Twined Language Specification

Lucas Weissman Zach Sturdevant

April 28, 2024

Introduction

2+ paragraphs. What problem does your language solve? What makes you think that this problem should have its own programming language?
”Twined” represents a transformative approach in how we manage and interact with textual information. By converting unstructured text data from diverse note-taking formats into structured, navigable knowledge graphs, ”Twined” provides a unique visual perspective that enhances comprehension and analysis.

Rules: Graph Rule: x, y, x, z $\rightarrow$ x, z, x, w, y, w, z, w

Hypergraph Rule: x, y, z $\rightarrow$ x, u, v, z, v, w, y, w, u

graph: Sunlight,Plant Growth,Water,Plant Growth,Soil Nutrients,Plant Growth

hypergraph: Sunlight,Water,Soil Nutrients,Plant Growth,Temperature,Humidity,Water,Plant Growth

Brainstorming Section:
“Twined” intertwines user’s information by managing and interacting with textual information. It converts unstructured text data from various note-taking formats into structured, navigable knowledge graphs. This will allow users to quickly visualize connections and gain insights that might be missed in traditional textual data formats.

A hypergraph is defined as a collection of nodes and hyperedges, where each hyperedge has the capability to connect three or more nodes, unlike traditional graphs where edges connect only two nodes. In terms of mathematical representation, edges in standard graphs are noted as pairs of numbers within curly brackets, such as 1, 2. Conversely, in hypergraphs, hyperedges that connect multiple nodes are denoted by three or more numbers within curly brackets, for example, 1, 2, 3. Visually, in graphs, edges are depicted as connections between white dots (nodes) using white arrows (edges). Hypergraphs extend this visualization by linking three or more white dots (nodes) with multiple arrows and a transparent white web, highlighting the complex relationships between nodes.

PS. We might adjust the intro to fit within the context of our current deliverables, or we could state the end goal briefily and current stage we are in with the test cases we have built. - Lucas

Design Principles

1+ paragraphs. Languages can solve problems in many ways. What are the aesthetic or technical ideas that guide its design?

The goal of this language is to produce useful output with minumal input from the user in an easy to understand way. In the current form the user has to design and build the graph themselves, but as continue we plan to add features that will allow the user to upload a text and say in english what they hope to get out of it and the language will handle the rest. Aesthetic we are attempting to have a simple start that move into a powerful GUI to allow anyone to use the language.
From a technical stand point we want to utilize generative AI to imporve the ability of people to access knowledge in a way that is efficient and allows for multiple kinds of inputs and customization of the outputs. Basically use technoligy to make peoples lives better and make knowledge more readily accessable to all.

Brainstorming Section:

Examples

1. (a) Program 1 input

```
{Nancy, (Fred, Sally, Sarah,)}
{Fred, (Nancy, Sally, Sarah,)}
{Sally, (Jeff, Jonny, Timmy,)}
{Jeff, (Sally, Jonny, Timmy,)}
```

(b) Program 1 output This graph represents the relationships between family individuals. Each node represents

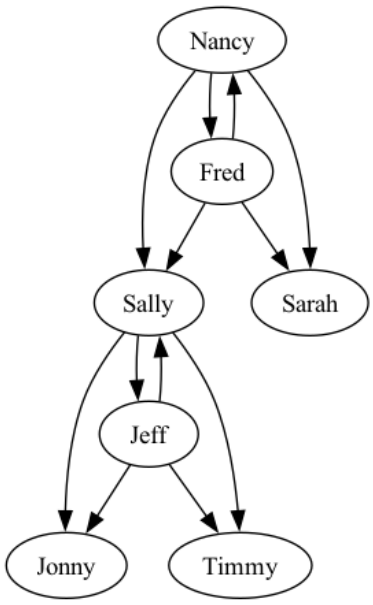


Figure 1: Family Relationship

a person, and each edge represents a direct relationship between two individuals. For example, ”Nancy” has relationships with ”Sally,” ”Fred,” and ”Sarah,” as depicted by the edges connecting their respective nodes.

2. (a) Program 2 input

```
{Sunlightlove, (PlantGrowth,)}
{Water, (PlantGrowth,)}
{SoilNutrients, (PlantGrowth,)}
```

(b) Program 2 output The graph represents factors affecting plant growth. Each node represents a factor such

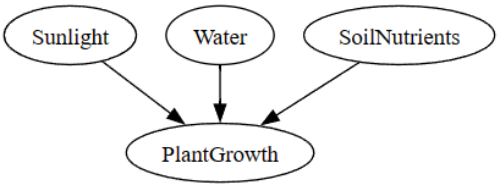


Figure 2: Plant Growth Relationship

as ”Sunlight,” ”Water,” and ”Soil Nutrients,” and each edge represents the influence of these factors on ”Plant Growth,” as depicted by the edges connecting their respective nodes.

