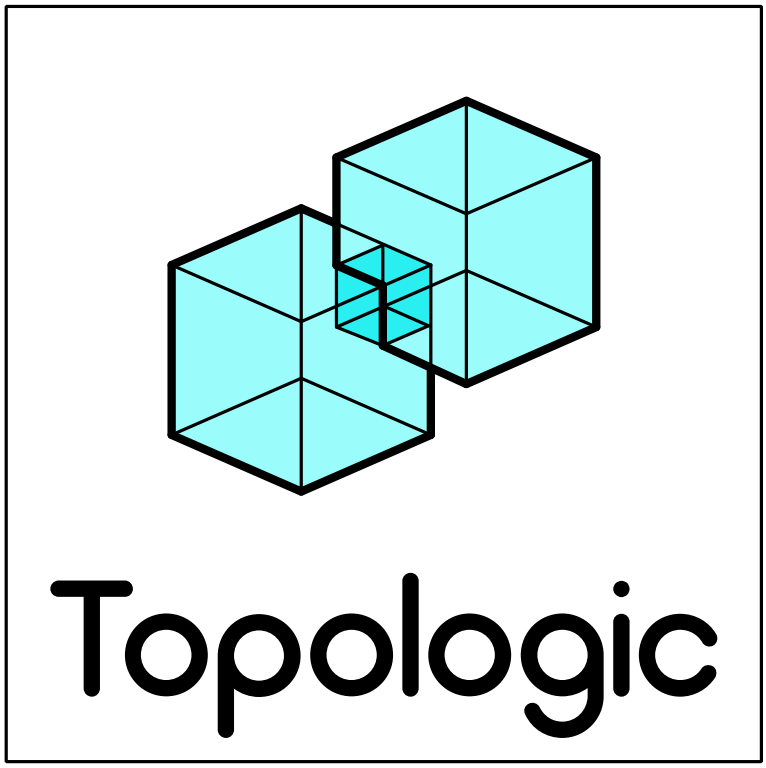


TUTORIAL 01
Getting Started

Topologic v. 0.8.6

June 2020

Katherine Sawey

TABLE OF CONTENTS

Introduction

In this tutorial, we assume that you are familiar with the concepts of visual data flow programming (VDFP) software such as Grasshopper. If you are not, we highly recommend that you first learn how to use Grasshopper before using Topologic.

We highly recommend that you turn off automatic run mode and save your work often as Grasshopper tends to crash frequently.

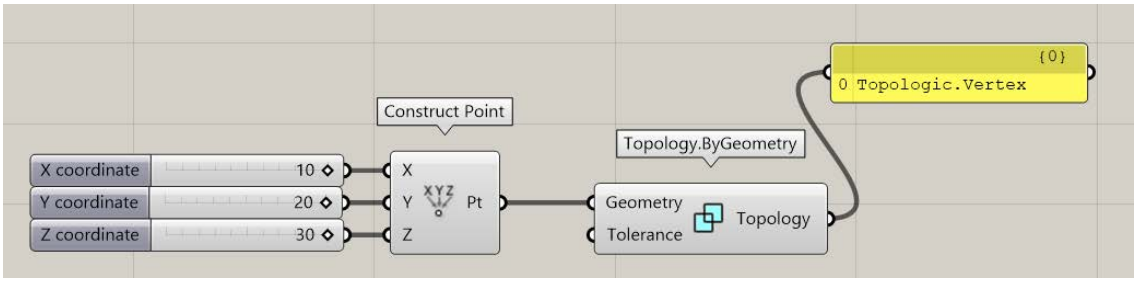
A copy of the Grasshopper definition that you will build in this tutorial is available for download from: https://topologic.app/wp-content/uploads/2019/01/Topologic-03-Tutorial01.pdf

