Lecture 26: Logic Programming Languages

Design objectives

- Real world knowledge representation of objects and relationship among objects
- Have constructs which can facilitate reasoning over knowledge
- Constructs for theorem proving and question answering
- Provide extensive support in Artificial Intelligence research

Logic programming

- The program is declarative in nature.
- It consists of two important pieces of knowledge – facts and rules
- A program is used to prove if a statement is true and to answer the queries.
- There are no constructs such as loops, conditional statements or functions.
- The language supports addition of new facts and deletion of old facts.
- The language has inbuilt implementation of the inference engine to process the facts and rules.

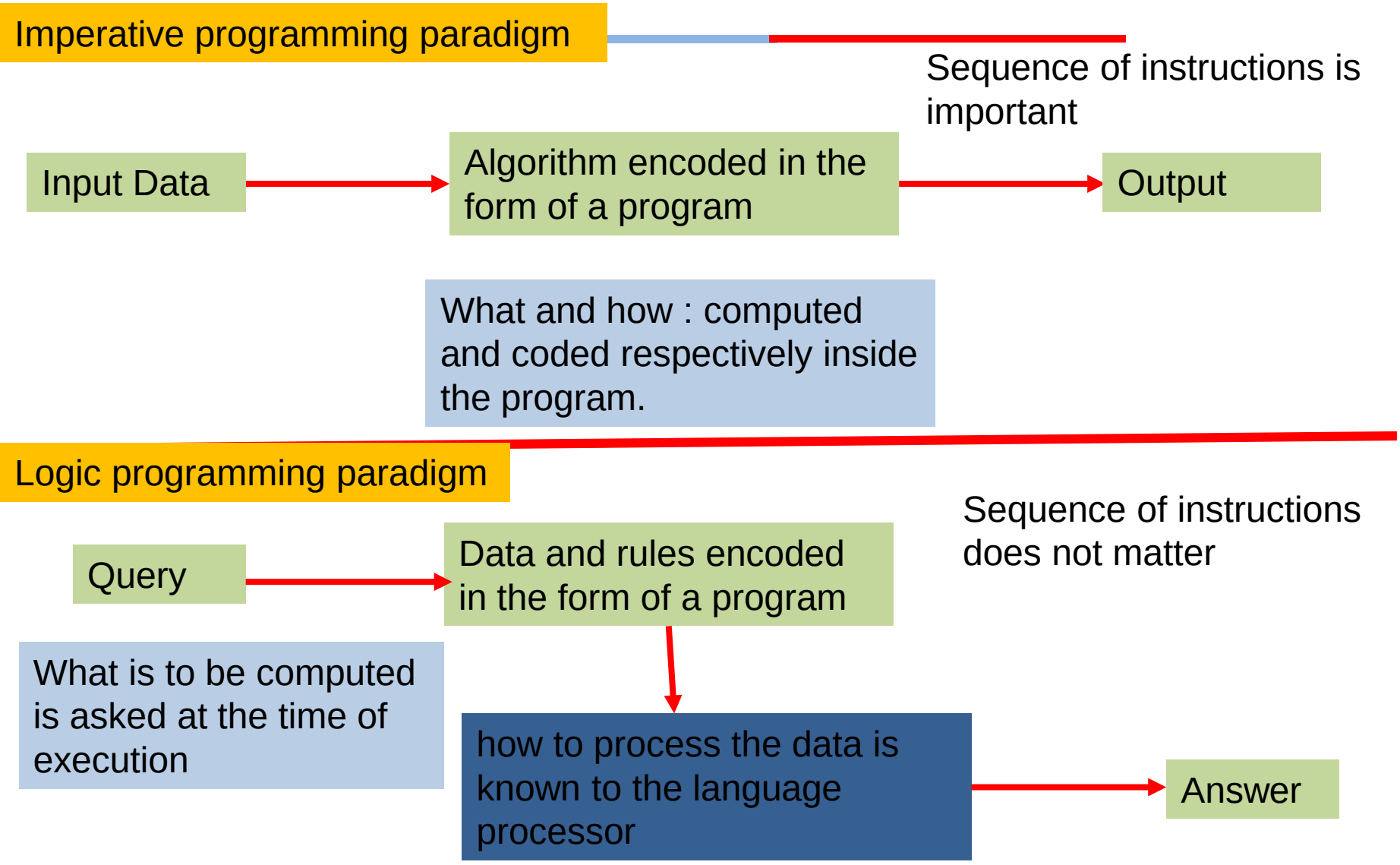
Logic Programming

- The language is declarative as the programs consist of declarations rather than the assignment and control flow statements.
- These declarations are the propositions in symbolic logic.
- The logic programming language uses declarative semantics.
- The declarative semantics is simpler than semantics of imperative languages.
- The meaning of a proposition does not depend on the textual context or execution sequence.

