Library Overview

- Important in understanding how the formalism translates into code.
- We have acronyms:
  - VTK (visualization toolkit)
  - SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers)
  - SIMD JSON (single input, multiple data JavaScript object notation)
  - YAGL (yet another graph library)

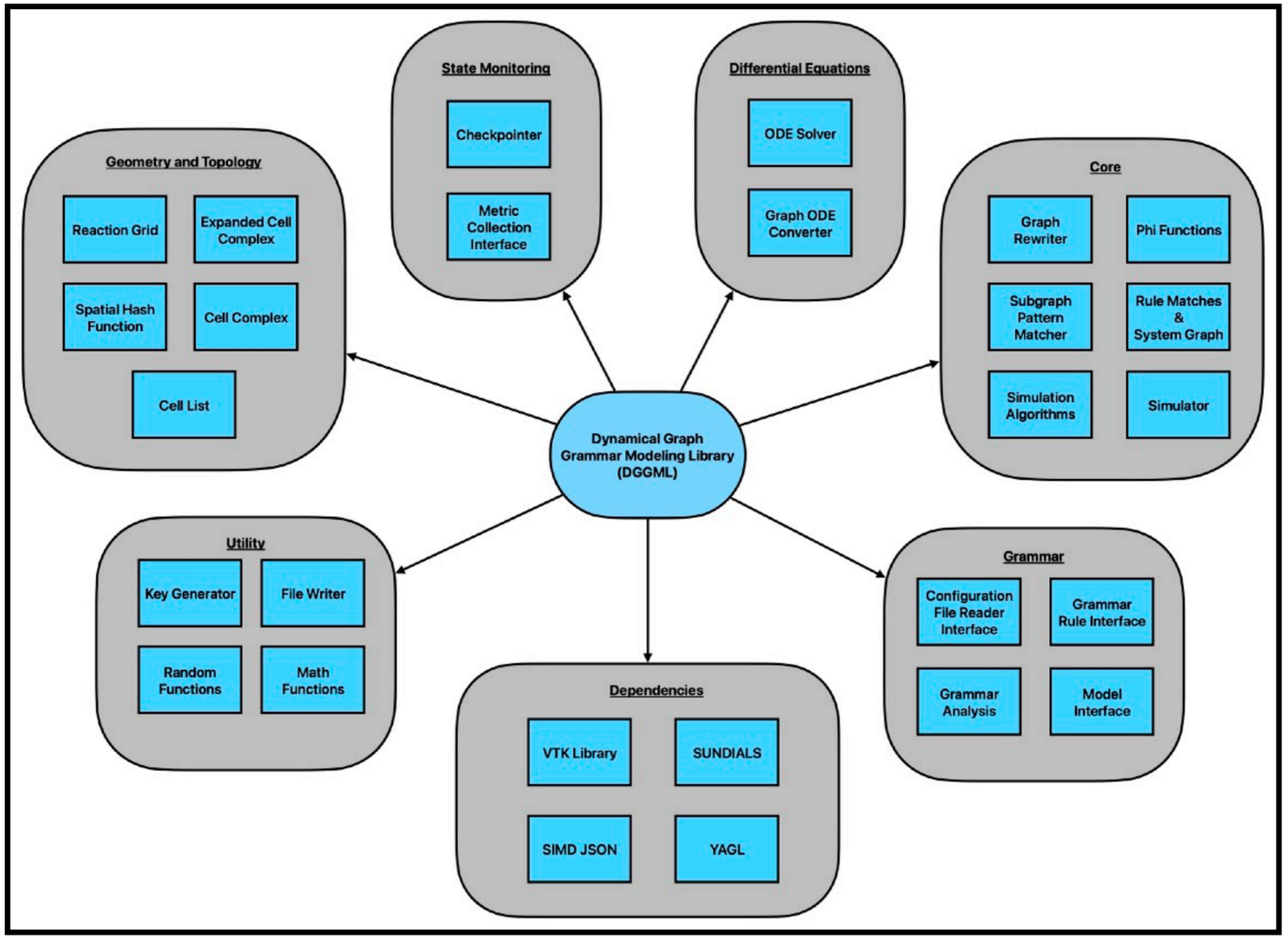

Figure 19: Conceptual overview of the DGGML design, a C++17 library.

Usage Example

Defining a rule in DGGML.

(1) **Stochastic Growth:**

(2) $(\bigcirc_1 \text{——} \bullet_2)\,\langle\!\langle (\boldsymbol{x}_1, \boldsymbol{u}_1), (\boldsymbol{x}_2, \boldsymbol{u}_2) \rangle\!\rangle$

(3) $\longrightarrow (\bigcirc_1 \text{——} \bigcirc_3 \text{——} \bullet_2)\,\langle\!\langle (\boldsymbol{x}_1, \boldsymbol{u}_1), (\boldsymbol{x}_2, \boldsymbol{u}_2), (\boldsymbol{x}_3, \boldsymbol{u}_3) \rangle\!\rangle$

(4) **with** $H(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|; L_{div})$

(5) **where** $\begin{cases} \boldsymbol{x}_3 = \boldsymbol{x}_2 - (\boldsymbol{x}_2 - \boldsymbol{x}_1)/100.0 \\ \boldsymbol{u}_3 = \boldsymbol{u}_1 \end{cases}$

Example of how a stochastic rule written in the DGG form is transformed into C++ code.

```
GraphType lhs_graph;
lhs_graph.addNode({1, {Intermediate{}}});
lhs_graph.addNode({2, {Positive{}}});
lhs_graph.addEdge(1, 2);

GraphType rhs_graph;
rhs_graph.addNode({1, {Intermediate{}}});
rhs_graph.addNode({3, {Intermediate{}}});
rhs_graph.addNode({2, {Positive{}}});
rhs_graph.addEdge(1, 3);
rhs_graph.addEdge(3, 2);


auto propensity = [&](auto& lhs, auto& m)
{
    auto& node1 = lhs.findNode(m[1])->second.getData();
    auto& node2 = lhs.findNode(m[2])->second.getData();
    auto len = calculate_distance(node1.position, node2.position);
    double propensity = heaviside(len, settings.DIV_LENGTH);
    return propensity;
};

auto update = [](auto& lhs, auto& rhs, auto& m1, auto& m2) {
    for(int i = 0; i < 3; i++) // set position
        rhs[m2[3]].position[i] = lhs[m1[2]].position[i]
            - (lhs[m1[2]].position[i] - lhs[m1[1]].position[i])/100.0;
    for(int i = 0; i < 3; i++) // next set the unit vector
        std::get<Intermediate>(rhs[m2[3]].data).unit_vec[i]
         = std::get<Intermediate>(lhs[m1[1]].data).unit_vec[i];
};

using RT = WithRule<GraphType>; // rule type
RT stochatic_growth("with_growth", lhs_graph, rhs_graph, propensity, update);
gamma.addRule(stochatic_growth);
```

2

3

4

5

1