TinyProlog: Logic programming language

Structure, unification, resolution

Tomas Petricek, Charles University

tomas@tomasp.net
@tomasp.net
https://tomasp.net
https://d3s.mff.cuni.cz/teaching/nprg077

Model of knowledge

Closed world assumption

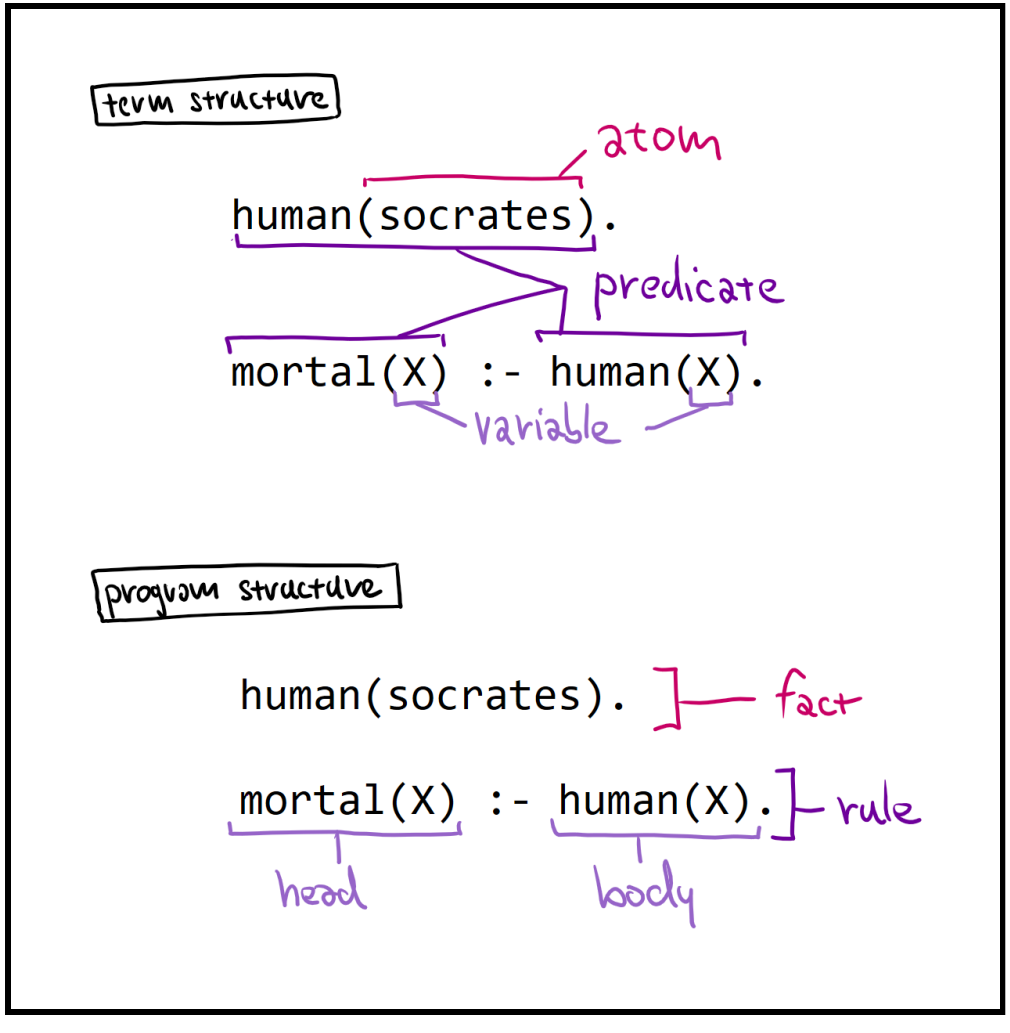
- Only declared facts are true
- No unknown children exist!
- Shapes the semantics of Prolog

Negation in Prolog

- Yes means provably true
- No means not provably true
- False only in a closed world

Charles University

TinyProlog programs



Program is a list of clauses which are:

1) Rules (head + body)
2) Facts (head)

A term can be:

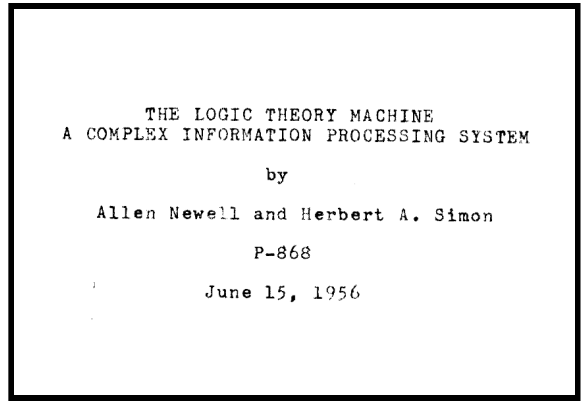
1) Variable
2) Atom
3) Predicate

Charles University

Theory behind resolution

Prolog programs as logic clauses

- Horn clause: $A \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$
- Equivalent: $A \vee \neg B_1 \vee \neg B_2 \vee \ldots \vee \neg B_n$


THE LOGIC THEORY MACHINE
A COMPLEX INFORMATION PROCESSING SYSTEM

by

Allen Newell and Herbert A. Simon

P-868

June 15, 1956

SLD resolution in Prolog

- Sound and refutation-complete resolution for Horn clauses
- Will prove 'false' if possible

Charles University

Variables in Prolog clauses

Universally quantified over formula, existentially over body

$$\forall x \forall y (grandparent(x, y) \leftarrow \exists z (parent(x, z) \land parent(z, y)))$$

Transformed using standard logical operations

$$\forall x \forall y (grandparent(x, y) \lor \neg \exists z (parent(x, z) \land parent(z, y)))$$
$$\forall x \forall y (grandparent(x, y) \lor \forall z \neg (parent(x, z) \land parent(z, y)))$$
$$\forall x \forall y \forall z (grandparent(x, y) \lor \neg parent(x, z) \lor \neg parent(z, y))$$

We need to use free variables when applying rule!

Charles University

Prolog resolution logic

- Start with user query as the goal
  Single (or multiple) term(s) with unbound variables
- Find applicable rule/fact by matching its head
  Unification to check if the rule can be applied
- Generate substitution from the matching
  Substitution generated by unification process
- Add goals based on the rule body
  Apply substitution and repeat until all goals solved

Charles University

Sketch

How resolution works

Numbers

Calculating inside Prolog

- Peano arithmetic encoded as zero & successor
- Constraint Logic Programming (CLP) extensions
- CLP(Z) adds a specialized solver for integers
- CLP(B), CLP(Q), CLP(R) and more

Charles University

Cyclic terms and occurs check

Occurs check

- Avoid terms of the form `A = f(A)`
- Supports rational trees (cyclic terms)
- Not checking is faster, but not right

Practical Prolog

- Some operations can fail:

  ```
  A = 1 + A, B is A.
  ```

- Checks can be turned on:

  ```
  set_prolog_flag(occurs_check, true).
  ```

Charles University

Demo

Enabling occurs check