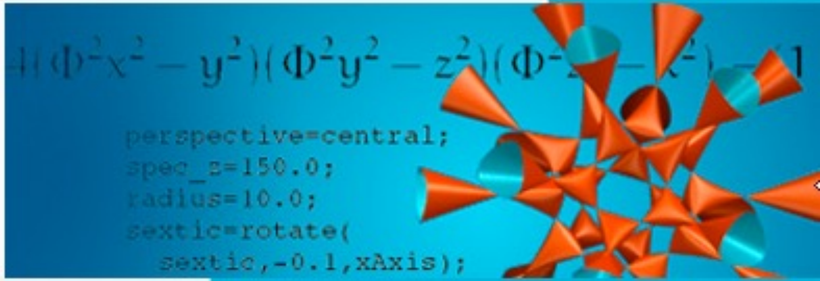
Philipps Universität Marburg

Declarative programming

Summer semester 2024

Prof. Christoph Bockisch, Steffen Dick
(Programming languages and tools)

Imke Gürtler, Daniel Hinkelmann, Aaron Schafberg, Stefan Störmer



[The art of Prolog: Introduction - 1.7]

Logic programming

- Starting point:
  - How do I want to describe problems/solutions?
  - Not: Which descriptions does the machine understand?
  - Similar to functional programming

- Logic program
  - Expressing knowledge explicitly: **logical axioms**
  - Describe the problem as a logical statement ("**target statement**")
  - Program: Set of axioms
  - Program execution: constructive proof of the target statement above the program

Logic programming

- Target statement typically quantified existentially: "There is a list X such that sorting the list of the list [3, 1, 2] results in the list X."
- Result of the program
  - Does the target statement follow from the assumptions? - Yes/No
  - Constructive proof: Specification of values for the unknowns for which the statement is true.
- Example: Assume that sufficient axioms about "sorting" have been defined:
  - X = [1, 2, 3]

History of Prolog

- Developed in the 1970s (Kowalski, Colmerauer)
- PROgramming in LOGik
- Influenced many developments:
  - 5th Generation Project
  - Deductive Databases
  - Constraint Logic Programming
- Prolog implementation:
  - SWI Prologue recommendation

SWI Prologue

- Easiest option (requires Internet access):
  - https://swish.swi-prolog.org
- For offline variant:
  - SWI Prolog Interpreter (install the 32-bit version)
    - https://www.swi-prolog.org/download/stable
  - Editor, e.g. for Windows
    - https://arbeitsplattform.bildung.hessen.de/fach/informatik/swiprolog/indexe.html
    - After the installation
      - the path to Prolog can be set (menu "Window" -> "Configuration")
      - specify the root directory of Prolog for "Prolog directory"
  - Other development environments: https://www.swi-prolog.org/IDE.html

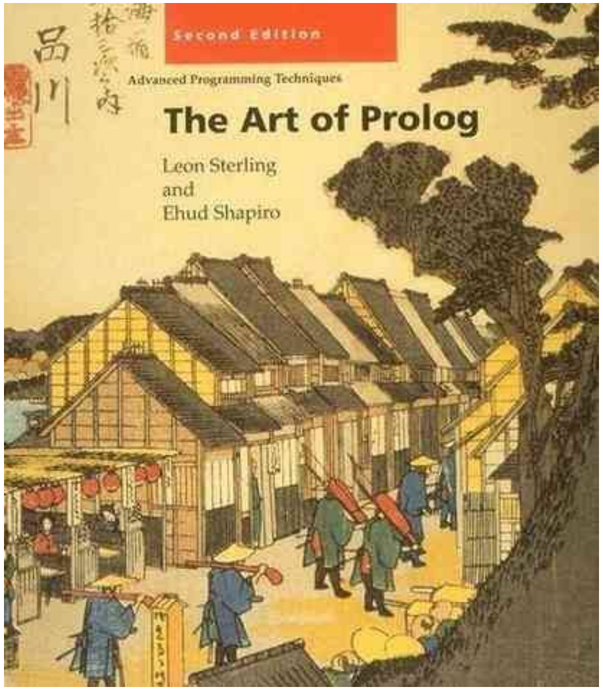
TuProlog

- No installation necessary
- Download from: https://apice.unibo.it/xwiki/bin/view/Tuprolog/
- Minimal development environment (without debugging support)
- Attention: Syntax differs from SWI Prolog in some cases. In case of doubt, the SWI Prolog syntax applies!

Literature

- Leon Sterling, Ehud Shapiro. The Art of Prologue

- http://www.learnprolognow.org



Programming in Prolog

- Programming in Prolog means
  - Facts about objects and their relationships and their relationships
  - Define rules about objects and their relationships
  - Questions about the objects and relationships

Socrates is a human being.

All people are mortal.

Is Socrates mortal?

