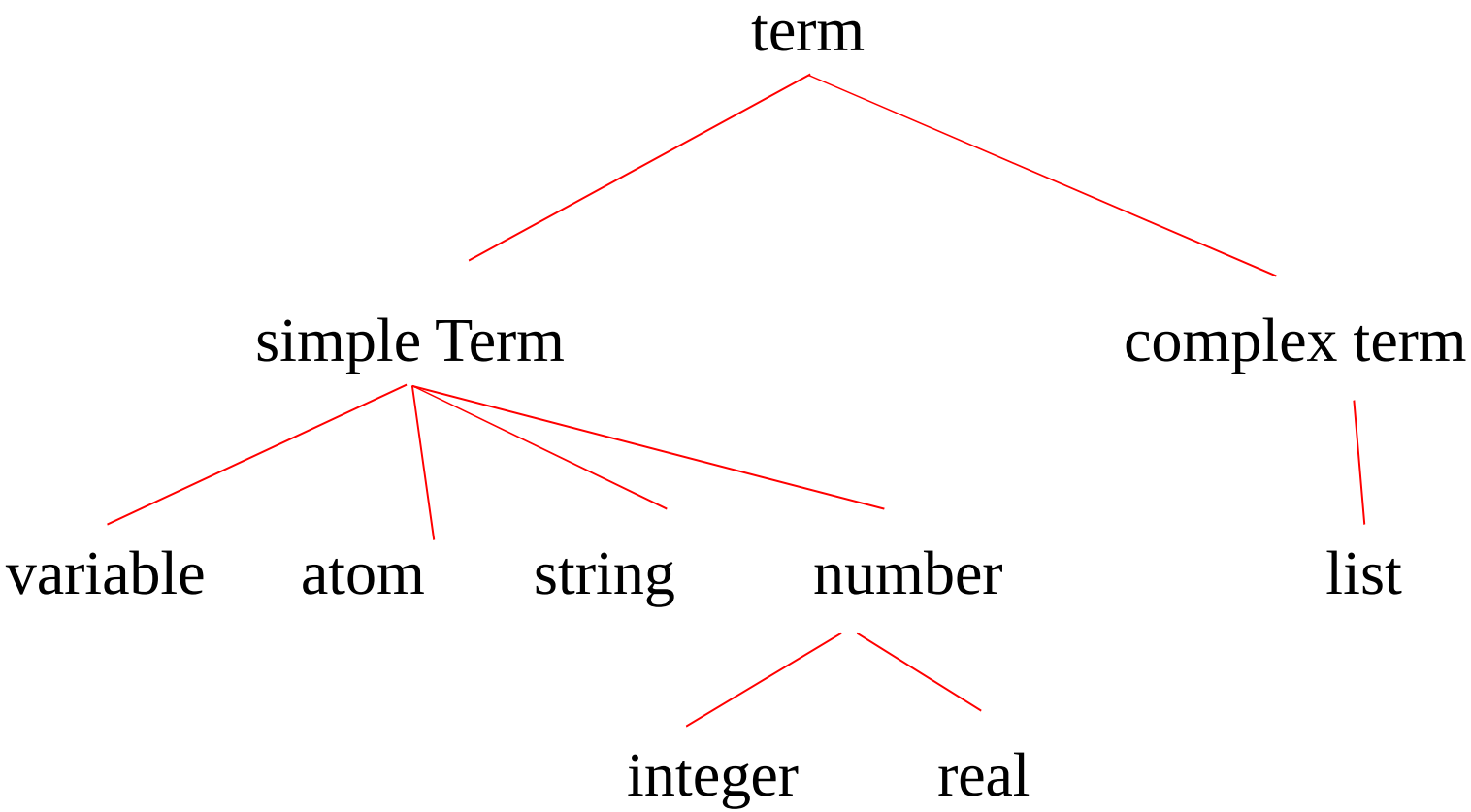
Prolog data types

- Variables
- Atoms
- Strings
- Numbers
- Functional terms
  - Lists

Prolog Data types (term types)

arguments of prolog predicates



Data types: Examples

- variable

  X, Xaver, U1, V17a, _12, _A1

  (Capital letter/underscore + alpha numerical)

