<pre>return antiunify_types(t1.arg, t2.arg, theta)</pre>
57	+	else:
58	+	return (None, None)
59	++	<pre>elif isinstance(t1, ListType) and isinstance(t2, ListType):</pre>
60	+	<pre>if t1.cons == t2.cons:</pre>
61	+	<pre>return antiunify_types(t1.cons, t2.cons, theta)</pre>
62	+	else:
63	+	return (None, None)
64	+	<pre>elif isinstance(t1, FreeType) and isinstance(t2, FreeType):</pre>
65	+	<pre>if t1.name == t2.name:</pre>
66	+	return (t1, theta)
67	+	else:
68	+	return (None, None)
69	+	<pre>elif isinstance(t1, FreeType):</pre>
70	+	<pre>return (t1, {t1.name: t2})</pre>
71	+	<pre>elif isinstance(t2, FreeType):</pre>
72	+	<pre>return (t2, {t2.name: t1})</pre>
73	+	else:
74	+	return (None, None)





# **Program Recognition** "Does this program solve my task?"

## Program

## Recognition

### Haskell: Functional, Highly Polymorphic



### Recognition

### Haskell: Functional, Highly Polymorphic





### Recognition

### Synthesis: Specification in, Program out

### Haskell: Functional, Highly Polymorphic





### Recognition

### Synthesis: Specification in, Program out

### Evaluation: Is the program that I need here?





### **Observational Study**



### Study Applications

### Future Studies

### **Observational Study**

- Think-aloud - 4 Haskellers - 2 - 10 years experience - 5 Tasks each How do functional programmers recognize the right program for their task?





{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD"snippet1 = ... snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply)



{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD"snippet1 = ... snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...testApplyNTimes = TestCase (assertEqual "applyNTimes" "00" (apply)





{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD"snippet1 = ... snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply)

# Task Description Stub



{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD"snippet1 = ... snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply)

#### Task Description

Stub

Snippets

{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a Stub applyNTimes f x n = error "TBD" snippet1 = ...snippet2 = ...snippet3 = ...Snippets snippet4 = ...snippet5 = ...testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply Test Case

Task Description

{-Function applyNTimes takes three inputs... -} applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD" snippet1 = ... snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply Test Case



#### 8







{-Function applyNTimes takes three inputs... -}

applyNTimes :: (a -> a) -> a -> Int -> a applyNTimes f x n = error "TBD"

snippet1 = ...snippet2 = ...snippet3 = ...snippet4 = ...snippet5 = ...

testApplyNTimes = TestCase(assertEqual "applyNTimes" "00" (apply Test Case



#### General Strategy

#### Higher-order snippets

#### Test Usage

#### Snippet Correction



- Multiple Passes - Hard tasks, more passes

#### General Strategy

```
10
     Function test takes two inputs, a boolean value b and an arbitrary value x.
     It returns an optional value, which contains x if and only if b is True.
13
14
     mbIfTrue :: Bool -> a -> Maybe a
15
     mbIfTrue b x = error "TBD'
17
     --mit snippet1 b x = Just x
     mit_snippet2 b x = bool (Just x) Nothing b
     mit_snippet3 b x = when b Nothing >> pure x
     mit_snippet4 b x = guard b >> pure x
     --mit\_snippet5 \ b \ x = bool (Just x) (Just x) b
22
23
24
     testMbIfTrue = TestCase
25
         (assertEqual "mbIfTrue" (Nothing) (mbIfTrue False 0))
26
27
        You can run the test in GHCi
```





- Multiple Passes - Hard tasks, more passes

#### General Strategy

```
10
     Function test takes two inputs, a boolean value b and an arbitrary value x.
     It returns an optional value, which contains x if and only if b is True.
13
14
     mbIfTrue :: Bool -> a -> Maybe a
15
     mbIfTrue b x = error "TBD'
17
     --mit snippet1 b x = Just x
     mit_snippet2 b x = bool (Just x) Nothing b
     mit_snippet3 b x = when b Nothing >> pure x
     mit_snippet4 b x = guard b >> pure x
     --mit\_snippet5 \ b \ x = bool (Just x) (Just x) b
22
23
24
     testMbIfTrue = TestCase
25
         (assertEqual "mbIfTrue" (Nothing) (mbIfTrue False 0))
26
27
        You can run the test in GHCi
```





applyNTimes applies a function, f, to an initial value, x, n times

applyNTimes :: (a -> a) -> a -> Int -> a

snippet1 f x n = last  $\$  replicate n (f x) snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last take n iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n

