

# *Logic Programming*

## *Computational Model*

Temur Kutsia

Research Institute for Symbolic Computation  
Johannes Kepler University of Linz, Austria  
`kutsia@risc.uni-linz.ac.at`

# Contents

Preliminaries

Abstract Interpreter

Choice Points

# Basic Notions

**Term:** Constant, variable, or compound term.

**Compound Term:** Functor, arguments

$$f(t_1, \dots, t_n)$$

**Functor:** Name, arity

$$f/n$$

**Goal:** Atom or compound term.

# Logic Programs

**Clause:** Universally quantified logical sentence

$$A \leftarrow B_1, \dots, B_k, k \geq 0$$

$A$  and  $B_i$ 's are goals.

**Declarative reading:**  $A$  is implied by the conjunction of the  $B_i$ 's.

**Procedural reading:** To answer the query  $A$ , answer the conjunctive query  $B_1, \dots, B_k$ .

**Logic Program:** Finite set of clauses.

# Computation

**Query:** Existentially quantified conjunction

$\leftarrow A_1, \dots, A_n, n > 0$

$A_i$ 's are goals.

**Computation of a Logic Program  $P$ :** finds an instance of a given query logically deducible from  $P$ .

# How to Compute

- ▶ Start from initial query  $G$ .
- ▶ Computation terminates – success or failure.
- ▶ Computation does not terminate – no result.
- ▶ Output of a successful computation: the instance of  $G$  proved.
- ▶ A given query can have several successful computations with different output.

# Abstract Interpreter

INPUT:

A logic program  $P$  and a query  $G$ .

OUTPUT:

$G\theta$ , if this was the instance of  $G$  deduced from  $P$ , or *failure* if failure has occurred.

# Abstract Interpreter

ALGORITHM:

**Let** *resolvent* be  $G$

**While** *resolvent* is not empty **do**

1. **Choose** a goal  $A$  from *resolvent*.
2. **Choose** a renamed clause  $A' \leftarrow B_1, \dots, B_n$  from  $P$  such that  $A$  and  $A'$  unify with an mgu  $\theta$  (**exit** if no such goal and clause exist).
3. Remove  $A$  from and **add**  $B_1, \dots, B_n$  to *resolvent*.
4. Apply  $\theta$  to *resolvent* and to  $G$ .

If *resolvent* is empty, **return**  $G$ , else **return failure**.



# Choosing and Adding

Choosing and Adding:

- ▶ Left unspecified in the abstract interpreter.
- ▶ Must be resolved in a realization of the computational model.

# Two Choices

Completely different nature.

## Choice of a goal:

- ▶ Arbitrary.
- ▶ Does not affect computation.
- ▶ If there exists a successful computation by choosing one goal, then there is a successful computation by choosing any other goal.

## Choice of a clause:

- ▶ Non-deterministic.
- ▶ Affects computation.
- ▶ Choosing one clause might lead to success, while choosing some other might lead to failure.

# Adding Goal to Resolvent

**Assume:** Always the leftmost goal to be chosen

**Then:** Adding new goal to the beginning of the resolvent gives depth-first search.

Adding new goal to the end of the resolvent gives breadth-first search.

# Prolog's Solution

- ▶ Choice of a goal: leftmost.
- ▶ Choice of a clause: Topmost.
- ▶ Adding new goal to the resolvent: At the beginning.