Topologic allows you to build 2D and 3D geometries just like Grasshoppper. The difference is that these objects have a richer data structure that allows us to query them for information that is not usually available using Grasshopper geometries. In particular, Topologic geometries have topological information embedded in them (thus the name). For example, you can ask a face to give you back its edges or its vertices. You may have noticed that we use the term face instead of surface and vertex instead of point. This is consistent with the literature on topology. Below is a mapping between Grasshopper geometries and Topologic topologies:

Grasshopper	**Topologic**
List	Cluster
(Group of interconnected brep)	CellComplex
Brep	Cell
Mesh (or any open geometry)	Shell
Surface	Face
Polycurve	Wire
Curve/Line	Edge
Point	Vertex

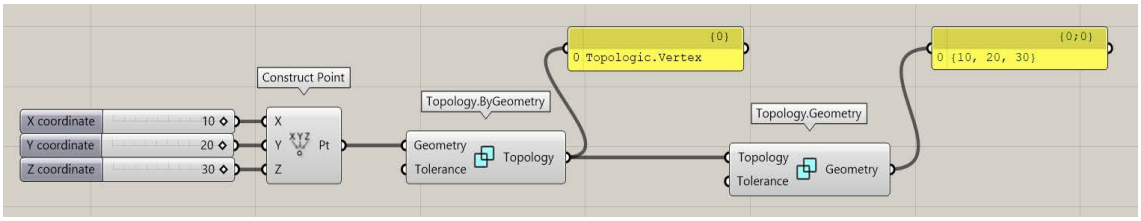
One entry and one exit point

While we recommend using Topologic for building, connecting and analysing topological entities, you are not locked into using Topologic exclusively. You can use regular Grasshopper geometric entities and convert them to Topologic entities by using the various constructor methods. Constructors usually start with the word **By**. For example, in the case below **Topology.ByGeometry** creates a Topologic Vertex from a Grasshopper Point. Topologic is designed to have mainly one entry point (**Topology.ByGeometry**) and one exit point
(**Topology.Geometry**). By design, topology nodes do not render anything to the screen. If you wish to see the result at any point in your workflow, you can drag and drop a **Topology.Geometry** and connect it to your last topology output.



The tolerance value is set by default to 0.001. This should automatically account for any tolerance issues in Grasshopper. However, if a conversion fails, try inputting a larger number.

To get the Grasshopper point back from Topology, drag and drop a **Topology.Geometry** node.