#### Higher-order snippets





applyNTimes applies a function, f, to an initial value, x, n times

applyNTimes :: (a -> a) -> a -> Int -> a

snippet1 f x n = last  $\$  replicate n (f x) snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last \$ take n \$ iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n

#### Higher-order snippets





applyNTimes ::  $(a \rightarrow a) \rightarrow a \rightarrow Int \rightarrow a$ 

Experts thought this snippet1 f x n = last  $\$  replicate n (f x) snippet was ill-typed! snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last \$ take n \$ iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n

#### Higher-order snippets

#### applyNTimes applies a function, f, to an initial value, x, n times







applyNTimes :: (a -> a) -> a -> Int -> a

Experts thought this snippet1 f x n = last  $\$  replicate n (f x) snippet was ill-typed! snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last take n iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n

Quickly got too confusing for everyone

#### Higher-order snippets

#### applyNTimes applies a function, f, to an initial value, x, n times

Had to rely on tests!





Create test knowing the expected output

#### Test Usage

#### Create test knowing the expected output



Run on remaining snippets

#### Test Usage

#### Create test knowing the expected output



Run on remaining snippets

#### Test Usage

Eliminate incorrect snippets

#### Create test knowing the expected output



Run on remaining snippets

#### Test Usage





#### Create test knowing the expected output



Run on remaining snippets

#### Test Usage

Accept snippets Yes



Confident in remainder?





#### Create test knowing the expected output



Eliminate Run on Confident in remaining incorrect remainder? snippets snippets No

#### Test Usage

Accept snippets

Yes

#### Update test

12





#### Study Results Snippet Correction applyNTimes applies a function, f, to an initial value, x, n times

applyNTimes ::  $(a \rightarrow a) \rightarrow a \rightarrow Int \rightarrow a$ 

snippet1 f x n = last  $\$  replicate n (f x) snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last take n iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n



#### Study Results Snippet Correction applyNTimes applies a function, f, to an initial value, x, n times

applyNTimes ::  $(a \rightarrow a) \rightarrow a \rightarrow Int \rightarrow a$ 

snippet1 f x n = last  $\$  replicate n (f x) snippet2 f x n = foldr id x (replicate x n) snippet3 f x n = last take n iterate f x snippet4 f x n = f (repeat x !! n) snippet5 f x n = scanl (const f) x (repeat x) !! n

Suggested a fix, but would not have written it this way

Off-by-one, should be (n -1)







### Observational Study



### Study Applications

### Future Studies

```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the matrix.
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if i == j:
                print(matrix[i][j], end=" ")
```



11 11 11 5 You are given a matrix of dimension NxM. 6 You need to print the diagonal elements of the matrix. 7 8 9 Example: [[1, 2, 3], 10 Cono-11 [4, 5, 6], 12 [7, 8, 9]] 13 Output: 1, 2, 4, 3, 5, 7, 6, 8, 9 14 11 11 11 15 def print\_diagonal(matrix): 16 for i in range(len(matrix)): for j in range(len(matrix[i])): if i == j:print(matrix[i][j], end=" ")



```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the matrix.
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
             if i == j:
                 print(matrix[i][j], end=" ")
```







```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the matrix.
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
             if i == j:
                 print(matrix[i][j], end=" ")
```







```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the matrix.
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
             if i == j:
                 print(matrix[i][j], end=" ")
```





11 11 11 5 You are given a matrix of dimension NxM. 6 You need to print the diagonal elements of the matrix. 7 8 9 Example: Copilot doesn't [[1, 2, 3], 10 understand the [4, 5, 6], 11 given example 12 [7, 8, 9]] 13 Output: 1, 2, 4, 3, 5, 7, 6, 8, 9 14 11 11 11 15 def print\_diagonal(matrix): 16 for i in range(len(matrix)): for j in range(len(matrix[i])): if i == j:print(matrix[i][j], end=" ")

