CA341 Logic Paradigm

**Mappings vs Relations**

Imperative, Functional evaluate an output for give input. Inputs evaluate to diiferent outputs.

**Logic Programming**

Consists of assertions, horn clauses, relations

Assertions are of the form T1…™

Horn Clauses have the form A0 if A1 and..An

If all assertions A1--An are true then A0 is true.

A logic program is a collection of horn clauses.

A query Q is computed as an asssertion.

If Qis inferred from the clauses then it succeeds

Otherwise Q fails.

General Resolution is used to test query Q.

Relations are the basic units in logic programs.

Horn clauses tend to be simple.

**Prolog**

Primitive types are atoms & numbers

Atom are real-world objects

Atoms can be compared but have no additional properties.

Composite Values are structures i.e. tuples, lists

**Prolog Lists**

Special case of structure. Empty list denoted by []

Non-empty list has head and the rest of the list aka tail

Prolog is dynamically typed. Data types are interchangeable

Lists must be recursively searched.

**Arithmetic in Prolog**

Prolog focuses on symbols.

Math operations are weakness

Math is forced in Prolog using keyword IS

**Recursion in Prolog**

No control flow sequences in Prolog

Repeat evaluations using recursion

**Closed-World Assumption**

If assertion A succeeds then inferring from the clauses in program A is true

If assertion A fails then this means A is false or unknown

Prolog break math logic to conflate false and unknown

If assertion A is false then it is assumed to not be true. This is closed world assumption

Negation is unusual for this. Not A negates truth value of A.

A = true => not A = false

A = false => not A = true

A = null => not A = true

**Binding & Scope**

Scope of relation is entire program

Scope of variable is whole clause in which it occurs

Variable is - Universally Quantified

Left Hand Side

- Existentially Quantified

Right Hand Side

**Non-termination**

Order of clauses should not have an effect

Have to build procedural interpretation

Assertions on right side are tried left to right

For relations, clauses involved are tried from first to last

Base before Recursive case

**Cut**

Controls Backtracking !

Freezes assignment of values to variables that occurred before cut operator

If an assertion succeeds then

Any assignment of value-variables are frozen

Any other clauses are ignored

**Database Manipulation**

Stores clauses in internal database

Four database manipulation commands

Assert Add clauses to end of database

Asserta Add clauses to start of database

Assertz Add clause to end of database

Retract Removes matching clauses