Class 15: Recursive Data Types

Schedule

You should read MCS Chapter 7 this week.

Problem Set 6 is due **20 October (Friday) at 6:29pm**.

Proving Correctness

```
def slow_power(a, b):
   y = 1
   z = b
   while z > 0:
      y = y * a
      z = z - 1
   return y
```

We model the Python program with a state machine:

$S ::= \mathbb{N} \times \mathbb{N}$
$G ::= \{(y, z) \to (y \cdot a, z - 1) \,|\, \forall y, z \in \mathbb{N}^+\}$
$q_0 ::= (1, b)$

It is important to remember this is a *model*. It does not capture many important aspects of execution of a real Python program. In particular, it only models inputs in $\mathbb{N}$, when the actual inputs could be other types in PYthon. It also assume all math opderations work mathematically, not Pythonically.

To prove partial correctness, we show $P(q = (y, z)) := y = a^{b-z}$ is a *preserved invariant*. Then, we show that it holds in state q_0. Finally, we show that in all final states, $y = a^b$.

Invariant is Preserved: We need to show that $\forall q \in S. \forall t \in S. (q, t) \in G \implies P(q) \implies P(t)$.

1. $q = (y, z)$. $P(q = (y, z))$: $y = a^{b-z}$.
2. If there is an edge from q to t, that means $t = (y \cdot a, z - 1)$ and $z \geq 1$ since this is the only edge from q in G.
3. We show $P(t = (y \cdot a, z - 1))$ holds by multiplying both sides of $P(q)$ by a:

$$ya = (a^{b-z}) \cdot a = a^{b-z+1} = a^{b-(z-1)}.$$

Invariant holds in q_0: $q_0 = (1, b)$. So, we need to show $P(q_0 = (1, b))$: $1 = a^{b-b}$. This holds since $a^0 = 1$.

Final states: All states where $z \geq 1$ have an outgoing edge, but no states where $z = 0$ do. So, the final states are all of the form $(\alpha, 0)$. If a final state is reachable from q_0, the invariant must hold since we proved it is preserve. Hence, in the final state $(\alpha, 0)$ we know $\alpha = a^b$.

This proves *partial correctness*: if the program terminates, it terminates in a state where the property ($y = a^b$ is satisfied). To prove *total correctness* we also need to know the execution *eventually* reaches a final state.

We prove this by showing that from any initial state $q_0 = (1, b)$, the machine will reach a final state $q_f = (y, 0)$ in b steps. The proof in class used the Well Ordering Principle. You could also prove this using regular Induction.

Pairs

What is the difference between scalar data and compound data structures?

Definition. A *Pair* is a datatype that supports these three operations:

$$\begin{aligned}&\mathit{make\_pair} : \mathit{Object} \times \mathit{Object} \to \mathit{Pair}\\&\mathit{pair\_first} : \mathit{Pair} \to \mathit{Object}\\&\mathit{pair\_last} : \mathit{Pair} \to \mathit{Object}\end{aligned}$$

where, for any objects a and b, $\mathit{pair\_first}(\mathit{make\_pair}(a, b)) = a$ and $\mathit{pair\_last}(\mathit{make\_pair}(a, b)) = b$.

```
def make_pair(a, b):
    def selector(which):
        if which:
            return a
        else:
            return b
    return selector

def pair_first(p):
    return p(True)

def pair_last(p):
    return p(False)
```

Lists

Definition (1). A *List* is either (1) a *Pair* where the second part of the pair is a *List*, or (2) the empty list.

Definition (2). A *List* is a ordered sequence of objects.

