LOGIC PROGRAMMING

TIM AJAR KECERDASAN ARTIFISIAL

AGENDA

- DEFINITION
- SOLVING PROBLEM USING LOGIC PROGRAMMING

WHAT IS LOGIC PROGRAMMING?

- The concept of programming paradigms originates from the need to classify programming languages. It refers to the way computer programs solve problems through code
- Here are some of the more popular programming paradigms:
  - Imperative: Uses statements to change a program's state, thus allowing for side effects.
  - Functional: Treats computation as an evaluation of mathematical functions and does not allow changing states or mutable data.
  - Declarative: A way of programming where programs are written by describing what needs to be done and not how to do it. The logic of the underlying computation is expressed without explicitly describing the control flow.

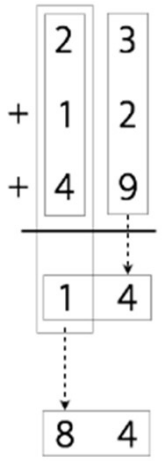
- Object oriented: Groups the code within a program in such a way that each object is responsible for itself. Objects contain data and methods that specify how changes happen.
- Procedural: Groups the code into functions and each function is responsible for a series of steps.
- Symbolic: Uses a style of syntax and grammar through which the program can modify its own components by treating them as plain data.
- Logic: Views computation as automatic reasoning over a database of knowledge consisting of facts and rules

Logic programming has been around for a while. A language that was quite popular during one of the last heydays of AI was Prolog. It is a language that uses only three constructs:

- Facts
- Rules
- Questions

In order to understand logic programming, it's necessary to understand the concepts of computation and deduction. To compute something, we start with an expression and a set of rules. This set of rules is basically the program. Expressions and rules are used to generate the output.

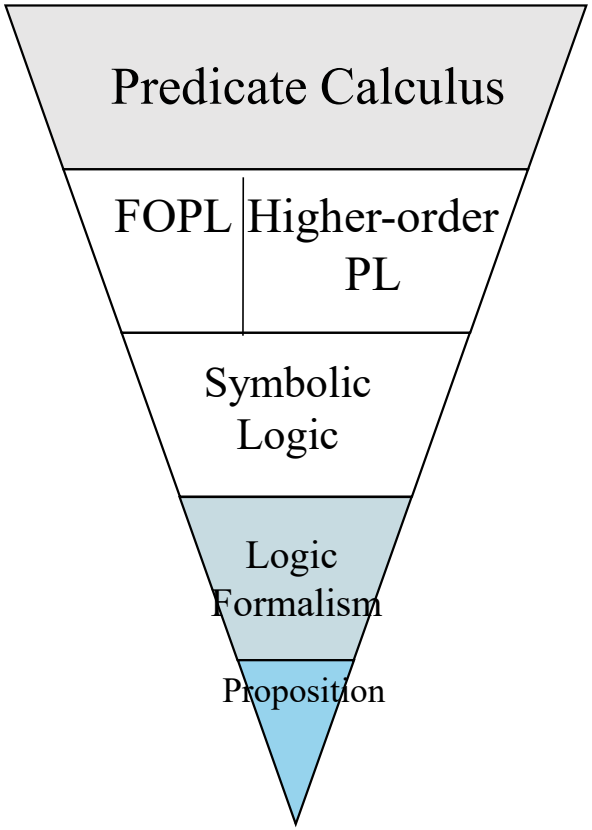
Example: Let's compute the sum of 23, 12, 49



The procedure to complete the operation would be as follows:

