Brandon Rouleau

Oct 8, 2021

Math for CS

Are these system specifications consistent?

“If the file system is not locked, then the new messages will be queued”:

~P->Q

“If the file system is not locked, then the system is functioning normally, and conversely”:

~P<->R

“If new messages are not queued, then they will not be sent to the message buffer”:

~Q->~S

“If the file system is not locked, then then the new messages will be sent to the message buffer”

~P->S

“New messages will not be sent to the message buffer”

~S

P = the file system is locked

Q = the new messages will be queued

R = the system is working normally

S = new messages will be sent to the message buffer

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | Q | R | S | ~P->Q | ~P<->R | ~Q->~S | ~S | ~P->S |
| T | T | T | T | T | T | T | F | T |
| T | T | T | F | T | T | T | T | T |
| T | T | F | T | T | F | T | F | T |
| T | T | F | F | T | F | T | T | T |
| T | F | T | T | T | T | T | F | T |
| T | F | T | F | T | T | T | T | T |
| T | F | F | T | T | F | T | F | T |
| T | F | F | F | T | F | T | T | T |
| F | T | T | T | T | F | F | F | T |
| F | T | T | F | T | F | T | T | F |
| F | T | F | T | T | T | F | F | T |
| F | T | F | F | T | T | T | T | F |
| F | F | T | T | F | F | F | F | T |
| F | F | T | F | F | F | T | T | F |
| F | F | F | T | F | T | F | F | T |
| F | F | F | F | F | T | T | T | F |

Yes, the system specifications are consistent because there is a row on the table where all of the specifications are true.

When it comes down to it the problem was quite clear and straight forward. However, I did run into some issues along the way like keeping track of what letter I had assigned to which specification . however, the main issue I had doing this was reading the truth table and getting mixed up between the letters. To fix this problem I gave each cell of the table its own color based on whether it was true or false. One thing that did help me a lot was keeping the notes that were given in class near by to double check and confirm whether what I was doing was correct or not.

The computer science field that I chose for this assignment is software development. A software developer is someone that will generally work in a team with other programmers and will work on both the front and back ends of a project. Boolean algebra lays down almost the entirety of the groundwork for a software developer by providing a simplified version of the complicated logic that a program may have. Due to the fact that the foundation of programming is almost completely based in logic the use cases for Boolean algebra for a software developer are almost infinite. Take an if, else statement for example, if this then this. The grand majority of if, else statements contain Boolean algebra in them and can include all the Boolean algebra operators in them ex: &&, ==, ||, etc… . Another example of where Boolean algebra would be used is the base of all computers, binary. Binary is a subsequent set of ones and zeros that every computer on the planet uses. These ones and zeros work in the exact same way as Boolean algebra, if a certain number of ones are on in the string, then the computer does a certain thing, whilst in Boolean algebra you use a certain amount of trues and false that lead through gates and result in a certain output. Not only that but a software developer can use Boolean algebra to make everyday tasks simpler and more efficient. For example: if the developer gets a set of program specifications from their client, they can use Boolean Logic to see if the clients’ specifications conflict or not. This by itself could save the software developer hundreds of hours of coding trying to make something that isn’t even possible to make. All in all Boolean algebra is something that can be extremely useful to all software developers as it provides a way for them to simplify the process of going through program specifications, and it forms most of the logic that surrounds pretty much all of programming.

References:

* <https://link.springer.com/chapter/10.1007/978-1-4471-3657-6_2>
* https://en.wikipedia.org/wiki/Truth\_table