**Logic in Databases**

**Group 2 - Practical Task 3**

**Lecturer:** Prof. Ingo Schmitt

1. **Overview**

This program, developed in Python and utilizing the SymPy library, is designed to transform and simplify logical expressions based on specific algorithmic steps. The primary objective is to identify overlapping literals within logical expressions and systematically simplify them by applying logical rules and transformation.

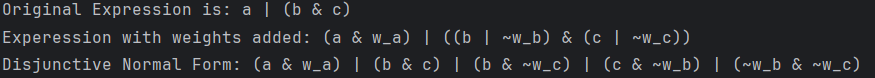
1. **Development Enviroment**

* **Language:** Python (version == 3.10)
* **Library:** SymPy (version == 1.2)
* **Tools:** Logical operators, Expression simplification, and tranformation functions provided by SymPy.

1. **Program Description**

* In this project, we create a program that take the values of symbols, alongside with the values of their weights and the logical expression as user’s input. For example, [provide the picture to demonstrate here].
* The program consists of several key functions, each corresponding to a step in the transformation process:
  1. **Add weights to the original logical expression and transform it to the Disjunctive Normal Form (DNF):**

We will take the logical expression *“Or(a, And(b,c))”* as an example for easy illustration. So the output should be:



* 1. **Simplify the expression**

We simplify the expression by applying idempotence (‘A v A = A’) and invertibility (‘A v ~ A = True’) rules

* 1. **Identification and Processing of Overlaps:**
* The first step is that we have to identify common literal across conjunctions. For easy demonstration, we will consider the logical expression without weight already tranformed to DNF form *“Or(And(a,b), And(Not(b), c), And(c,e))”*. In this example, there are only one overlap “c”, we do not consider “b” as an overlap because they are mutal exclusion XOR.

**A black background with white text

Description automatically generated**

* The next step will be replacing all conjunctions of logical expression that do not contain an overlap with **(o & x^i) | (~o & x^i)**

****