Logic Programming

What can go wrong?

- cyclic data structures (unification occurs check)
- control that goes wrong

Theory

- 4-part debuggiing model

```
[box 1] [box 2] [box 3]
call: box 1 -> box2
fail: box 2 -> box 1
succeed: box 2 -> box 3
backtrack: box 3 -> box 2 (where succeed is)
```

Unification

- process of unifying two terms via substitution so they become the same

```
?- member(X, [Y, Z, a])    member(A, [_, L]) :- member(A, L)
{X = A, Y = 2, L = [Z, a]}
member(A, [Y, Z, a])
```

- unification is two-way pattern match

```
p(X, X).
?- p(a, Y).
{X = a, Y = a}

?- p(f(X, g(y)), f(h(z), W))
{X = h(z), W = g(y)}

?- p(Z, f(Z))
{Z = f(f(f(...} -> infinite data structure
```

Peano Arithmetic

- Nonnegative integers, +, *, -, ∀∃

```
z represents 0
s(X) represents X + 1  s(s(Z))
plus(z, X, X).
plus(s(X), Y, s(XplusY)) :- plus(X, Y, XplusY).
lt(z, s(_)).
lt(X, s(X)).
lt(X, s(Y)) :- lt(X, Y)

?- lt(X, X).
x = s(s(s(...
```

Propositional Logic

p	~p	q	p&q	p|q	p^q	p=q	p->q	p<-q
0	1	0	0	0	0	1	1	1
0	1	1	0	1	1	0	1	0
1	0	0	0	1	1	0	0	1
1	0	1	1	1	0	1	1	1

`p <= q: p -> q`
`p >= q: p <- q`
`(p -> q & q <- q): p <--> q`

first order logic

- logical variables, ∀∃ (quantifiers)
- predicates

Prolog: facts and rules -> ~query

Scheme

- very simple syntax
  - each program has a simple representation as data
  - continuations