Sebastian Burke

2020-12-04

201-C01-HR

Logic Laws In Computer Science

Throughout this year, we have studied how to effectively use logic laws to our advantage in our code. When dealing with Booleans, we can use logic laws to manipulate their outcome. Below are some common logic laws that we deal with when coding:

1. DeMorgan’s Law
2. Double negation
3. Distributive Law
4. Absorption Law

Logic laws are a vital component of computer science, it is they that allow us computer science students to code the wonderful programs that our imagination pseudocodes for us. Coding is the birthstone of technology; it is the cultivated effort of millions of people throughout the last 77 years, when the first computer as created. It allows for the creation of programs and the birth of ideas via computer screens. When writing code for high-level languages like Java, C, C+, Ruby, Python, there is often arithmetic that must be implemented in order to create the desired effect of a given method/class. With this arithmetic, many common rules of code follow, one of them being logic laws. When writing if statements, or creating loops, a set of conditions must be set in order to let the program know when to execute them. When setting these conditions, the goal is to have the specific block of code run only if and when they are met. Logic laws allow for the specification of conditions that would otherwise be impossible to state. For example, DeMorgan’s Law states that when using a negation symbol (!) to state a condition, a given “&&” or “||” will be swapped correspondingly, this could create an unexpected outcome. A common practice of logic laws in the work field is if you were writing a program from your basement with an evil plan. You were going to destroy the Earth. In your program, you create a JButton and instantiate it to “BOOM BUTTON”. You code for a few weeks creating the rest of your class, when you receive a call from your best friend Timmy, he’s your watchman, if he’s calling it probably means bad news.

“[*Your name here*] the police are on their way; you need to run. Now…” *\*Beep\**

You slam your keyboard in frustration, there is still so much code to write. There is no chance you’ll be able to finish before the police get here. It must have been the girl who lives upstairs who alerted them, she seemed too nice. Plus, it probably wasn’t the best idea to yell your evil plans in a maniacal fit every night in celebration of a successful method, it’s not your fault though, that’s just typical programmer behavior. You have to leave your work behind, there is no other option.

***No.***

You can still do this, you sit back down and attempt to code 4 weeks of code in a minute, 60 seconds, you know it’s possible. You pull up your visual paradigm class diagram and get to work. By some miracle, you finish. When your front door clashes against the tiles upstairs, you know your time is up. You run the program and the Frame comes into view. You mouth the words on the button. Your cursor glides towards it as you hear thumping footsteps galloping in the main floor.

***.Click.***

Nothing…. BOOM BUTTON failed.

You are tackled to the floor and everyone in the world but you lets out a sigh of relief.

“How could this be… my code was PERFECT!”

On the way to the police station you foam from the mouth as an unconceivable rage spews endlessly from your soul. You don’t make mistakes, you coded everything exactly like you had it in your IPO… even with comments.

Then you realize, you used !( 10 && 15) when DeMorgans’s Law required you to use !(10 || 15).

How disappointing.