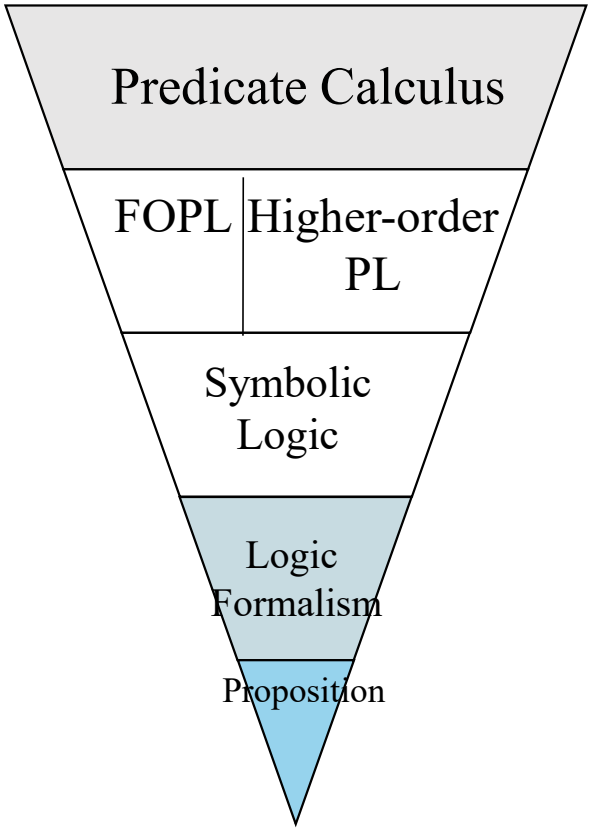
1. Add 3 + 2 + 9 = 14
2. We need to keep a single digit, which is the 4, and then we carry the 1
3. Add 2 + 1 + 4 (plus the 1 we carried) = 8
4. Combine 8 and 4. The final result is: 84

UNDERSTANDING THE BUILDING BLOCKS OF LOGIC PROGRAMMING



- Predicate calculus also called logic of quantifiers is mathematical representation of formal logic that used for logic programming
- FOPL, First-order programming logic refers to logic in which the predicate of a sentence or statement can only refer to a single subject

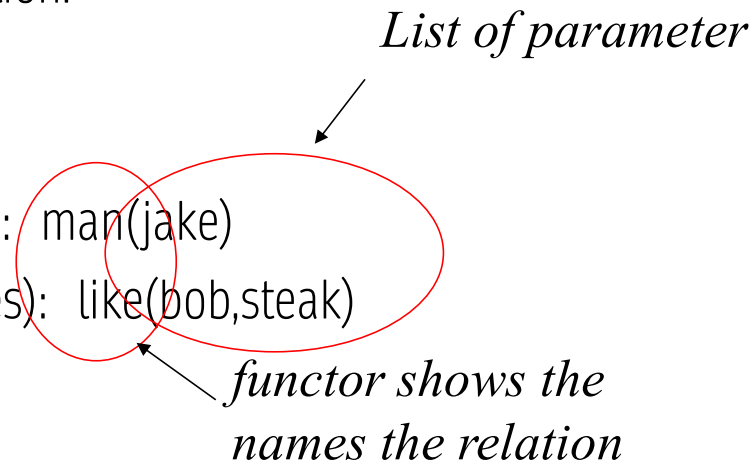
UNDERSTANDING THE BUILDING BLOCKS OF LOGIC PROGRAMMING (1)



- Symbolic logic used for the three basic need of formal logic
  - to express propositions
  - to express the relationships between propositions
  - to describe how new propositions can be inferred from other propositions that are assumed to be true
- Formal logic was developed to provide a method for describing proposition.
- Proposition is a logical statement also known as fact
- Consist of object and relationships of object to each other

PROPOSITION

- Object:
  - ❑ Constant represents an object, or
  - ❑ Variable represent different objects at different times
- Simple proposition called as atomic propositions, consist of compound terms – one element of mathematic relation which written in a form that has the appearance of mathematical function notation.



- Example (constants):

  single parameter (1-tuple): man(jake)

  double parameter (2-tuples): like(bob,steak)

PROPOSITION

- Two modes for proposition:
  - proposition defined to be true (fact), and
  - the truth of the proposition is something that is to be determined (queries)
- Compound propositions have two or more atomic proposition, which are connected by logical operator (is the same way logic expression in imperative languages)

LOGIC OPERATORS

Name	*Symbol*	*Example*	*Meaning*
negation	¬	¬ a	not a
conjunction	∩	a ∩ b	a and b
disjunction	∪	a ∪ b	a or b
equivalence	≡	a ≡ b	a is equivalent to b
implication	⊃	a ⊃ b	a implies b
	⊂	a ⊂ b	b implies a

TRUTH TABLES

A	B	~A	~B	$A \cup B$	$A \cap B$	$A \supset B$
T	T	F	F	T	T	T
F	T	T	F	T	F	T
T	F	F	T	T	F	F
F	F	T	T	F	T	T

