Prolog

Introduction to Logic Programming

Story

- Created in the 1970’s by Alain Colmerauer
- Classic AI applications:
  - Expert Systems
  - Knowledge databases
  - Natural language analysis

Append

- Scala

```
def append[a](xs:List[a], ys:List[a]) : List[a] =
xs match {
  case Nil => ys
  case x :: xs1 => x :: append(xs1, ys)
```

- Prolog

```
append([], Ys, Ys).
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).
```

Vocabulary

- Variable: Xs, Ys, Zs (first letter is uppercase)
- Predicate: `append`
  - Equivalent to a procedure that can succeed or fail.
- Clause: can be either
  - A fact: `append([], Ys, Ys)`
  - A rule: `append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs)`

Vocabulary

- Request:
  - A predicate where some parameters are determined, and others are not.
    - Undetermined parameters are defined by variables.
  - Prolog attempts to find an affectation for the undetermined parameters that makes the predicate true.

Request examples

- `?append([1], [2, 3], X).`
  ```
  X = [1, 2, 3]
  ```
- `?append([X], [2, 3], [1, 2, 3]).`
  ```
  X = [1]
  ```
- `?append([1, 2], Y, [1, 2, 3]).`
  ```
  Y = [3]
  ```
- `?append(X, Y, [1, 2, 3]).`
  ```
  X = [], Y = [1, 2, 3]
  X = [1], Y = [2, 3]
  X = [1, 2], Y = [3]
  X = [1, 2, 3], Y = []
  ```