Logic programming

- The programming in a logic programming language is nonprocedural unlike the imperative and functional languages.
- The programmer of imperative and functional languages know exactly what is to be accomplished and instructs the computer on exactly how to achieve that.
- The programs in logic languages do not need to specify exactly how a result is to be computed. The language implementation is such that it knows how the result will be calculated.

Imperative Vs. logic program execution



Knowledge processing in logic

- The programmer codes the knowledge of the real world in logic programming language.
- The language implements the algorithm such as **forward chaining, backward chaining, resolution** etc. internally to process the supplied knowledge.
- **Resolution** is an inference rule that allows inferred propositions to be computed from given propositions.

Comparative ways of thinking in different language paradigms

Insertion sort in Haskell programming language – A functional approach

```haskell
insert :: Int -> [Int] -> [Int]
insert x [] = [x]
insert x (y:ys)
   | x < y =  x:y:ys
   | otherwise = y : insert x ys
insertionSort :: [Int] -> [Int]
insertionSort [x] = [x]
insertionSort (x:xs) = insert x (insertionSort xs)

main = do
   print(insertionSort [3, 19, 1, 872, 56, 39, 23, 10, 345, 180, 200, 2, -10])
```

Recursive function call returning a list

Insertion sort in Prolog programming language – A predicate calculus based logical approach

```prolog
insert_sort(List, Sorted):-isort(List, [],Sorted).
isort([], Acc, Acc).
isort([H|T], Acc, Sorted):- insert(H, Acc, New), isort(T, New, Sorted).
insert(X, [Y|T], [Y|New]):-X>Y, insert(X,T,New).
insert(X, [Y|T], [X,Y|T]):-X=<Y.
insert(X, [], [X]).
```

Predicate evaluated to true or false

Prolog

- It is a programming language that was developed in 1972.
- It was initially used for natural language processing.
- It is widely used for specifying algorithms, searching databases, writing compilers, pattern matching etc.
- The language deals with representation of the facts and rules, and processes them using implicit inference engine.
- Logic programing deals with relations rather than functions.

Logic

- A proposition is a logical statement that is made if it is true. E.g. Today is Tuesday.
- Symbolic logic uses propositions to express the objects of the real world and relationship between them. E.g. reads(X, Y)
- The propositions can be atomic or compound.
- For symbolic logic, the first order predicate calculus is used.

Propositions

- The objects are represented by simple terms.
- The terms are either constants or variables.
- The propositions consist of compound terms.
- A compound term is composed of two parts: a functor and an ordered list of parameters.
- The functor is the function symbol that names the relation. E.g. likes(jerry, tom), friends(X, jerry) etc. These are also known as predicates in First Order Logic.
- Propositions can either represent the facts and rules, or they can represent the query.