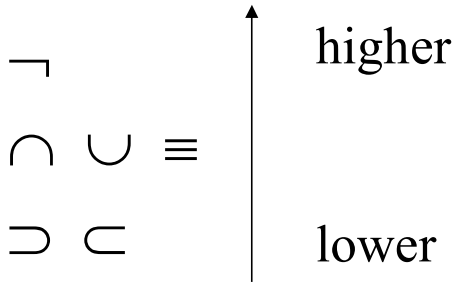
COMPOUND PROPOSITIONS

Example:

$a \cap b \supset c$

$a \cap \neg b \supset d$

Precedence:



VARIABLES IN PROPOSITION

- Variable known as *quantifiers*
- Predicate calculus includes two quanifiers, X – variable, and P – proposition

Name	*Example*	*Meaning*
universal	∀X,P	For all *X*, *P* is true
existential	∃X,P	There exists a value of *X* such that *P* is true

EXAMPLE: VARIABLE IN PROPOSITION

Example

$\forall X.\,(\mathrm{woman}(X) \supset \mathrm{human}(X))$

$\exists X.\,(\mathrm{mother}(\mathrm{mary}, X) \cap \mathrm{male}(X))$

VARIABLES IN PROPOSITION

Example

$\forall X.\ (\texttt{woman}(X) \supset \texttt{human}(X))$

$\Rightarrow$ for any value of X, if X is a woman, then X is a human (NL: woman is a human)

$\exists X.\ (\texttt{mother}(\texttt{mary}, X) \cap \texttt{male}(X))$

$\Rightarrow$ there exist a value of X such that mary is the mother of X and X is a male (NL: mary has a son)

CLAUSAL FORM

- Simple form of proposition, it is a standard form for proposition without loss of generality
- Why we need to transform PC into CF?
  - too many different ways of stating propositions that have the same meaning

Example:

$\forall X.\ (\mathbf{woman}(X) \supset \mathbf{human}(X))$

$\forall X.\ (\mathbf{man}(X) \supset \mathbf{human}(X))$

CLAUSAL FORM

- General syntax for CF

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

$\Rightarrow$ if all the As are true, then at least one B is true

Example:

human(X) $\subset$ woman(X) $\cap$ man(X)

likes(bob, trout) $\subset$ likes(bob, fish) $\cap$ fish(trout)

CLAUSAL FORM

Example:

$$likes(bob, trout) \subset likes(bob, fish) \cap fish(trout)$$

consequent antecedent

- Characteristics of CF:
  - Existential quantifiers are not required
  - Universal quantifiers are implicit in the use of variables in the atomic propositions
  - No operator other than conjunction and disjunction are required

CLAUSAL FORM

Example:

$$\texttt{likes(bob, trout)} \subset \texttt{likes(bob, fish)} \cap \texttt{fish(trout)}$$

$\Rightarrow$ if bob likes fish and trout is a fish, then bob likes trout

CLAUSAL FORM

Example:

$\texttt{father(louis, al)} \cup \texttt{father(louis, violet)} \subset \texttt{father(al, bob)} \cap \texttt{mother(violet, bob)} \cap \texttt{grandfather(louis, bob)}$

$\Rightarrow$ if al is bob's father and violet is bob's mother and louis is bob's grandfather, louis is either al's father or violet's father

PROVING THEOREMS

- Method to inferred the collection of proposition
  - use a collections of proposition to determine whether any interesting or useful fact can be inferred from them

- Introduced by Alan Robinson (1965)

PROVING THEOREMS

- Alan Robinson introduced resolution in automatic theorem proving
  - resolution is an inference rule that allows inferred proposition to be computed from given propositions
  - resolution was devised to be applied to propositions in clausal form

PROVING THEOREMS

- Idea of resolution:

$P1 \subset P2$ and $Q1 \subset Q2$

which given

P1 is identical to Q2

$\therefore$ $Q1 \subset P2$

PROVING THEOREMS

Example:

older(joanne, jake) $\subset$ mother(joanne, jake)
wiser(joanne, jake) $\subset$ older(joanne, jake)

$\therefore$ wiser(joanne, jake) $\subset$ mother(joanne, jake)

SOLVING PROBLEMS USING LOGIC PROGRAMMING

Logic programming looks for solutions by using facts and rules. A goal must be specified for each program. When a logic program and a goal don't contain any variables, the solver comes up with a tree that constitutes the search space for solving the problem and getting to the goal.