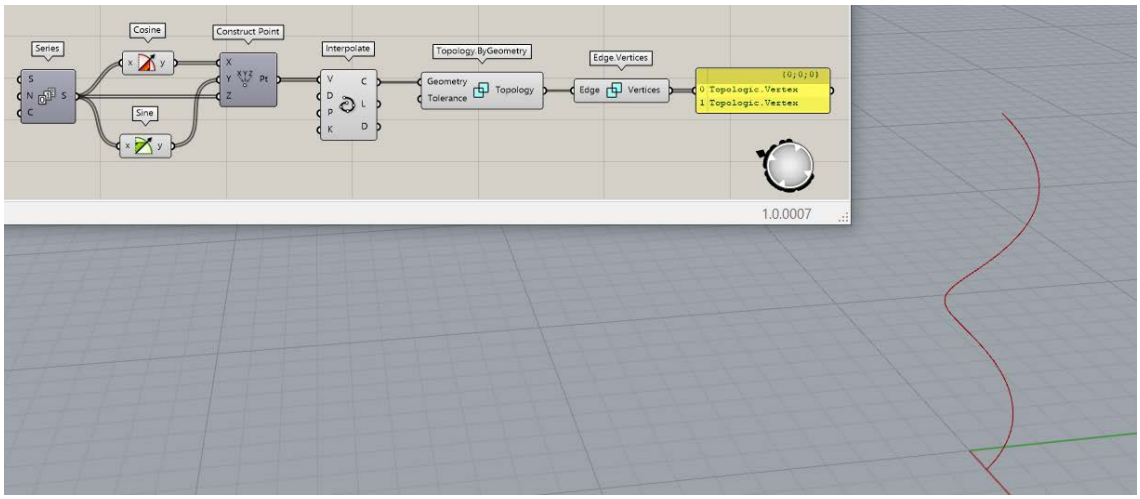


a list within a list

You may be wondering why the output from Topologic is a nested list. This is a side-effect of the fact that in Topologic, you can add one topology to the contents of another topology. This way, you can create nested hierarchies. When Topologic sends the topologies back to Grasshopper as geometries, it preserves their content hierarchy. In this case, there is only one topology and thus the hierarchy appears unnecessary. We did not wish to create exceptions to the mechanism by which Topology converts its topologies to geometries.

Geometry vs. Topology

It is important to note that Topologic concerns itself more with Topology rather than Geometry. However, every topology in Topologic has an associated geometry. Consider the following example: Build a helix in Grasshopper and convert it to Topologic. To Topologic, a helix, or any continuous curve for that matter, is nothing more than an edge with a start vertex and an end vertex. Topologic stores the geometry along with the topology so we can retrieve it later.



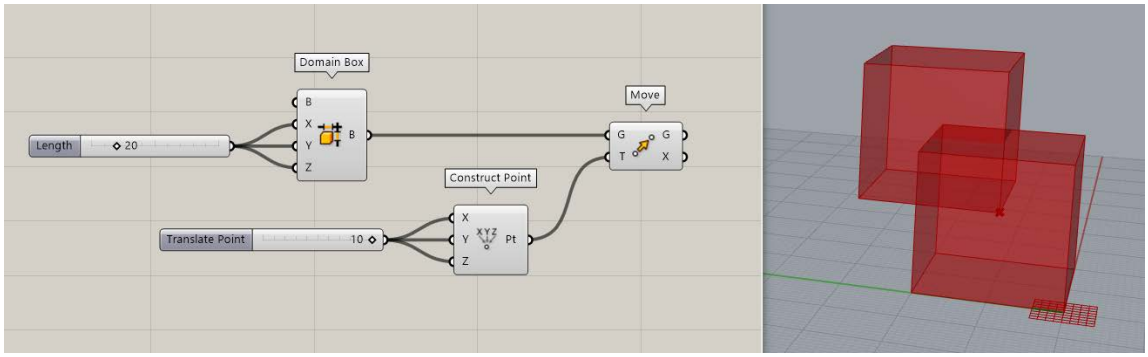
Boolean Operations

Boolean operations can be divided into two categories: regular and irregular. Regular Boolean operations include: union, difference, intersection, and XOR (exclusive OR). Irregular Boolean operations include: merge, impose, slice, imprint, divide, and join. By the time v 1.0 of Topologic is released, it will be able to conduct all these operations.

Building two intersecting cuboids

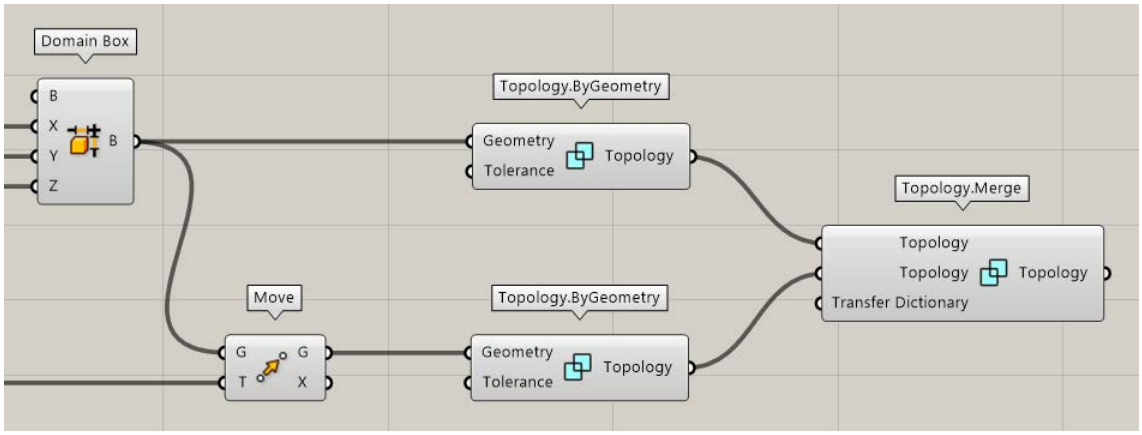
To test the Boolean set operations, we will first build two simple intersecting cuboids in Grasshopper, convert them into Topologic Cells, apply different Topologic Boolean operations on them, and examine the result both visually and analytically. Since most other 3D modelling software can support regular Boolean operations, here we will demo some irregular operations. Once you are comfortable with Topologic, try out the other Boolean operations on your own.

