Ning

26 September 2019

Models and Verification

Class Summary

[OPENING]: You can check the photos of my notes to have the mathematicals, but most of the stuff here will be summary and how I understand the class so far.

[SUMMARY]: From last week, we understand that we can create “sorts”, which are just how we structure and define data structures in our formal language. There are attached functions to the data structures that we make, and those functions have some action that we define through the use of variables and axioms. The idea here is that we can formally define through a set of logic what we expect our data structure to be able to do. From last week, we noticed that defining axioms is pretty open ended, and that means that we need some sort of approach or we might get lost along the way. This type of organization is called Graceful Presentation. It has two key concepts: first, we want no contradictory axioms. Second, we don’t want to “forget” to list any axioms that might be useful to describe the functions that we have. The first step is to start with defining what we want the operation (function) to do. Then we iterate from there. We first start by defining the generators of the function. We can think of this as the very beginning. For instance, we can say that in the addition function of natural numbers, we start with the generator 0, and a successor function (see page\_1). The zero element here is pretty intuitive, it seems like an apropos beginning. The intuition behind the successor function seems more iterative (at least to me) since we know that adding is just iterating a value a certain number of times, so naturally defining that iterative step makes sense. Here we hit the first part of the notion of graceful presentation: in the case of defining the successor function, we will notice that is the notion of addition, but we need to decompose it in order to define it in more broad terms. Here (page\_1) we see that we can take and decompose it to 0. Then the next step makes sense: any number plus 0 will yield itself (additive identity). Then, further decomposing, we know that the successor (which is just a fancy way of saying incrementing by “one”, I put one in quotes here because we can generalize addition to a number line that goes by increments of 2.) is the next axiom that we write. If we fit in some number there, we can see that it makes sense (for instance, lets say x is 1 and y is 3. The successor of 1 is 2, and 2+3=5. This result is the same for finding the successor of 1+3, which is to say, finding the successor of 4, which is again 5.)

[NOTE]: There’s and important note here that we made regarding using “” and decomposing it instead of using because we are concerned with positives only (hence positive conditional axioms). The technical reason behind this is more complicated, but the example that the professor gave is somewhere along the lines of an example using DeMorgan’s law . The first part will be an “for all of x” whereas the second part turns into “there exists a not x.” Which means that it doesn’t necessarily have the same... precision? or generality? maybe? That’s how I understood the cursory part of it at least.

[SUMMARY]: There are a lot of examples given in the pages (check the images), but the idea we concluded with is that like the language of logic (like the stuff we did in geometry in 9th grade), the language of algebraic specification has a syntax, deduction, semantic, and proof system. We will be focusing on the deduction and proof systems mostly (which I guess in retrospect is what we did with logic in geometry as well).