Mesh Abstraction:

$M = <N, E, T>$

$N = [n_1, n_2, \ldots..]$

$n_i = <p, v_n>$

$E = [e_1, e_2, \ldots..]$

$e_i = <n_i, n_j, v_e>$

$T = [t_1, t_2, \ldots..]$

$t_i = <n_i, n_j, n_k, v_e>$

Abstraction Function:

(see code for representation)

|N| = `primal_.size()`
|E| = `primal_.num_edges()`
|T| = `dual_.size()`
$n_i \rightarrow$ `<primal_.node(i).position(), primal_.node(i).value().value_>`
$e_i \rightarrow$ `<primal_.edge(i).node1(), primal_.edge(i).node2(), primal_.edge(i).value().value_>`
$t_i \rightarrow$ `<dual_.node(i).value().nodes_[0], dual_.node(i).value().nodes_[1], dual_.node(i).value().nodes_[2], dual_.node(i).value().value_>`

Representation Invariant:

Let `t_i ← dual_.node(i)`, where `0 <= i < dual_.num_nodes()`
Let `n_i ← primal_.node(i)`, where `0 <= i < primal_.num_nodes()`
Let `e_i ← primal_.edge(i)`, where `0 <= i < primal_.num_edges()`
Let `t_i.n_ ← t_i.value().nodes_`
Let `t_i.e_ ← t_i.value().edges_`
Let `n_i.t_ ← n_i.value().triangles_`
Let `e_i.t_ ← e_i.value().triangles_`
`Where v denotes a vector:`
Let elem ∈ `v ←` ∃`i, 0 <= i < v.size() s.t. v[i] == elem`
`Where a denotes an array of size 3 (e.g. t_i.n_, t_i.e_)`
Let elem ∈ `a ←` ∃`i, 0 <= i < 3 s.t. a[i] == elem`

//all edges belong to at least one triangle and all the edge’s triangles have that edge

∀`e_i →e_i.t_.size() > 0`

//all triangles have valid nodes and edges

```
∀t_i, j, 0<=j<3 → ∃k s.t. t_i.n_[j] == n_k
∀t_i, j, 0<=j<3 → ∃k s.t. t_i.e_[j] == e_k
```

//all nodes and edges have valid triangles

```
∀n_i → ∀t ∈ n_i.t_, t == t_j
∀e_i → ∀t ∈ e_i.t_, t == t_j
```

//a triangle and its nodes/edges map to each other

```
∀n_i, t_i → t_i ∈ n_i.t_ ⇔n_i ∈ t_i.n_
∀e_i, t_i → t_i ∈ e_i.t_ ⇔e_i ∈ t_i.e_
```

//all edges belong to at least one triangle

```
∀e_i → e_i.t_.size() > 0
```

//nodes in a triangle are ordered and unique

```
∀t_i → t_i.n_[0] < t_i.n_[1] < t_i.n_[2]
```

//each edge in a triangle is between two nodes in that triangle

```
∀t_i,0<=j<3 → t_i.e_[j] == <t_i.n_[j], t_i.n_[(j+1)%3]>
```

//triangles are uniquely defined by their nodes

```
∀t_i,t_j, i!=j → ∃k 0<=k<3 s.t. t_i.n_[k] != t_j.n_[k]
```

//triangles that share an edge are neighbors in the dual graph

```
∀e_i → ∀t1, t2 ∈ e_i.t_, t1 != t2, dual_.has_edge(t1, t2)
```