Start Grasshopper and build two cubes as you see in the figure below.

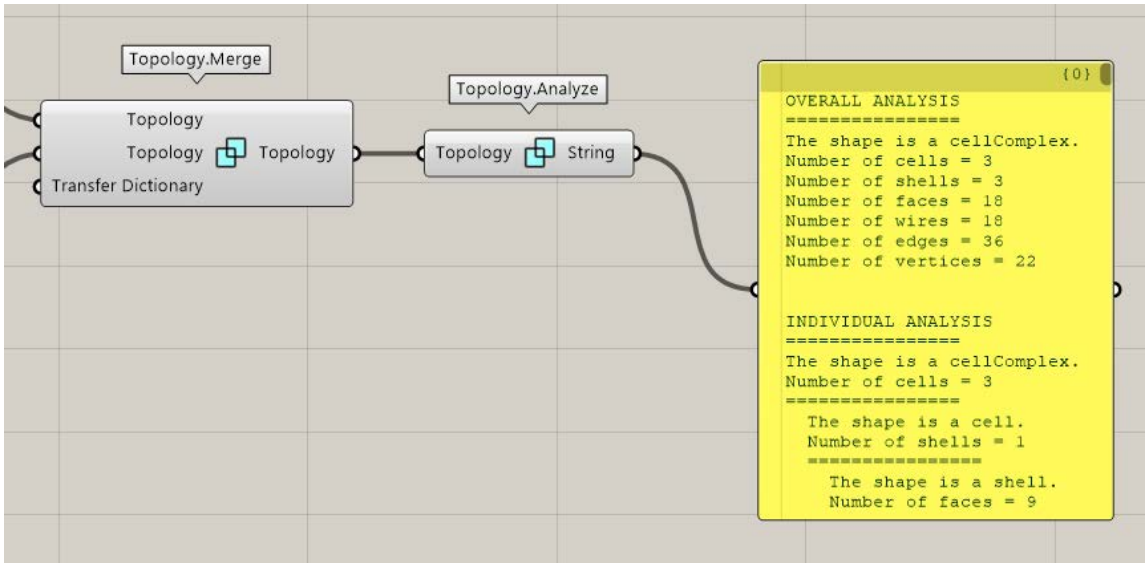


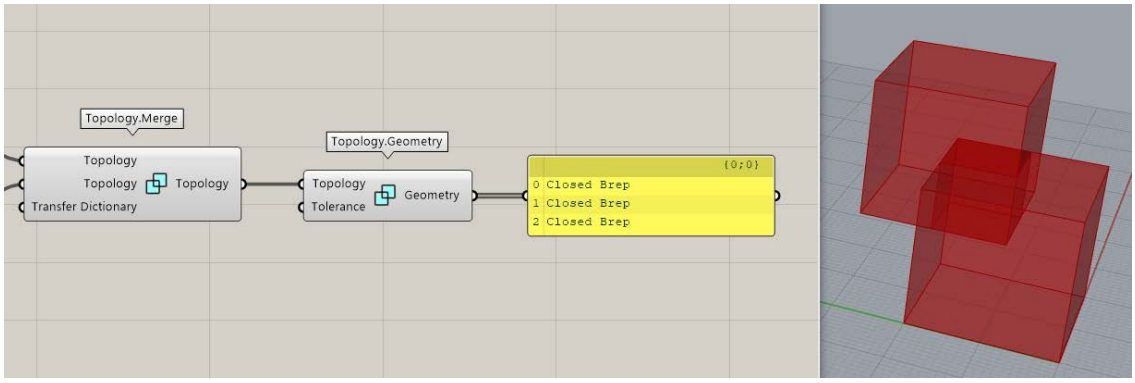
Converting to Topologic

After checking that the cubes are the correct size and at the correct location, right-click on each **Point.ByCoordinate** and **Cuboid.ByLengths** and turn OFF **Preview**. Connect each **Cuboid** output to a **Topology.ByGeometry** node then connect each output **Cell** to **Topology.Merge** as you see in the figure below.

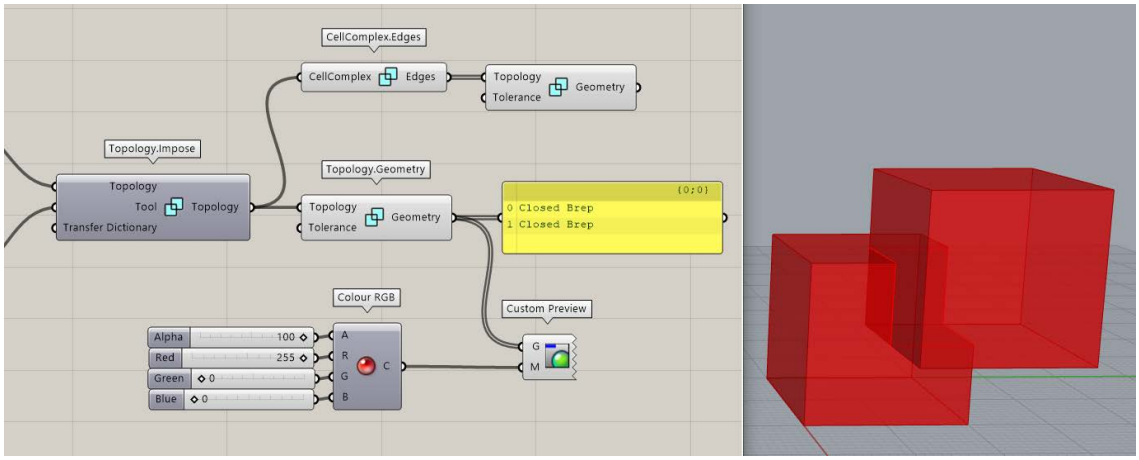


Unlike a regular union of manifold geometry, merging two intersecting cells (let's call them A and B) in Topologic yields a non-manifold CellComplex that contains three cells, the parts of cell A that are not in cell B, the parts in cell B that are not in Cell A and the parts of A and B that intersect (the small cube in the middle). We can examine this result analytically by adding a **Topology.Analyze** node and visually by using **Topology.Geometry** and examining the resulting Grasshopper geometry.





Replace **Topology.Merge** with another Boolean operations such as **Topology.Impose** and examine the results. Visually, you might not notice any difference, but if you render the solids in a translucent colour, you will notice that one cube remains whole, while the other cube is cut. It is important to note that, in Topologic, this continues to be one entity (CellComplex) made of two cells. However, since Grasshopper does not have a direct mapping to a CellComplex, Topologic converts the CellComplex into a series of solids to send to it.



Congratulations. You are getting more familiar with Topologic.

.

The Leverhulme Trust

This project is funded by a Leverhulme Trust Research Project Grant
(Grant No. RPG-2016-016)

CONTACT US

Web: topologic.app

Email: info@topologic.app

Phone: +44 (0) 29 2087 5981

Welsh School of Architecture
Cardiff University
Cardiff CF10 3NB
United Kingdom