17





```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the matrix.
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
             if i == j:
                 print(matrix[i][j], end=" ")
```





```
11 11 11
You are given a matrix of dimension NxM.
You need to print the diagonal elements of the mat
Example:
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
11 11 11
def print_diagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
             if i == j:
                 print(matrix[i][j], end=" ")
```







#### Filter snippets by test

5	ппп		
6	You are given a matrix of dimension NxM.		
7	You need to print the diagonal elements of the matrix.		
8			
9	Example:		
10	[[1, 2, 3],		
11	[4, 5, 6],		1
12	[7, 8, 9]]	matrix	resul
13			
14	Output: 1, 2, 4, 3, 5, 7, 6, 8, 9	[[1,2,3]	1 5
15			, <sup>7</sup>
16	def print_diagonal(matrix):		
	for 1 in range(len(matrix)):	[[1],[2],[3]]	1
	for j in range(len(matrix[i])):		
	LT L == J	[[1,2],[3]]	erro
17	princ(matrix[i]), end = )		
т(			-





#### Filter snippets by test

5	ппп		
6	You are given a matrix of dimension NxM.		
7	You need to print the diagonal elements of the matrix.		
8			
9	Example:		
10	[[1, 2, 3],		
11	[4, 5, 6],		1
12	[7, 8, 9]]	matrix	resul
13			
14	Output: 1, 2, 4, 3, 5, 7, 6, 8, 9	[[1,2,3]	1 5
15			, <sup>7</sup>
16	def print_diagonal(matrix):		
	for 1 in range(len(matrix)):	[[1],[2],[3]]	1
	for j in range(len(matrix[i])):		
	LT L == J	[[1,2],[3]]	erro
17	princ(matrix[i]), end = )		
т(			-

Projection Boxes [Lerner '20]



### Users can get inspired by wrong solutions





#### Filter snippets by test

5	ппп		
6	You are given a matrix of dimension NxM.		
7	You need to print the diagonal elements of the matrix.		
8			
9	Example:		
10	[[1, 2, 3],		
11	[4, 5, 6],		1
12	[7, 8, 9]]	matrix	resul
13			
14	Output: 1, 2, 4, 3, 5, 7, 6, 8, 9	[[1,2,3]	1 5
15			, <sup>7</sup>
16	def print_diagonal(matrix):		
	for 1 in range(len(matrix)):	[[1],[2],[3]]	1
	for j in range(len(matrix[i])):		
	LT L == J	[[1,2],[3]]	erro
17	princ(matrix[i]), end = )		
т(			-

Projection Boxes [Lerner '20]



# wrong solutions







### Observational Study



### Study Applications

### **Future Studies**

```
5
      11 11 11
 6
     You are given a matrix of dimension NxM.
 7
      You need to print the diagonal elements of the matrix.
 8
9
      Example:
     [[1, 2, 3],
10
    [4, 5, 6],
11
      [7, 8, 9]]
12
13
     Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
14
      11 11 11
15
     def print_diagonal(matrix):
16
          for i in range(len(matrix)):
```



```
5
      11 11 11
 6
      You are given a matrix of dimension NxM.
 7
      You need to print the diagonal elements of the matrix.
 8
 9
      Example:
      [[1, 2, 3],
10
11
       [4, 5, 6],
      [7, 8, 9]]
12
13
      Output: 1, 2, 4, 3, 5, 7, 6, 8, 9
14
15
      11 11 11
      def print_diagonal(matrix):
16
          for i in range(len(matrix)):
```

```
.....
```

```
Remove the majority element from an array.
Example input: [3, 3, 4, 2, 4, 4, 2, 4, 4]
Example output: [3, 3, 2, 2]
.....
```

