Modeling Uncertain Modalities

by Sven Nilsen, 2018

Uncertain modalities are important because they describe formally reasoning about uncertainty about states, such as those occurring in Newcomb-like decision problems. They are not probabilities, so one needs to learn how to think about them conceptually. In this paper I show how to model uncertain modalities as constraints on functions by using Cartesian product of a sum type and a bit.

A sum type is a kind of list of values which variable can only take one value at a time. For example, in Rust the `enum` keyword is used to declare sum types. Several common data structures such as booleans and integers are also sum types.

For simplicity, I will make up a new type `X` which takes only values `1`, `2` and `3`. I will also use the function space of type `X → X`. To picture how this function space looks like, I use a way of encoding a function such that it is easy to enumerate through all of them.

132 means 1 → 1 2 → 3 3 → 2

Next, you simply list all the functions:

111
112
113
121
122
123
131
132
133
211
212
213
221
222
223
231
232
233
311
312
313
321
322
323
331
332
333

For each possible value that the `X` type can have, the function maps to some other value of type `X`. The function could return another type, but it does not matter as long you keep in mind that the values listed above are output values, not inputs.

An uncertain modality is a function of type `X → bool` that returns `true` for all values that are inside an equality class. An equality class means that all members are equal in some way. Here, they are equal in the way that the function of type `X → X` must return the same value for all members.

For example, if the two first values `1` and `2` are in an equality class by some uncertain modality (written `(1, 2)`), it means that only functions that returns the same value for those members are accepted (blue are valid and red are invalid):

```
111
112
113
121
122
123
131
132
133
211
212
213
221
222
223
231
232
233
311
312
313
321
322
323
331
332
333
```

However, this is not the full story of what an uncertain modality means. This is where a Cartesian product comes into play. A Cartesian product is just a tuple of two types, e.g. `(X, Y)`. In Type Theory this is often written `X × Y`.

You might be familiar with the fact that when you add one bit to an integer type, you can describe twice as many numbers. We are going to need a Cartesian product because the first element shows what the function returns for individual values of `X`, while the second element, a bit, shows the constraints of uncertain modality. This gives the type `X × bit → X`. The first value of the bit is called “known” while the second element is called “unknown”. I only give a few examples because this function space is very large:

known	unknown
123	223
131	111
332	332
332	222

One can think of this as two functions. If the bit is “known”, then use the first function. If the bit is “unknown”, then use the second function. Notice that the second function must have equal value (in blue) for the equivalence class defined by the uncertain modality `(1, 2)`. This can be any value. The other values must be the same as for the “known” side.

For example, the following is not accepted:

known	unknown

123	221	ERROR: The values outside the uncertain modality are not equal

This explains what an uncertain modality means: You “write over” the outputs of the function with a new value that is the same for all values in the equivalence class defined by the uncertain modality.

It turns out that these constraints on the type `X × bit → X` can be reduced to `X’ → X` where `X’` has a new member representing the equivalence class.

known	unknown	reduced
123	223	1232
131	111	1311
332	332	3323
332	222	3322

The key insight about this process is that for every uncertain modality, there is one equivalence class, which adds one new member to the sum type. When this method is followed, the new function of type `X’ → X` automatically satisfies the constraints.

A decision problem can be though of as solving the problem which function of these are optimal to achieve a goal and according to available resources and constraints.

Uncertain modalities makes it possible for a participant in decision dilemmas to generalize from the “intuitive” problem to a harder problem simply by treating the harder problem as a new case. The uncertain modality has a formal semantics that is precise, such that all “what if” questions can be treated as further extensions.