Deductive Programming and Unification

Prolog *terms*

- atoms
  `cs251     'hello world'  carrots`

- Variables
  `X ABC    Course  Course_number`

- compound terms: `functor(arg, U, ments)`
  `major(cs111)`
  `prereq(cs230, cs251)`

Prolog *facts* and *rules*

- facts
```
major(cs111).
major(cs230).
major(cs235).
major(cs251).
elective(cs304).
prereq(cs111, cs230).
prereq(cs230, cs235).
prereq(cs230, cs251).
prereq(cs230, cs304).
```

- rules: *head :- body.*
```
core(C) :- major(C), prereq(cs230, C).
```

  – conjuction: ,   disjunction: ;

Prolog queries

```
?- elective(cs304).
true.

?- elective(cs235).
false .

?- core(cs235).
true.

?- prereq(cs230, C).
C = cs235 ;
C = cs251 ;
C = cs 304 ;
false.
```

Unification (Prolog =)

Find environment(s)/substitution(s) under which two terms are equivalent.

Example Terms to unify	Unifying Environment
a = a	
a = X	$X \mapsto a$
p(X) = p(a)	$X \mapsto a$
p(X) = p(Y)	$X \mapsto Y$
X = a, p(a) = p(X)	$X \mapsto a$
X = a, X = Y	$X \mapsto a, Y \mapsto a$

Prolog examples: courses.pl

- Basics
- Unification
- **Unification/Proof search** algorithm demo

Applications

- Prolog (&friends):
  - AI, NLP, logic, mechanized verification
- Datalog (non-Turing complete subset):
  - data analytics, program analysis
- Unification:
  - ML type inference
  - Codder
  - proof systems, mechanized verification
  - ...

Codder example (CS 111 checker)

```
# Pattern
def sumList(_xs_):

    ___
    _sum_ = 0

    ___
    for _elem_ in _xs_:

        ___
        _sum_ += _elem_

        ___

    ___
    return _sum_
```