- Only **one data structure: logical term**

Facts

- Statement: a relation (predicate) applies between objects

- Objects are displayed as atomic identifiers

- Example
  - father(abraham,isaac).

The relation "father" applies between abraham and isaac.

Relation

Atoms

- Spelling: Relations and atoms begin with a small letter

Facts

- (Arithmetic) operations can be represented as facts
  - Here incomplete
- Example
  - plus(0,0,0). plus(0,1,1). plus(0,2,2). plus(0,3,3). plus(1,0,1). plus(1,1,2). plus(1,2,3). plus(1,3,4).
- A finite set of facts defines a program

Facts

- father(terah,abraham).
  father(terah,nachor).
  father(terah,haran).
  father(abraham,isaac).
  father(haran,lot).
  father(haran,milcah).
  father(haran,yiscah).
- mother(sarah,isaac).
- male(terah).
  male(abraham).
  male(nachor).
  male(haran).
  male(isaac).
  male(lot).
- female(sarah).
  female(milcah).
  female(yiscah).

Queries

- Querying information from a logic program
- Query whether a relation between objects is fulfilled
- Example
  - ?- father(abraham, isaac).
  - *true*

Does the relation father apply between abraham and isaac?

Queries

- Queries and facts have the same syntax
  - The context determines whether it is a query or a fact
- **Fact**
  P.
  - Statement that the target P is true
- **Query**
  ?- P.
  - Question whether P is true
- A simple query consists of a single target

Philipps Universität Marburg

Queries

- Answering a query regarding a program
  - Is the query a logical consequence of the program?

- Logical consequence is determined by applying **rules of deduction**

- (Simplified) procedure:
  - Search for a fact that implies the query
  - If such a fact is found, the query is fulfilled (answer Yes)
  - The answer is no, if no such fact is found

- A first rule of deduction - ***identity***:
  - P implies P

Philipps Universität Marburg

Queries

- Answer No means
  - The statement could not be proven from the program

- Answer No does ***not*** mean,
  - That the statement is actually false

Queries

- Answer No means
  - The statement could not be proven from the program
- Answer No does ***not*** mean,
  - That the statement is actually false
- Example
  - plus(0,0,0). plus(0,1,1). plus(0,2,2). plus(0,3,3).
  - plus(1,0,1). plus(1,1,2). plus(1,2,3). plus(1,3,4).
  - ?- plus(1, 4, 5).
    - Returns *false* because the facts about the addition are incomplete

Logical variables

- **Variable**: unspecified object
  - Spelling: Identifier beginning with a capital letter

- **Abstraction** that stands for several queries

- Example: Who is Abraham the father of?
  - Go through all objects,
    and determine for which the query returns "Yes".
    - ?- father(abraham, lot).
    - ?- father(abraham, milcah).
    - ...
    - ?- father(abraham, isaac).
  - Use of logical variables
    - ?- father (abraham, X).

Logical variables

- **Variable**: unspecified object
  - Spelling: Identifier beginning with a capital letter

- **Abstraction** that stands for several queries

- Example: Who is Abraham the father of?
  - Go through all objects,
    and determine for which the query returns "Yes".
    - ?- father(abraham, lot).
    - ?- father(abraham, milcah).
    - ...
    - ?- father(abraham, isaac).
  - Use of logical variables
    - ?- father (abraham, X).

Is abraham the father of any object?

Program execution: constructive proof. Therefore the result is: Yes with X=isaac

Logical terms

- Inductive definition: Logical term
  - Constants and variables are logical terms
  - Compound terms ("structures") are terms

- Compound term
  - Functor + sequence of at least one argument
  - Arguments are terms
  - Funktor is characterized by:
    - Name
    - Arity (number of arguments)

Facts and queries are compound terms

Compound terms

- Terms that do not contain any variables are called "**basic terms**"
- Examples
  - s(0).
  - hot(milk).
  - name(john,doe).
  - list(a,list(b,nil)).
  - tree(tree(nil,3,nil),5,R).
  - foo(X).

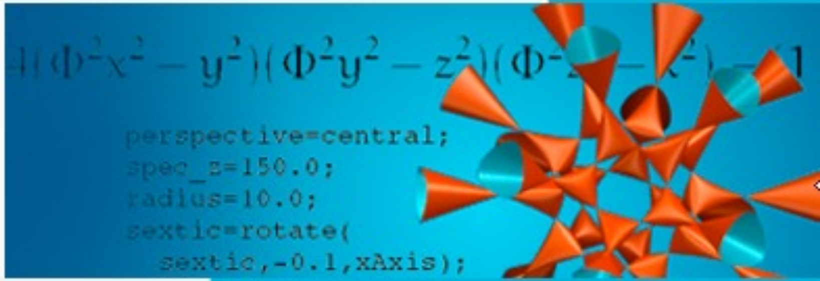
name=s, arity=1, functor- s/1

Basic term

Structures are recursive

No basic term (**non-basic term**)

An evaluation strategy for Prolog



Substitution

- **Definition**: A substitution is a finite (possibly empty) set of pairs:
  - $X_i = t_i$
  - X_i is a variable
  - t_i is a term
  - All X_i are different:
    - For all i and j with $i \neq j$, $X_i \neq X_j$
  - No variable that is being replaced occurs in one of the terms being replaced:
    - For all i and j, X_i does not occur in t_j
- We typically write the following for a substitution: θ
- Applying the substitution θ to the term A results in $A\theta$

Philipps Universität Marburg

Substitution

- Example:
  - θ = {X = isaac}
  - A = father(abraham,X).
  - Then $A\theta$ = father(abraham,isaac).

