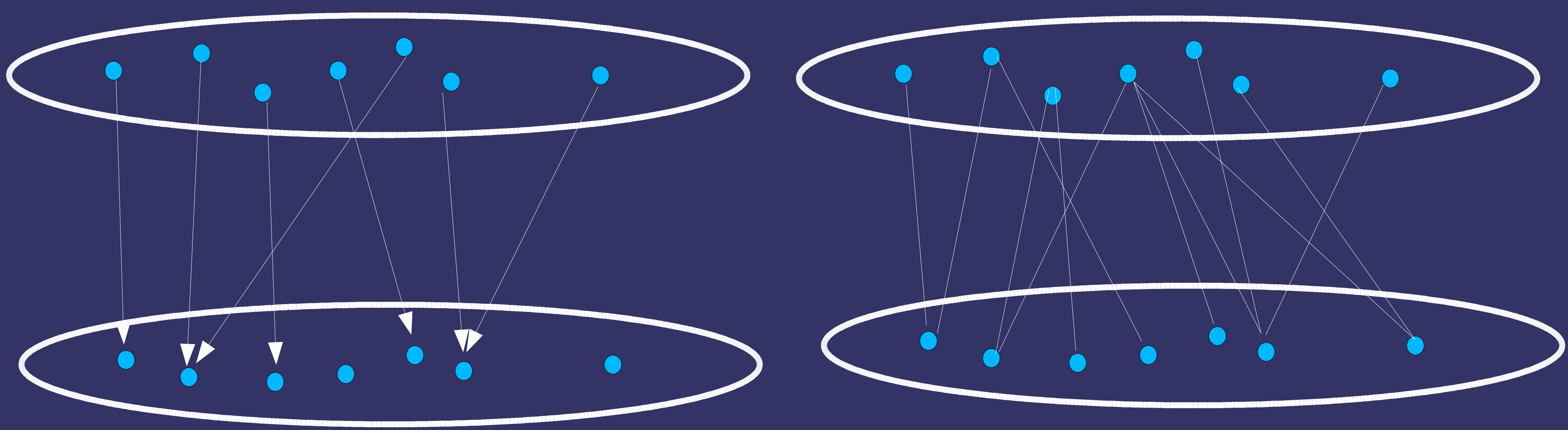
Prolog

- Logic programming
  - Origins: automatic deduction systems, theorem provers
  - Basic idea: computation can be viewed as a kind of proof
- Prolog (1970s)
  - 1981 Japan's fifth generation project

Overview

- Programs in functional and imperative languages are mappings (many to one)
- Logic programms are relations (many to many)



Append

- Relation append is a set of tuples of form (X,Y,Z) where Z consists of the elements of X followed by the elements of Y.

([a],[b],[a,b]) is in relation append
([a],[b], []) is not in relation append

First-order predicate calculus

- Constants : numbers/names
- Predicates : functions that are true or false
- Functions : non-boolean values
- Variables : unspecified quantities
- Connectives : and, or, not, implication ->
- Quantifiers : for all, there exists

Logical statements

In English:
A horse is a mammal
A human is a mammal
Mammals have four legs and no arms, or two legs and two arms
A horse has no arms
A human has arms

In FOPC:
mammal(horse).
mammal(human).
for all x, mammal(x) ->
    legs(x,4) and arms(x,0) or legs(x,2) and arms(x,2
arms(horse,0).
not arms(human,0).

Inference rule

- Infer: legs(horse,4).
- Axioms, theorems proved by inference

(a -> b) and (b->c)
----
a->c

A logical programming language is a notational system for writing logical statements together with specific algorithms for implementing inference rules

How does it work ?

Facts:

    mammal(horse).
    mammal(human).
    for all x, mammal(x) ->
        legs(x,4) and arms(x,0) or legs(x,2) and arms(x,2)
    arms(horse,0).
    not arms(human,0).

Query: there exists y, legs(human, y) ?

Answer: yes: y = 2

Deductive:
Specify properties of solutio
and find it without specifyin
exactly how

Horn Clauses

- Horn clauses
  - a_1 and a_2 and a_3 and .... a_n -> b
  - body implies head
- Can express most, but not all, logical statements

An example

English: x is a grandparent of y if x is the parent of someone who is the parent of y.

First-order predicate calculus:

for all x, for all y, (there exists z, parent(x,z) and parent(z,y) -> grandparent(x,y).

Horn clause:

parent(x,z) and parent(z,y) -> grandparent(x,y)

Procedural interpretation

- b <- a_1 and a_2 and a_3 .... and a_n
  - viewed as a procedure for obtaining b
- sort(x,y) <- permutations(x,y) and sorted(y)

```
gcd(u,0,u).
gcd(u,v,w) <- not zero(v), gcd(v, u mod v, w).
```

Resolution and Unification (how queries are expressed)

- $a <- a_1 ..... a_n$
- $b <- b_1 ..... b_m$
- If bi matches a then we can infer the clause:
- $b <- b_1, ..., b_{i-1}, a_1, ... , a_n, b_{i+1} .... , b_m.$

An example

Facts and rules:
legs(x,2) <- mammal(x), arms(x,2).
legs(x,4) <- mammal(x), arms(x,0).
mammal(horse).
arms(horse,0).

Query:
<- legs(horse,4).

Resolution:
legs(x,4) <- mammal(x), arms(x,0), legs(horse,4).

Unification:
legs(horse,4) <- mammal(horse), arms(horse,0), legs(horse,4)
<- mammal(horse), arms(horse,0).

Resolution
mammal(horse) <- mammal(hosre), arms(horse,0).
<- arms(horse,0).
arms(horse,0) <- arms(horse,0).
<-

Initial query is true

Prolog

ISO Prolog based on Edinburgh Prolog (de facto standard today)

```
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
ancestor(X,X).
parent(amy,bob).
```

Order can be important:

ancestor(x,bob).

If left to right then x is amy

If right to left then x is bob

Actual code example

Queries

Queries are yes/fail rather than yes/no
No means I can not prove it