Yet Another Graph Library (YAGL)

Brief Overview

- Simple header-only C++17 library.
- Nodes make use of special variant nodes to to take on different types of data.
  - A further specialization is defined for DGGML.
- Nodes are stored in a map for quick lookup.
- Includes a few algorithms such as finding connected components and finding graph isomorphisms.

```
template <typename ... Types>
struct VariantData
{
    using node_variant_t = std::variant<Types ...>;
    node_variant_t data;
}
```

Base type for the variant data in YAGL.

```
template <typename ... Types>
struct SpatialData3D : VariantData<Types ...>
{
    double position[3];
}
```

Specialization of the variant data type for DGGML.

```
struct Negative
{
    double velocity[3];
    double unit_vec[3];
};
```

Example of a type.

```
struct Positive
{
    double velocity[3];
    double unit_vec[3];
};
```

Example of another type.

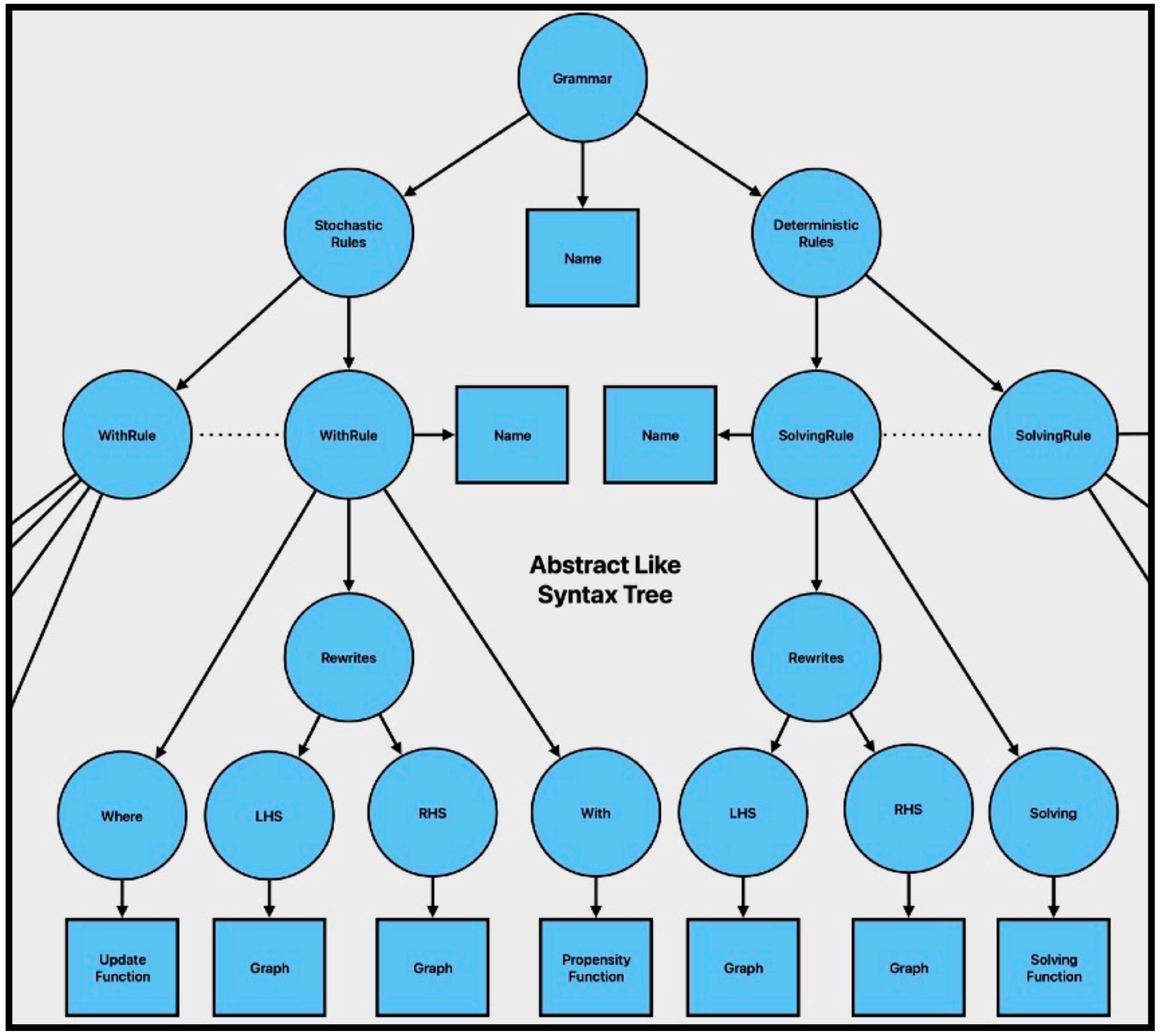
```
using graph_type =
    YAGL::Graph<unsigned int,
    SpatialNode3D<Negative, Positive>>;
```

Graph defining interface.

Grammar Analysis

Abstract Syntax Tree

- All grammars have stochastic and deterministic rules.
- Stochastic rules have:
  - Left-hand side (LHS) and right-hand side (RHS) graphs used in rewriting.
  - **With** clause for the propensity.
  - **Where** for sampling and updating the RHS output parameters.
- Deterministic rules have:
  - A rewrite as well, but that is semantically encoded in the **solving** clause.



A section of an abstract syntax tree (AST) for a DGG.