Let's consider the following:

Kathy orders dessert => Kathy is happy

This can be read as an implication that says, If Kathy is happy, then Kathy orders dessert. It can also be construed as Kathy orders dessert whenever she is happy

SOLVING PROBLEMS USING LOGIC PROGRAMMING (1)

let's consider the following rules and facts:

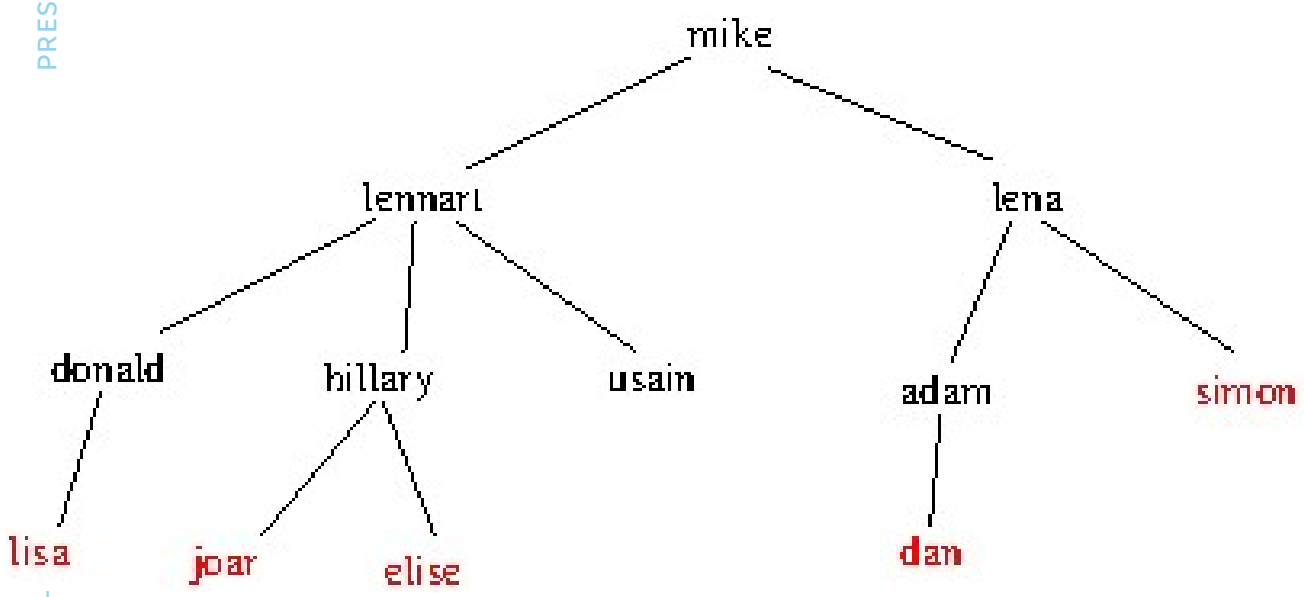
```
canfly(X) :- bird(X), not abnormal(X).
abnormal(X) :- wounded(X).


bird(john).
bird(mary).
wounded(john).
```

Here is how to interpret the rules and facts:

- John is wounded
- Mary is a bird
- John is a bird
- Wounded birds are abnormal
- Birds that are not abnormal can fly

From this, we can conclude that Mary can fly, and John cannot fly.