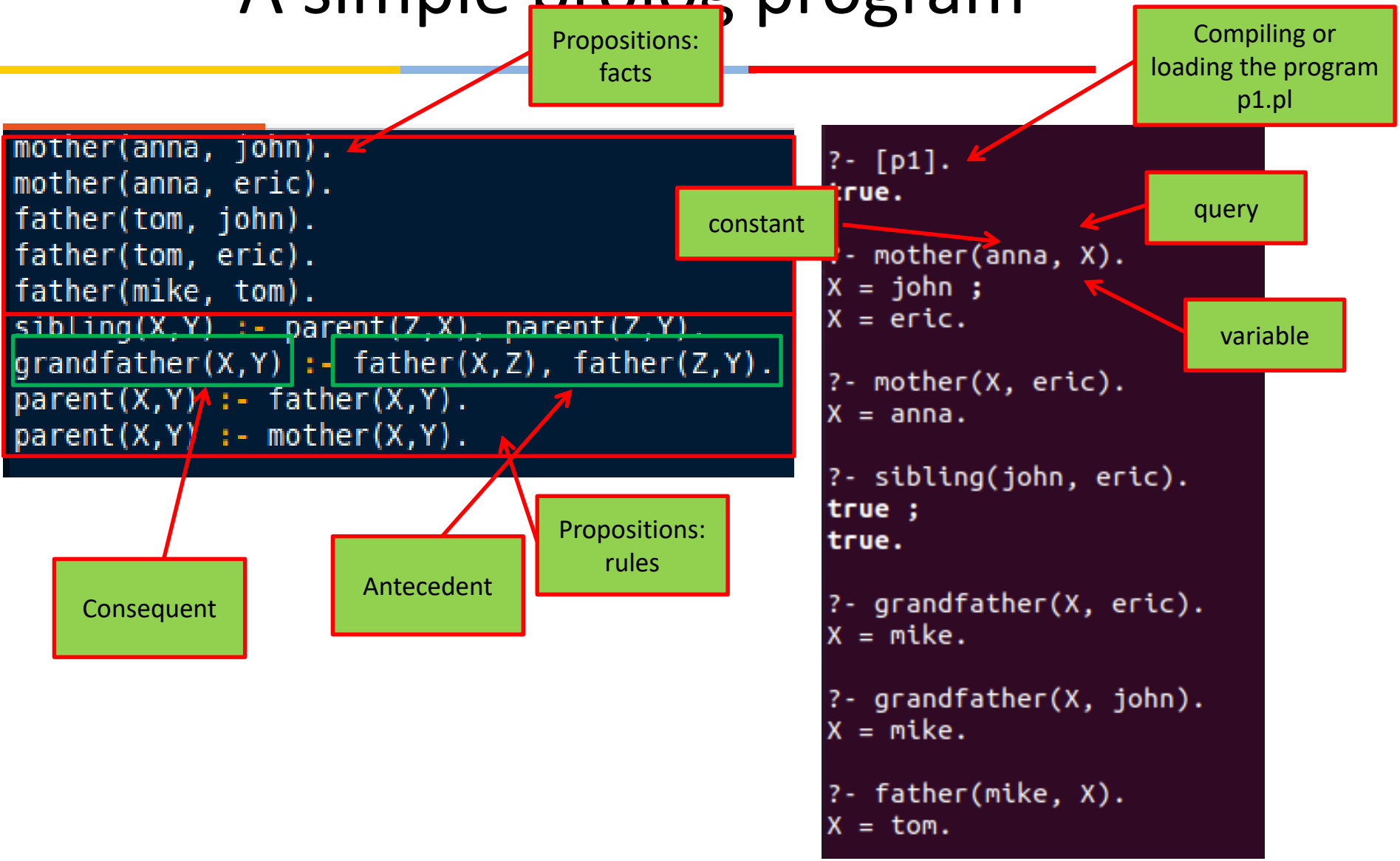
Basic Elements of Prolog

- Term: A Prolog term is a constant, a variable or a structure. A constant is either an atom or an integer.
- Atom: These are the symbolic values of Prolog. E.g., delhi, alice, bob, abc_d etc.
- Variables: Variable is any string of characters or digits that begin with _ (in some versions of prolog) or with a letter in upper case.
- Structure: They represent the atomic propositions of predicate calculus.

Data and program in Prolog

- Distinction between data and program is blurred in Prolog. For example, mother(eric) is itself a data (fact) and is also an atomic proposition.
- Facts and rules are used as data.

A simple prolog program



Program execution

```
mother(anna, john).
mother(anna, eric).
father(tom, john).
father(tom, eric).
father(mike, tom).
sibling(X,Y) :- parent(Z,X), parent(Z,Y).
grandfather(X,Y) :- father(X,Z), father(Z,Y).
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

Query:
?- mother(X, john).

Processing
Available clauses
1. mother(anna, john).
2. mother(anna, eric)

Combines query clause and facts and rules
mother(anna, john) $\cap$ mother(X, john)
mother(anna, eric) $\cap$ mother(X, john)

It then matches the literals in the query and uses substitution for X as σ = {X/anna}

Program execution

```
mother(anna, john).
mother(anna, eric).
father(tom, john).
father(tom, eric).
father(mike, tom).
sibling(X,Y) :- parent(Z,X), p
grandfather(X,Y) :- father(X,Z
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

Query:
?- grandfather(X, eric).

Processing
Available clauses
1. Grandfather(X,Y) :- father(X,Z), father(Z,Y

First it instantiates Y using σ = {Y/eric}

grandfather(X, eric) :- father(X,Z), father(Z,eric)

Looks for predicate father's instances

father(tom, john)
father(tom, eric)

Uses σ = {Z/tom} to get

grandfather(X, eric) :- father(X,tom), father(tom,eric)

Next uses father(mike, tom) and instantiates
grandfather(mike, eric) :- father(mike,tom), father(tom,eric)

As the antecedent is true, the consequent becomes tru with σ = {X/mike}

Output is X = mike