Instance

- **Definition**: A term *A* is an **instance of a term** *B*,
  - If there *is* a substitution θ so that
  - *A* arises from *B* by substitution: $A = B\theta$

- Example
  - father(abraham,isaac).
    - Is an instance of father(abraham,X).
    - With the substitution {X = isaac}
  - mother (sarah, isaac).
    - Is an instance of mother(X, Y).
    - With the substitution {X = sarah, Y = isaac}

Existence queries

- Variables in queries are "**existentially quantified**":
  - The query means: do terms exist so that the query is fulfilled if each variable is replaced by the corresponding term?
- General
  - Query ?- $p(T_1, T_2, \ldots, T_n)$. with the variables $X_1, X_2, \ldots, X_k$
    - Is there $X_1, X_2, \ldots, X_k$ so that $p(T_1, T_2, \ldots, T_n)$.?
- Example:
  - ?- father(abraham, X).
    - "Does an X exist so that Abraham is the father of X?"

Existence queries: Generalization

- **Deduction rule**: ***Generalization***
  - An existence query is the logical consequence of its instances
  - Given a query *P*
  - *P* is satisfied if there is an instance of *P* with any substitution θ
  - And $P\theta$ is fulfilled

- Example
  - The fact father(abraham,isaac).
  - Implies that there is an X,
  - So that ?- father(abraham, X). is fulfilled, namely for θ={X = isaac}

Existence queries: Generalization

- Meaning of a **non-basic term** (**generalization**)
  - Given a query with logi
  - Search for a fact that is
    - If such a fact exists, th
    - Representation of the e
    - If there is no correspon

Non-basic term: a term that contains at least one variable.

- In general, an existence query has several solutions
- Example
  - ?- father(haran,X). has the solutions {X = lot}, {X = milcah}, {X = yiscah}
  - ?- plus (X, Y, 4). has the solutions {X = 0, Y = 4}, {X = 1,Y = 3}, ...

Existence queries: Generalization

- Meaning of a **non-basic term** (**generalization**)
  - Given a query with logical variables and a program of facts
  - Search for a fact that is an instance of the query
    - If such a fact exists, then this instance is the solution
    - Representation of the solution by substitution that leads to the instance
    - If there is no corresponding fact, the result is No

- In general, an existence query has several solutions
- Example
  - ?- father(haran,X). has the solutions {X = lot}, {X = milcah}, {X = yiscah}
  - ?- plus (X, Y, 4). has the solutions {X = 0, Y = 4}, {X = 1,Y = 3}, ...

Universal facts

- Variables can also occur in facts
  - Here too: Abstraction through the reuse of commonalities
  - Variables summarize several facts

- Example:
  - Instead of
    - likes(abraham, pomegranates).
    - likes(sarah, pomegranates).
  - Universal fact
    - likes(X, pomegranates).

- Summary of rules
- Any number multiplied by 0 results in 0: `times(0, X, 0).`

Universal facts

- Variables in facts are "**universally quantified**"

- General
  - A fact $p(T_1, ..., T_n)$. with the variables $X_1, ..., X_k$ means that the p applies to all values of the variables $X_1, ..., X_{.k}$

- Example
  - likes(X, pomegranates).
    - "The following applies to all X: X likes pomegranates."

Restrictions via variables

- Variable names can be used multiple times
- Occurrences of variables must be **consistently** replaced by the same objects
- Example
  - Queries
    - `?- plus(X, X, 4).`
      - Does an X exist such that X + X equals 4? - Yes with {X = 2}
  - Facts
    - `plus(0,X,X).`
      - For all values of X, 0 + X equals X

Universal facts

- Meaning of a **basic term** (**instantiation**)
  - Given a basic query and a prog
  - Search for a fact of which the q

Basic term: a term that does not contain a variable.

- Example
  - ?- plus(0, 2, 2).
    - Yes, because it is an instance of

Basic query: a query that does not contain a variable.

Universal facts

- Meaning of a **basic term** (**instantiation**)
  - Given a basic query and a program of universally quantified facts
  - Search for a fact of which the query is an instance

- Example
  - ?- plus(0, 2, 2).
    - Yes, because it is an instance of the universal fact plus(0,X,X).