FAMILY TREE

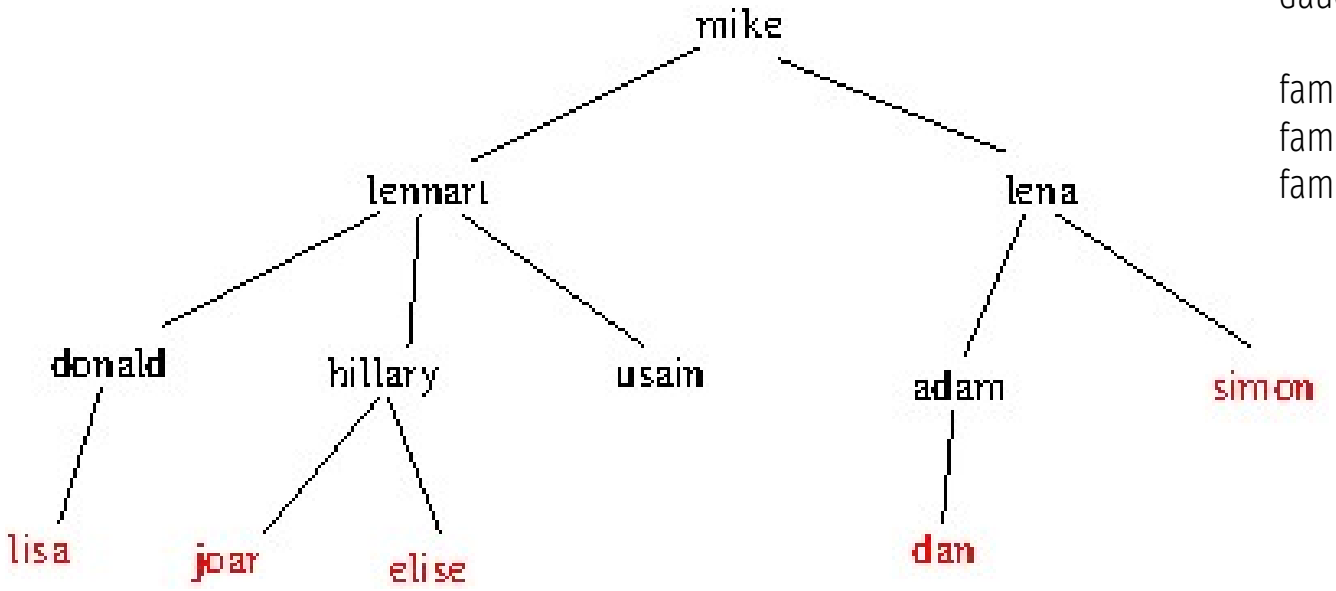



```
male(lennart).
male(mike).
male(donald).
male(usain).
male(joar).
male(adam).
male(dan).
male(simon).

female(hillary).
female(elise).
female(lisa).
female(lena).
```

```
parent(mike, lennart).
parent(mike, lena).
parent(lennart, donald).
parent(lennart, hillary).
parent(lennart, usain).
parent(lena, adam).
parent(lena, simon).
parent(adam, dan).
parent(donald, lisa).
parent(hillary, joar).
parent(hillary, elise).
```

```
child(lisa).
child(joar).
child(elise).
child(dan).
child(simon).
```

FAMILY TREE




```
%% predicate rules
father(X,Y) :- male(X),parent(X,Y).
mother(X,Y) :- female(X),parent(X,Y).
son(X,Y) :- male(X),parent(Y,X).
daughter(X,Y) :- female(X),parent(Y,X).

family_children(X, X):-child(X).
family_children(X, Child):-parent(X,Y),
family_children(Y, Child).
```

VALIDATING PRIMES

```
# to install logpy use:
# pip install logic

import itertools as it
import logpy.core as lc
from sympy.ntheory.generate import prime, isprime

# Check if the elements of x are prime
def check_prime(x):
    if lc.isvar(x):
        return lc.condeseq([(lc.eq, x, p)] for p in map(prime, it.count(1)))
    else:
        return lc.success if isprime(x) else lc.fail

# Declate the variable
x = lc.var()

# Print first 7 prime numbers
print('\nList of first 7 prime numbers:')
print(lc.run(20, x, check_prime(x)))
```

- OUTPUT

```
List of primes in the list:
{3, 11, 13, 17, 19, 23, 29}
List of first 7 prime numbers: (2, 3, 5, 7, 11, 13, 17)
```

ANALYZING GEOGRAPHY

Let's use logic programming to build a solver to analyze geography. In this problem, we will specify information about the location of various states in the US and then query the program to answer various questions based on those facts and rules. The following is a map of the US:

BUILDING A PUZZLE SOLVER

Another interesting application of logic programming is solving puzzles. We can specify the conditions of a puzzle and the program will come up with a solution. In this section, we will specify various bits and pieces of information about four people and ask for the missing piece of information. In the logic program, we specify the puzzle as follows:

- Steve has a blue car.
- The person who owns a cat lives in Canada. Matthew lives in the USA.
- The person with a black car lives in Australia.
- Jack has a cat.
- Alfred lives in Australia.
- The person who has a dog lives in France.
- Who has a rabbit?

	Pet	Car Color	Country
Steve	dog	blue	France
Jack	cat	green	Canada
Matthew	rabbit	yellow	USA
Alfred	parrot	black	Australia

The goal is the find the person who has a rabbit. Here are the full details about the four people.

THANK YOU