FlowScript Design Document

By Kenny Zhang (ID: 48618419)

Language Elements and Related Features

Variables:

- Syntax:
    - name = value
    - Example: foo = 5
- Semantics:
    - Variables will be declared using a variable name, and a value. The datatype of variables will be implicitly defined by the value given to it. This will be implemented in a similar way to how Python does it.
    - These variables are intended to store primitive data types that can be manipulated using scripts. This is essential for basic object oriented programming, which flowscript will be.

Data types:

- Syntax:
    - name = (datatype) value
    - Example: num = (int) true
- Semantics:
    - FlowScript (FS) will include the basic primitive data types that most other languages have, i.e. boolean, int, float, string, array, etc. While variables cannot be explicitly defined as one data type, values assigned to variables can be casted to a data type, ensuring that that variable will be that datatype.
    - FS will also include more graph specific data types like nodes, edges, and graphs which will be in their own classes/objects. I will go into further detail when we discuss classes later.

Operators:

- Syntax:
    - directed edge = node -> node
    - undirected edge = node - node
    - child member = parent.child
    - array member = array[index]
- Semantics:
    - FS will have two operators for edges. One, represented by an arrow, will represent a directed edge. Another, represented by a dash, will represent an undirected edge. These two operators can define a value that can be assigned to an edge.
    - FS will also have bracket and dot operators for convenience. The brackets will be used for easy array member access and the dot for easy child member access.

Control Structures:

- Syntax:

    - normal edge = node -> node
    - if conditional edge = node (condition) -> node
    - if else conditional edge = node (condition) -> nodeIfTrue : nodeIfFalse
    - loop = node a -> node a
    - event = [expression/operation/condition]
- Semantics:
    - FS will have the standard control structures. Conditionals can be applied to edges and nodes. These include standard if statements and if and else statements.
    - Loops are also available by pointing a node at itself. Conditionals can be applied to these too.
    - Events are nodes that will do an operation/expression when it is reached.

Functions:

- Syntax:
    - returnType name(parameters) {}
- Semantics:
    - Functions will *function* (hah) the same as they do in most programs. Function calls can be attached to nodes by using event nodes.

Objects:

- Syntax:
    - Node
        - string name
        - string[] destinations
        - job job
        - function event (optional)
    - Edge
        - boolean directed
        - string start
        - job startJob
        - string end
        - job endJob
        - function condition (optional)
    - Graph
        - boolean directed
        - node[] nodes
        - edge[] edges
- Semantics:
    - Objects include nodes, edges, and graphs. Nodes can hold values and also hold an array of places they can go to from that node. Nodes can also hold jobs that will use the values of the node or incoming nodes to produce outputs. Edges can hold two values representing its start and end, as well as jobs that are linked to the start and end nodes that illustrate dependencies. Graphs hold an array of

nodes and edges that make up what it is. Edges and graphs can be defined as directed or not. Nodes and edges can have events attached to them.

FlowScript Examples

Process and execution conditions

```
node a = 5;
node b = 6;
node c = 7;
c (5 < 6) a : b;
```

Define specific process, by name, that will be executed

```
node b;
process a;
b.event = process;
```

Define dependency order for execution of processes

```
edge a;
a.startJob = compileJob;
a.endJob = parseJob;
```

Enable ability to show/allow parallel execution of process

```
node a;
node b;
node c;
node d;
a.destinations = [b, c, d];
```