- atom

  a, aaaaa, xaver, t12, 'Xaver', 'a'

  (small letter+alpha numerical or

  atom (without quotes)/variable in single quotes)

Data types: Examples

- String

  "X", "Xaver", "U1, V17a, _12, _A1"

  (in double quotes, there are also other conventions)
- Integer

  1, 2, 2333356

  (natural number)
- Real

  1.0 -23.45 187.6E12 -0.0234e15 12.0E-2

  (floating point number)

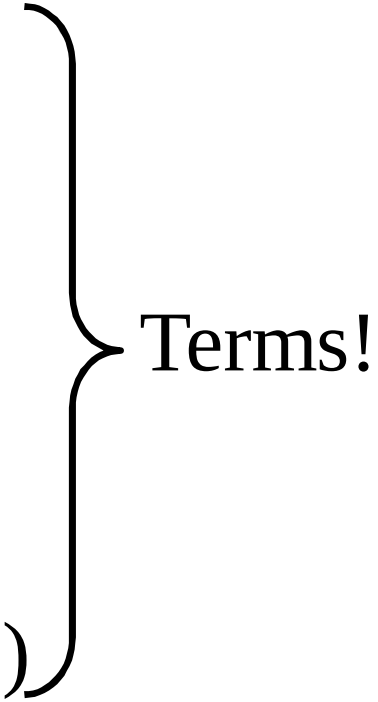
Important Built_in-predicates relating to term classes

- var(+Term)
- nonvar(+Term)
- number(+Term).
- integer(+Term)
- float(+Term)
- atom(+Term)
- string(+Term)

atomic(+Term)

Arithmetic

- + (Addition): 3 + 4
- - (Subtraction): 3 - 4
- * (Multiplication): 3 * 4
- / (Division): 21 / 5 (=4.2)
- // (Integer-Division) : 21 / 5 (=4)
- ** (to the power of): 10 ** 3 (=1000)



Arithmetic:

Evaluation of terms

- is: X is 3 *4 Definition

- =:= 3*4 =:= 4 * 3
- < 3*4+7 < 20
- =< 3*4+7 =< 19
- > 20 > 3*4+7
- >= 19 >= 3*4+7

Predicates with evaluation

Evaluation ≠ Unification !

- X is 3 *4 $\rightarrow$ Exit: X = 12

- X = 3 * 4 $\rightarrow$ Exit: X = 3 * 4

Notions of ‚Equality'

- X is 3 *4 → X is evaluation of term
- 3*4 =:= 4*3 → equality of evaluation results

- X = 3 * 4 → Unification:
  Creation of ‚identical form,
- f(X,g(Y) = f(g(Z),Z) → X=g(Z), Z=g(Y)
- unify(T1,T2) → from library(unify): with occurs_check
- \+ \+ unify(T1,T2) → Unifiability:
  tests for possibility of ‚identical form‘ only
- T1 == T2 → are T1 and T2 identical (with same vars)?

Atoms, Numbers, Strings
Conversions
Strings=List of Ids

- atom_chars(At,Li) $\rightarrow$ At=atom, Li=[aID,tID,oID,mID]
- number_chars(Nu,Li)$\rightarrow$ Nu=Nu1Nu2, Li=[Nu1ID,Nu2ID]

- X=``atom‘‘.

- name(At,Li) ...

Term Functor, Args Conversion

- f(a,b(c),d) =.. [f,a,b(c),d].

Atoms, Strings
Partial Structures

- concat(Beg,En,At) → Beg=a, En=b, At=ab
  (nonvar(En)

- append(Beg,En,St) → Beg="a", En="b", St="ab"

Modify atoms, strings

- Delete subatom (letter) from atom:

  atom_delete(to,atom,am) :- ….

  ….