1 Motivation

Memoizing objects in Python is quite easy. However, the following program has a semantic error.

```python
from functools import cache

@cache
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p0 = Person("Michael", 31)
p1 = Person("Michael", 31) # p1 == p0 due to memoization
p1.name = "Caio"
p1.age = 22

print(p0.name, p0.age)
print(p1.name, p1.age)
```

As *p0* and *p1* point to the same object in memory, the program produces the unsatisfactory output:

```
Caio 22
Caio 22
```

2 Description of the Alias Set Algorithm

Consider the following Hush program, where `Point` and `Triangle` are memoized.

```
let p0 = Point(1, 1)
let p1 = Point(1, 1) // p1 -> p0 due to memoization
let p2 = p1

let t0 = Triangle(p0, p1, p2)
```

Analogous to the Python example, this program can be invalidaded if we change *p0*, *p1* or *p2*, as they all point to the same object in memory. However, by using an *alias set* we can keep track of references and instatiate new objects when we see fit.

3 Current State of Memoization in Hush

- We have memoization and objects as closures → We can memoize immutable objects.
- Raise exception when mutating memo objects. [DONE]

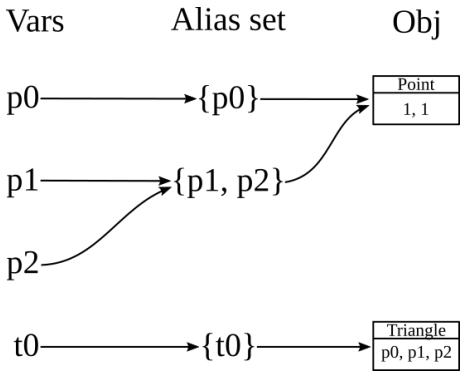

Figure 1: Memory layout of Program 2

- Memoization of arbitrary objects. [WIP]