In order to implement test-driven development, it is important to first define in “jargon” or business terms the boundaries or parameters surrounding the problem. So when examining the rules of converting a roman numeral to a number or vice versa, it is important to take each constraint apart.

Some of the more obvious constraints are given by observing the equalities between the number system and the numerals as below: Numerals are defined as the roman numerals

The one’s place is defined by interacting combinations of “I”, “V” and “X”.

* These combinations start with “I” repeated up to 3 times either on its own or after “V” (with th
* Then there are two other combinations with “I” preceding “V” or “X”
* And finally “V” on it’s own

These are specific, but the schema can actually be carried forward to the other number places

The ten’s place, as you can see is defined similarly as the one’s place but instead with “X”, “L”, and “C” as the options, similar to the above the same rules apply but to these symbols.

* This group of combinations similarly start with “X” repeated up to 3 times, either on its own or after “L” or “C”
* Then there are two other combinations of the 10s place with “X” preceding “L” or “C”
* Then finally “L” on it’s own

The pattern is similar for “C”, “D”, and “M” but with the additional rule that

To make a distinction between each set of rules, it is useful to define what differentiates them, which has been done vaguely, but to be precise, it should be clearly stated.

The 1’s, 10’s, 100’s, and 1000’s place distinguish how to use the rules with 4 options for the rulesets.

This only defines the option for each roman numeral set, but does not define the conversion.

The conversion can be done from the largest part of the number downward by examining the place. If going from numbers to numerals, the only two things that need to be defined to obtain the correct conversion are the place value locations and the number in that place value.

* If that number is between 1 and 3, or 6 and 8 it follows the first framework
* If that number is 4 or 9 it follows the second framework
* And finally if the number is a 5 the last framework is followed, then the next lowest place value is evaluated

In the reverse direction the rules are slightly different, but can be more easily abstracted by looking from right to left instead of left to right. On the right, you would expect to have the smallest roman numeral. If there is a roman numeral that is smaller than it to the immediate left than you know that they are paired and can only represent a 4 in that place or a 9 in that place, which you will find by taking their number values and subtracting the smaller from the larger. If the number is the same as the other, you want to look again to the left if that number is the same, you will want to multiply those numbers by the number of times they are listed. After this is evaluated you can check the next two items, instead of changing the number however, you should add all these evaluated numbers together to get your final number.

To even start this problem, I needed to thoroughly understand it. Then to use test driven development, I had to understand test driven development, which was not as clear to me in the book as it was in some supplemental videos that I watched.