# Processing of Logic Programming Languages

A presentation by Julie Gyselová and Luděk Burda

In this presentation, we will introduce logic programming languages and the core principles of their processing as described in Compiler Design: Virtual Machines by Reinhard Wilhelm and Helmut Seidl.

A logic program written in a logic programming language consists of a sequence of clauses (made of predicates and variable assignments) and a query that asks whether a particular statement is satisfiable for at least one variable assignment. The logic program yields a binary answer (Yes or No) if there are no unbound variables in the query. If there is an unbound variable in the query, the result of the logic program is all possible assignments of the unbound variable for which the statement is satisfiable. The main idea of processing such programs is to convert the logic program into a procedural one.

The book introduces a virtual machine that consists of a program store with a program counter that points to the instruction that is going to be processed next, a stack with a stack pointer that points to the topmost occupied location on the stack, and a frame pointer which points to the stack frame where local variables of the current call are located, and a heap in which representations of values are stored. Representations of variables in the heap correspond to their addresses on the stack. Unbound variables are symbolically represented as self-references within the heap.

The virtual machine implements an instruction for each type of node the program consists of, such as atoms, variables, or predicates. Within each node, terms are allocated in the heap and processed in a post-order, which ensures that references to successors are always readily available on the stack.

Literals (such as predicate calls or variable unifications) in logic languages are processed similarly to procedure calls in imperative languages. A key part of translating literals is unification, which finds the most general substitution of variables so that two terms become equal. A crucial part of translating predicates and clauses predicates consist of is backtracking, which explores different solutions.