

Concepts of programming languages

Prolog

Winand, Roald, Sjoerd, Alexey and Anvar



Universiteit Utrecht

[Faculty of Science
Information and Computing
Sciences]

Terms

- ▶ building blocks of facts, rules, and queries.
- ▶ 4 kinds of terms:
 - ▶ atoms
 - ▶ numbers (both are called constants)
 - ▶ variables
 - ▶ complex terms



Either:

- ▶ string of characters (upper-case, lower-case, digits, _), begins with lower-case ch. E.g.: big_kahuna_burger, listens2Music.
- ▶ arbitrary string of characters enclosed in " (single). E.g: 'The Beatles', ' &^%&#@ \$ &* '.
- ▶ string of special characters (e.g. ; or :-) E.g: in rule syntax term1 :- term2;



Numbers

- ▶ Floats (e.g. 1657.3087 or π)
- ▶ Integers (23 , 1001 , 0 , -365)
- ▶ Straightforward syntax



Variable

- ▶ starts with upper-case letter or _ (E.g: X, Y_50, List1, _input)
- ▶ anonymous variable _



Complex term

- ▶ building block: functor (which is an atom) with arguments (terms) E.g: `playsAirGuitar(jody)`, `loves(vincent,mia)`.
- ▶ nested functors make up complex terms E.g: `and(big(burger),kahuna(burger))`, `vertical(line(point(X,Y),point(X,Z)))`.



Clauses

- ▶ Rules (clauses) state information that is conditionally true of the situation of interest.
- ▶ $\text{term1} \text{ :- term2}$
- ▶ term1 is true if term2 is true.



some examples again

```
father(Y,Z):- man(Y), son(Z,Y).  
wizard(X):- hasBroom(X), hasWand(X).
```



Unification (how it works)

Two terms unify if they are the same term or if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal.



what this means??

Some examples: $* x = 1.$ $* \text{list}(X, X) = \text{list}(1, 2)$ $* X = \text{father}(X)$



more on unification..

- ▶ two terms either unify or not
- ▶ if they unify, we are interested to know how the variables have to be instantiated to make the terms unify.



more precise rules:

Two terms (term1 and term2) unify:

- ▶ 1. If they are both constants, they unify iff they are the same atom (or number)
- ▶ 2. If term1 is a variable and term2 is any term, then they unify and term1 is instantiated to term2.
- ▶ 3. If both terms are variables, they're both instantiated to each other.
- ▶ 4. If both are complex terms and ... (next slide)
- ▶ 5. Iff it follows from the rules above that they unify.



Some examples first

Terms that unify

- ▶ 1. *burger_1* and *burger_1*
- ▶ 2. *X* and *vincent* (*X* is inst-ed to *vincent*)
- ▶ 3. *X* and *Y*



For complex terms:

If term1 and term2 are complex terms, they unify iff:

- ▶ they have the same functor and arity (nr. of args)
- ▶ all their corresponding arguments unify
- ▶ the variable instantiations are compatible



Example:

Knowledge base (KB): `vertical(line(point(X,Y),point(X,Z)))`

Query: `vertical(line(point(1,1),point(2,3)))`.

Processing logic:

- ▶ 1. Try unification of the complex term `vertical` (1 argument) in the query to that in the KB.
- ▶ 2. Try unification of the functor (complex term) `line` in query and KB.
- ▶ 3. Try unification of the arguments of the functor `line`.
- ▶ 4. ... Unify `point(1, 1)` with `point(X, Y)`, instantiate `X` to 1 and `Y` to 1.
- ▶ 5. Unify `point(1, 3)` with `point(X, Z)`. Conflict: `X` has been inst.-ed to 1 and cannot unify with 2 now.
- ▶ 6. Hence, two complex terms do not unify.