3. (a) Program 3 input

```
{European Fascism, (Germany, Italy, Spain,)}
{Germany, (Adolf Hitler,)}
{Italy, (Benito Mussolini,)}
{Spain,(Francisco Franco,)}
```

(b) Program 3 output

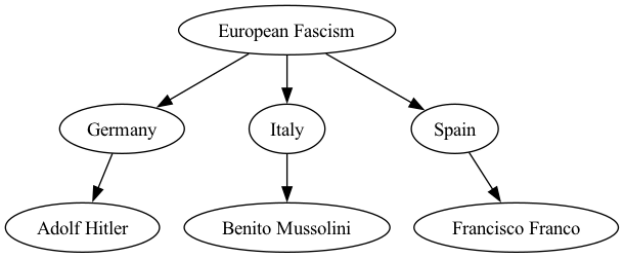


Figure 3: Famious Eureopean Fascists

This graph shows connections between European Fascism and leaders of different Fascist countries. Each node represents a topic and how those topics are connected in a top down manner.

Brainstorming Section

I have added above an example of image insertion from one of our test cases, we could replace it with a few more/new ones. - Lucas

Language Concepts

(What are the core concepts a user needs to understand in order to write programs? Think in terms of both “primitives” and “combining forms.” What are the key ideas and how are they combined?)
Currently the user needs to understand how graph nodes connect. However when comepleted the user shouldn't need an understanding of primatives, just input the information they want to explore and utilize the GUI to explore the information.
The language concepts behind ”Twined” incorporate elements from graph theory and data visualization to support the creation of knowledge graphs and hypergraphs. These concepts are integral in defining how textual data is parsed, how entities within the text are identified as nodes, and how relationships (edges) are established based on the context and content of the data. This allows ”Twined” to dynamically interpret and display information in a way that highlights both hierarchical and associative relationships.

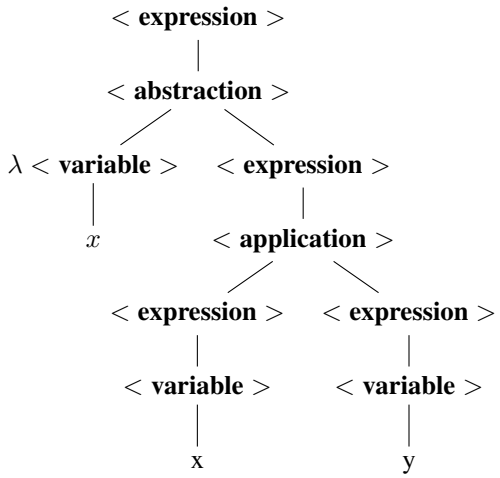
Delete this TODO and replace with 1+ paragraphs.

Brainstorming Section
Terms to be used/contextualized: Graph Theory Data Visualization Knowledge Graphs Hypergraphs Parsing Nodes Edges Hierarchical Relationships Associative relationships

Formal Syntax

< **expression** >	::=	< **Node** >
	|	< **edgeList** >
	|	< **nodeName** >
	|	< **listOfNodes** >
< **Node** >	::=	{ < **String** >, < **nodeName** > }
< **nodeName** >	::=	< **String** >
< **edgeList** >	::=	(< **nodeName** > +)
< **listOfNodes** >	::=	< **Node** > +

Delete this TODO and replace with BNF.



Brainstorming Section

Semantics

1. Primitive values:
   (a) NodeName: A series of characters digits or spaces that represent the name of a node. stored in a string.
   (b) EdgeList: A series of NodeNames, or 0 stored in the form (NodeName, NodeName,) that contains the NodeNames of nodes connected to the node, stored in a list.
2. Combining forms:
   (a) Node: A tuple of the form NodeName, (EdgeList) that represents a full node, which is the name of the node and its EdgeList. In the future the node name will be used to access a dictionary that contains the information associted with that node
   (b) NodeList: A series of nodes in the form Node Node separated by whitespace including newlines these are used to represent all nodes in a graph
3. Evalutation:
   (a) The programs in our language read inputs given in a txt file, it parses the text file and separating the series of nodes and their edges.
   (b) When the program is evaluated it produces a graph using the nodes and their edgelist on the screen

Brainstorming Section