Pathy, a mapping and route finding language

Pathy is a logical language with elements of an imperative language.

Variable types and classes

Oobject orientation exists in that you create instances from a set of predefined classes and assign instances properties unique to that instance, but cannot create your own classes. Thus there is no inheritance

The aforementioned classes are the main variable types and they come in five types: Nodes, Links, Junctions, Entities and Actions.

For calling them variables, they're not actually very mutable once declarations have been completed.

- Once declared, actions are non-mutable, but that's because they have no properties to alter.
- Nodes can have Actions assigned to them.
- Links can have their weight and directional assignment changed (one-way, two-way, blocked).
- Junctions can be made to only allow certain exits active at a time and switch between them, like railway points.
- Entities can have their energy level modified (or one added) and their position can be changed.

Actions are unique in how they are declared and used. Actions are declared with a name and then copies of it are assigned to nodes. This is as close to object orientation as Pathy gets, but it's more re-use of a name than behaviour.

Pathy accepts Strings and Numbers as parameters, but cannot store them in their own variables. This is due to the fact that it's not expected that large sweeping changes would be required in this language. Anything that would require that functionality should be within the scope of a text editor's standard find and replace functionality.

"Assignment" is not a concept past declaring an instance exists with specific properties and has a name in the world. Once set it cannot be overwritten with anything else. In order to use that name for something else, the instance a name refers to must be deleted and then that name declared to be something else.

For a logical language there isn't really much in the way of operators. Most 'work' is done in query functions.

Script

A Pathy script has two sections: the declarations and the queries. In the declarations you create the "world" in that you create nodes and junctions, add links between them, create actions and assign them to nodes, then create entities that start at a node. The queries are then used to ask questions of the world and manipulate it to a certain extent.

The script MUST adhere to this form:

[Declaration Statements]

DecEnd;

[Query Statements]

"DecEnd" signifies the end of declarations and the start of queries. Any queries that occur before DecEnd and any declarations that occur after DecEnd will cause the Pathy interpreter to throw an error.

Each statement, including DecEnd, ends with a semicolon. Line breaks are meaningless and can actually be omitted, making for a nicer interaction with other languages and programs interacting with the Pathy interpreter.

Identifiers can use any alphanumeric character, but must start with a letter. Once an Identifier is set it can be used to associate that instance with another instance in accordance with the properties of the types of instance.

Example: When declaring a link, you must associate it with any two nodes or junctions to form the endpoints. This is done in the Link's constructor by passing in the chosen nodes and/or junctions as parameters. This is mandatory for the link to exist.

Example 2: After declaring an Action, you can associate it with one or more nodes to show that it is available at those nodes.

There is no concept of user-defined functions, and due to the lack of user-defined classes, as a result scope doesn't exist as a concept.

Declarations and Queries

When declaring your world you are only allowed to use variable declarations and declaration functions to create the world.

If a mistake is made you can delete an instance from the world assuming it won't leave any Links with only one end attached or an Entity having the node it's at disappearing from under it. Any actions on a removed node are implicitly unassigned from that node prior to removal.

Querying is simply asking the world what is there and if something can occur. This can be something as simple as asking what other nodes a node is liked to, to something like asking if an entity can make it from A to B with the energy it has.

Example: I declare 5 nodes (a, b, c, d and e), and 3 actions (x, y and z). I associate the actions amongst the nodes so that x appears on a, b, and c; y appears on c, d and e; and Z appears on a, c and e. I can then ask Pathy for a print out of questions such as: Where can I do z? What can I do at a? What actions do c and e share?

Running

The idea behind Pathy is that another program can use it to query a world model without having to worry about importing libraries.

By keeping a copy of the declarations in the program, all that need to happen is that before it passes the script off to the interpreter, the queries are added so that the program may get out of it exactly what it needs at the time.

All return is text-based so that it may be easily output to a file via the command line, or for piping into another process.

Functionality not implemented

The idea of Pathy was that it was supposed to be able to do pathfinding. However that functionality has not been built in yet, and will likely be included in a future version of Pathy.