Logic Programming

- clause
  - single universally quantified logical statement
  - `ab(X, f(Y), Z) :- bc(Z, f(Y), X)`
- how do we test if a clause is more specific than another? **substitution**
  - `{A = X, B = f(Y), C = Z} => reverse => {X = A, f(Y) = B, Z = C}` **wrong**
    - LHS has to be a single variable
- 3 kinds of clauses

facts	rules	queries

`prereq(CS31, CS131).`
`prereq(CS35L, CS131).`
`prereq(CS131, CS132).`
* `prereq = prt`
	∀A, B
	`prt(A, B) :- prt(A, B).`
	∀A, B, Z
	`prt(A, Z) :- prt(A, B), prt(B, Z).`
	`prt/1`
		`?- prt(CS31, R).`
		∀R (`prt(CS31, R) -> false`)
		-> `R = CS131`

- rules - clauses with conditions
- facts + rules = program
- Prolog does proof by contradiction
- It searches through the database in order to find the first answer
- Prolog does DFS on the tree, left-to-right with backtracks

```
, - and
\= - not equal
; - or
lowercase - constant
uppercase - variable
```