Universidad de los Andes

ISIS-4217 Paradigmas de programación
Logic programming
Date: November 23, 2025

Hand-in each of the following tasks on an independent file. Make sure of following the functions specification as well as the file name conventions. Take into account, your code will be machine graded, if it does not follow the specification, it will be marked as a non-compiling program (a 0). Additionally, make sure to test your code before handing it in, if your code does not compile or run, it will be graded as a zero.

Make sure to add examples to test the functionality of each the procedures used to solve the task. Hand in the .pl file(s) for your solution

We want to extend prolog with the possibility to reason over (a sub-set of) modal logic sentences. Therefore, we must be able to generate queries that reason about the possibility of a sentence being valid at some point in the (future) model (using the $\diamond$ operator), or at every point in the (future) model (using the $\Box$ operator).

Task 1. Build a modal logic model. Modal logic models are graphs, where each node contains a list of valid predicates in that node (for simplicity, we will consider logic predicates as simple atoms (logic constants)). This means, your knowledge base should contain the graph definition in terms of nodes and the edges between them (*e.g.,* `edge(a,b)` defines a graph of two nodes, a and b). in our case, nosed will be lists of predicates, the predicates valid in a node.

Task 2. Write the program for the modal predicates `some/3` that receives a predicate, a node, and a model and returns true if the predicate is valid at some point in the future (the $\diamond$ operator).

Task 3. Write the program for the modal predicates `always/3` that receives a predicate, a node, and a model and returns true if the predicate is valid in all future states (the $\Box$ operator).