#### def remove\_majority(arr):

```
Remove the majority element from an array.
11 11 11
# Create a dictionary to store the frequency of each element.
freq = \{\}
for i in arr:
    if i in freq:
       freq[i] += 1
    else:
        freq[i] = 1
# Find the majority element.
majority = max(freq, key=freq.get)
# Find the frequency of the majority element.
freq_majority = freq[majority]
# Find the length of the array.
length = len(arr)
# Find the number of majority elements.
num_majority = length // freq_majority
# Find the number of elements that are not the majority element.
num_not_majority = length - num_majority
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = 0
for i in arr:
   if i != majority:
        if i in freq:
            freq[i] -= 1
        else:
            freq[i] = -1
        if freq[i] == 0:
            del freq[i]
        else:
            num_not_majority_majority += 1
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = num_not_majority - num_not_majority_majority
# Find the number of majority elements that are the majority element.
```





```
.....
```

```
Remove the majority element from an array.
Example input: [3, 3, 4, 2, 4, 4, 2, 4, 4]
Example output: [3, 3, 2, 2]
```

#### def remove\_majority(arr):

```
Remove the majority element from an array.
11 11 11
# Create a dictionary to store the frequency of each element.
freq = \{\}
for i in arr:
    if i in freq:
       freq[i] += 1
    else:
        freq[i] = 1
# Find the majority element.
majority = max(freq, key=freq.get)
# Find the frequency of the majority element.
freq_majority = freq[majority]
# Find the length of the array.
length = len(arr)
# Find the number of majority elements.
num_majority = length // freq_majority
# Find the number of elements that are not the majority element.
num_not_majority = length - num_majority
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = 0
for i in arr:
   if i != majority:
        if i in freq:
            freq[i] -= 1
        else:
            freq[i] = -1
        if freq[i] == 0:
            del freq[i]
        else:
            num_not_majority_majority += 1
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = num_not_majority - num_not_majority_majority
# Find the number of majority elements that are the majority element.
```





#### .....

Remove the majority element from an array. Example input: [3, 3, 4, 2, 4, 4, 2, 4, 4] Example output: [3, 3, 2, 2]

#### def remove\_majority(arr):

```
Remove the majority element from an array.
11 11 11
# Create a dictionary to store the frequency of each element.
freq = \{\}
for i in arr:
    if i in freq:
       freq[i] += 1
    else:
        freq[i] = 1
# Find the majority element.
majority = max(fr)
# Find the frequ
                         All-at-once
freq_majority =
# Find the lengt
length = len(arr
# Find the numbe
num_majority = lenger
# Find the number of elements that are not the majority element.
num_not_majority = length - num_majority
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = 0
for i in arr:
    if i != majority:
        if i in freq:
            freq[i] -= 1
        else:
           freq[i] = -1
        if freq[i] == 0:
           del freq[i]
        else:
            num_not_majority_majority += 1
# Find the number of majority elements that are not the majority element.
num_not_majority_majority = num_not_majority - num_not_majority_majority
# Find the number of majority elements that are the majority element.
```



### Which style helps users select the right program faster?



......

Domaria the majority alamont

of each element.

for i in arr: *if i in freq:* freq[i] += 1 else: freq[i] = 1# Find the majority element. majority = max(fr# Find the frequ All-at-once freq\_majority = # Find the lengt lenath = len(arr # Find the numbe num\_majority = lenger # Find the number of elements that are not the majority element. num\_not\_majority = length - num\_majority # Find the number of majority elements that are not the majority element. num\_not\_majority\_majority = 0 for i in arr: *if i != majority:* if i in freq: freq[i] -= 1 else: freq[i] = -1*if freq[i] == 0:* del freq[i] else: num\_not\_majority\_majority += 1 # Find the number of majority elements that are not the majority element.



### Natural Language Synthesis

Simple Input Hard to predict

Which style helps users select the right program faster?

VS

### Enumerative Synthesis

### Understands Examples Simple mental model

# Food for thought, while you get food

- Will you be able to tell when someone uses a synthesizer?
- enumerative, etc.)
- Would you judge code differently depending on provenance?
- Should synthesized code be labeled?



- Does your confidence in a synthesizer differ by system (neural,

### Conclusion

### Ran observational study for program recognition on FP users



#### Code Exploration

Snippet Correction





### Ran observational study for program recognition on FP users



Process of Elimination

Snippet Correction

Thank you all! Michael B. James

#### Code Exploration

