Prolog

Summer semester 2018

History and Motivation

- Alain Colmerauer/Bob Kowalski 1972
  - Programmation en logique/Programming in Logic
  - Counter-project to Lisp

    (List processing, Alonzo Church‘s Lamda calculus)

Programming in logic

Basic idea:

- Write down what is the case:
- Facts and Rules
- Ask questions to the ‚data base‘ (facts and rules)
- How to obtain the/an answer is the task of the system
- not procedural

  (don‘t bother about **how** a question is to be answerded / no programming of the search for an answer)

Prolog

Enormous upsing through

- Project:

  Fifth Generation Computing Systems

  (Japan, 1982)

  $\rightarrow$ heavy parallelism

  $\rightarrow$ logic programming (Prolog)

  (Ehud Shapiro **The family of concurrent logic programming languages** ACM Computing Surveys. September 1989)

Applications

- Expert systems
- Automatic theorem provers
- Machine translation
- Natural language access to data bases or expert systems
- Grammar development for NLP
- Intelligent text processing
- Dialog systems
- text understanding systems

Colmerauer

- Université Aix-Marseille
- Prolog for TAUM-METEO
- Prolog2
- Constraint programming

Kowalski

- Logical programming for knowledge representation (KR) und problem solving (ex.: ‚Planner‘)
- Event calculus
- …

Program

- Basic notions: Facts and Rules
- (Logical basis)
- (Proof methods)
- Data types
- Lists
- Control
- Nonmonotonic features: Negation by failure, procedural built-ins
- Meta-Programming
- (NLP-applications)

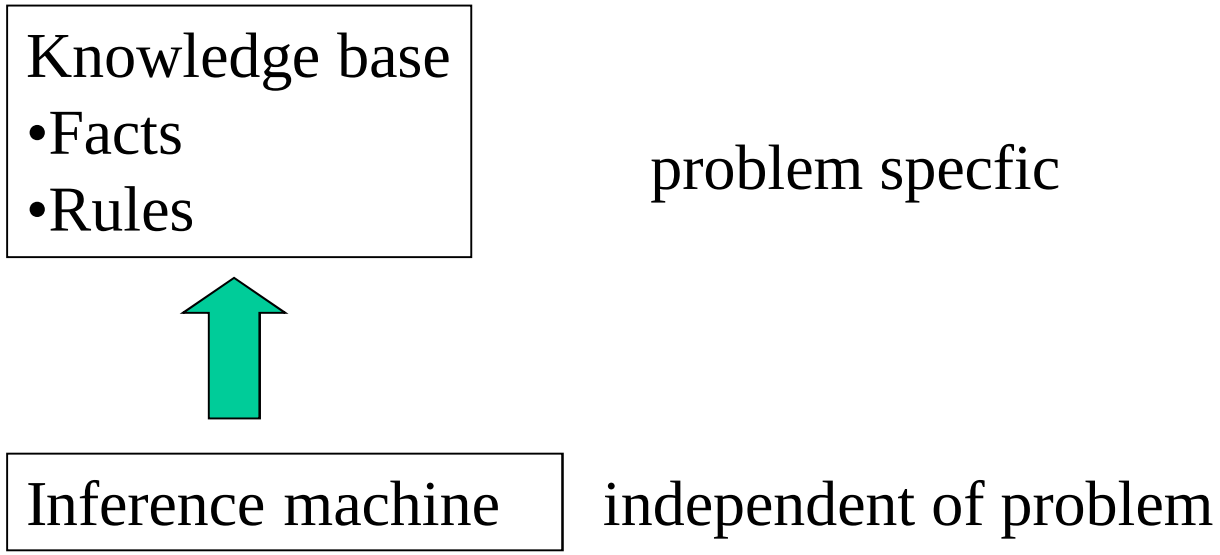
(Lit.: König/Seiffert, Blackburn et al, KleineBüning/Schmitgen)

Problem solving

- Procedural

  Program is a set of instructions and commands
  (classiscal languages Algol, Fortran, Pascal, but also Java, Python to a certain extent)

- Declarativ



Fakten und Regeln (Axiome)

- Facts
- Example: Structure of family Maier

```
person(lore).
person(gerd).
person(gitte).
person(uli).
father(gerd,gitte).
father(gerd,uli).
```

Facts

- Use comments to explain *predicats* and *arguments* used!

```
/* Family Maier: persons */
person(lore).
person(gerd).
person(gitte).
person(uli).
/* Parent-child relation */
father(gerd,gitte).
```

fact basis

```
/* Family Maier: */
/* Persons  */
person(lore).
person(gerd).
person(gitte).
person(uli).
/* Women */
female(lore).
female(gitte).
/* Men */
male(gerd).
male(uli).
/* dogs, railways */
dog(fido).
toyrailway(speb1).
/* Parent-child relation */
father(gerd,gitte).  % gerd is father of Gitte
mother(lore,gitte)  % Lore is mother of Gitte
```

Rules

```
/* Family Maier ist conservative: */
/* It holds the rule:  */
/* Every father gives his son a toy railway */

givest(X,Y,Z) :-              % gives a Z to Y, if
  father(X,Y),                    % X is father of  Y
  male(Y),                 % Y is male and
  toyrailway(Z).    % Z is toy railway
```

This ist a clause.
Clausels have the form: Head :- Body
Facts are Clauses with empty body

All clauses with the same head define a predicate

Queries

- Yes/no-questions: answer: y/n
  - Is Fido a dog?

- Alternative questions: answer: an ‚object‘
  - Who is the mother of Uli?
  - Who is the father of Gitte and Uli?
  - Who obtains the toy railway speb1 from Gerd ?

Prolog-Versions

- VM prolog (mainframe)
- IBM prolog (OS/2)
- Quintus prolog (Windows)
- Sicstus prolog (Windows/Linux)
- SWI prolog (Windows/Linux/Mac-OS)
  - freely available
  - http://www.swi-prolog.org/
- …

Quintus prolog

- download quintus and components from

  http://www.sfs.uni-tuebingen.de/~keberle/NLPTools/NLPToolsHP.html
- unzip quintus, install,
- unzip components to working dir,
- set prolog path (modify qpvars.bat respectively)
- call ‚prolog‘
- load a program

  ([PROGNAME]. or ['/Pfad/PROGNAME']. or consult(..).)

  for instance fam_maier.pl : [fam_maier].

  (Prolog programs have extension pl)
- ask queries, define new facts and rules in pl-files, load
- *trace* queries

Queries

- with constants

  ? female(gitte).

  → yes
- with variables
- ? female(X).
- → X=lore
- alternative solutions?

  ;

  → X=gitte

  ;

  No

Rules

- simple conditionals

  happy(uli) :- has_toyrailway(uli)
- conjunctively connected conditions

  balanced(uli) :- happy(uli), satisfied(uli).
- distributively connected conditions

  balanced(uli) : happy(uli); satisfied(uli).
- equivalent representation via two rules:

  balanced(uli) :- happy(uli).

  balanced(uli) :- satisfied(uli).
- universally quantified rule:

  balanced(X) :- happy(X), satisfied (X).

Prolog

- Facts and rules

  (conditions: predicates connected by ‚and‘ / ‚or‘)

  are propositions of a logic language

→ What are the basics of predicate logic?