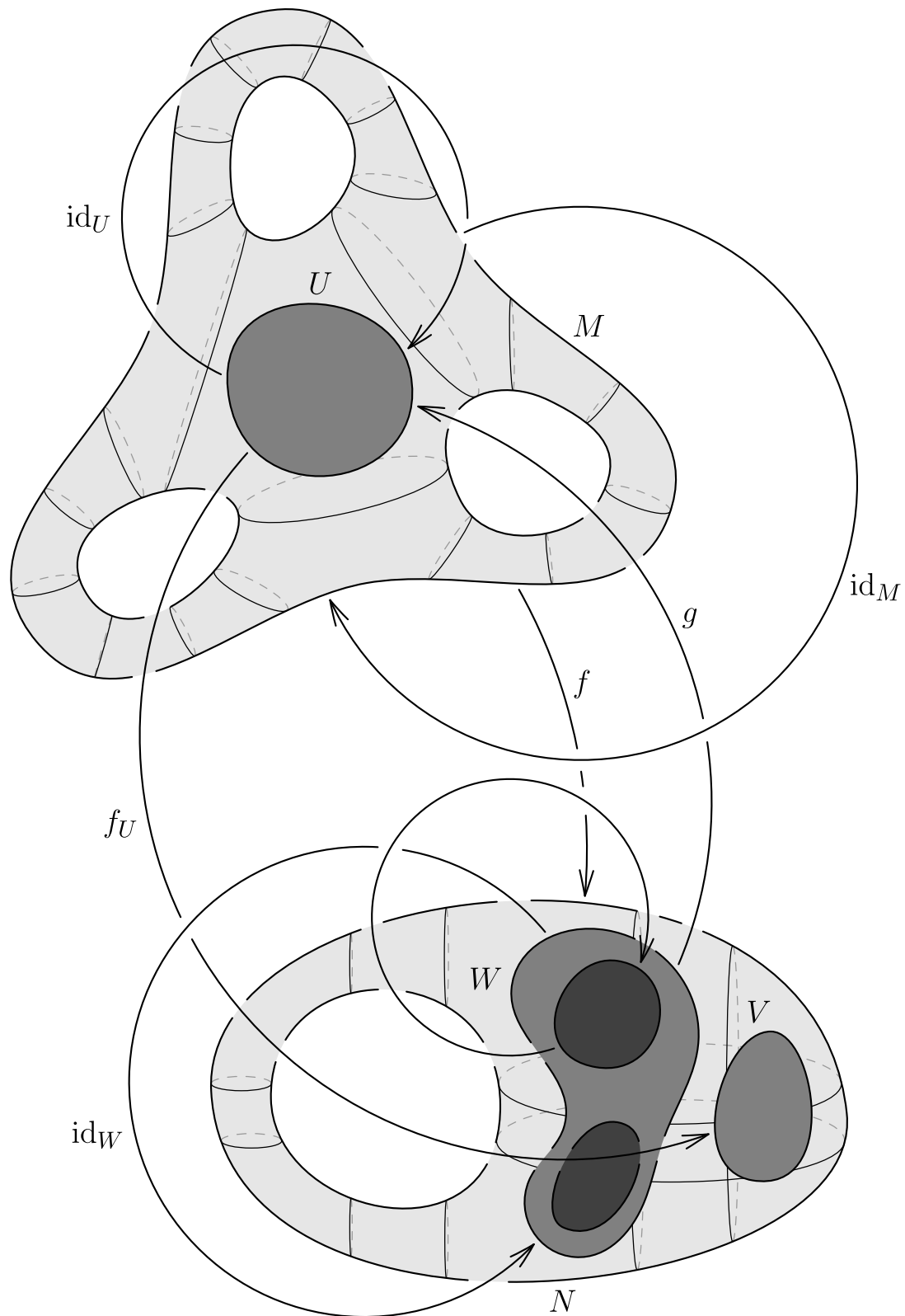# Diagrams in higher mathematics with Asymptote
Roman Maksimovich

Abstract

This document contains the full description and user manual to the `smoothmanifold` Asymptote module, home page https://github.com/thornoar/smoothmanifold.

Contents

1. Introduction

In higher mathematics, diagrams often take the form of "blobs" (representing sets and their subsets) placed on the plane, connected with paths or arrows. This is particularly true for set theory and topology, but other more advanced fields inherit this style. In differential geometry, one draws spheres, tori, and other surfaces in place of these "blobs". In category theory, commutative diagrams are commonplace, where the "blobs" are replaced by symbols. Here are a couple of examples, all drawn with `smoothmanifold`:
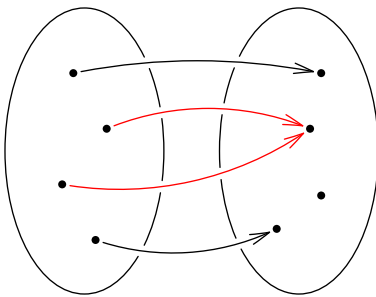

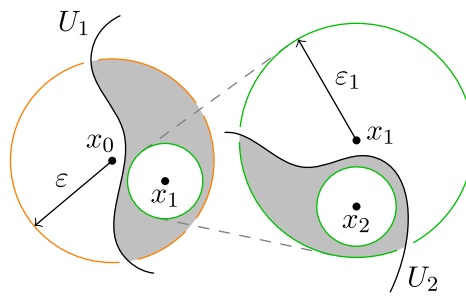Figure 1: An illustration of non-injectivity (set theory)

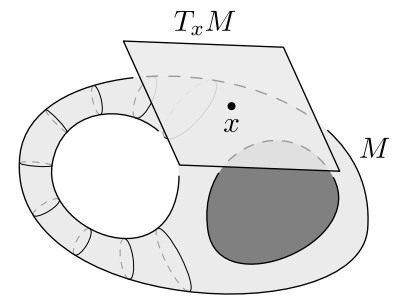Figure 2: The proof of the Baire category theorem (topology)

Figure 3: Tangent space at a point on a manifold (diff. geometry)

Take special note of the gaps that paths leave on already drawn paths upon every intersection. I find this feature quite hard to achieve in plain Asymptote, and module `smoothmanifold` uses some dark magic to implement it. Similarly, note the shaded areas on Figure 2. They represent *intersections* of areas bounded by two paths. Finding the bounding path of such an intersection is non-trivial and also implemented in `smoothmanifold`. Lastly, Figure 3 shows a three-dimensional surface, while the picture was fully drawn in 2D. The illusion is achieved through these cross-sectional "rings" on the left of the diagram.

To summarize, the most prominent features of module `smoothmanifold` are the following:

- **Gaps in overlapping paths**, achieved through a system of deferred drawing;
- **Set operations on paths bounding areas**, e.g. intersection, union, set difference, etc.;
- **Three-dimensional drawing**, achieved through an automatic (but configurable) addition of cross sections to smooth objects.

Do take a look at the source code for the above diagrams, to get a feel for how much heavy lifting is done by the module, and what is required from the user.