Austin Smothers

Professor Lu

CSC 135

PL Assignment 3

1. 
   1. 

**Deductive**

* 1. 

**Pattern Matching**

1. 
   1. 

|  |  |
| --- | --- |
| Procedural Programming   * Architecture: Von Neumann machine * Syntax: Sequence of statements * Computation: Sequential statements execution * Control: Logic and control are mixed together | Logical Programming   * Abstract Model * Logic Formulas (Horn Clauses) * Deduction of the clauses * Logic and control can be separated |

* 1. 

Resolution order control: Ordering of pattern matching during resolution

Closed world assumption: It only has the knowledge of its DB, making it a (true/fail) system instead of a (true/false) system

The negation problem: Prolog NOT operator != logical NOT operator due to Horn Clause

* 1. 

Logic is used to represent program. Deductions are used as computation. A higher level language does more automatically – we can concentrate more on what is to be done and less on how to do it. Ideal is to only specify logic and let system take care of control.

1. 

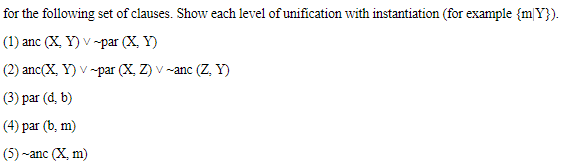
Given a set of clauses S & a goal of G,

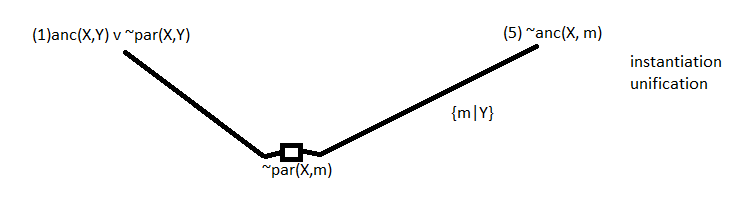
{S} U {¬G}

Existence of contradiction => derivation of empty clause

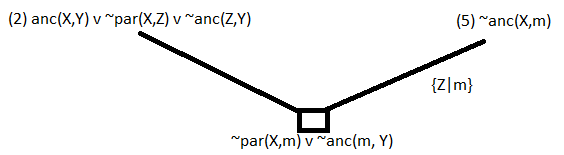
Based on {S} U {¬G} is inconsistent if {S} U {G} is consistent

1. 



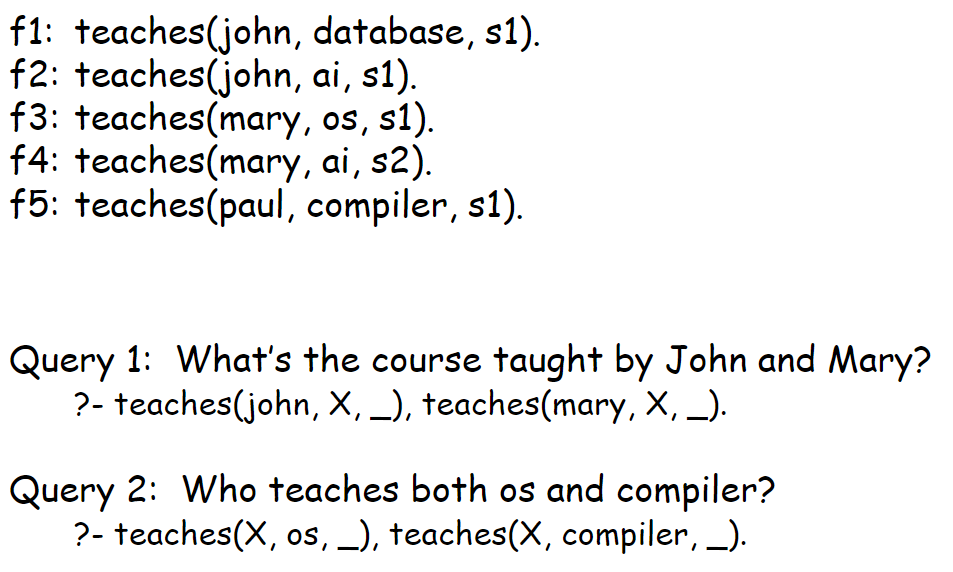
* 1. 

No contradiction

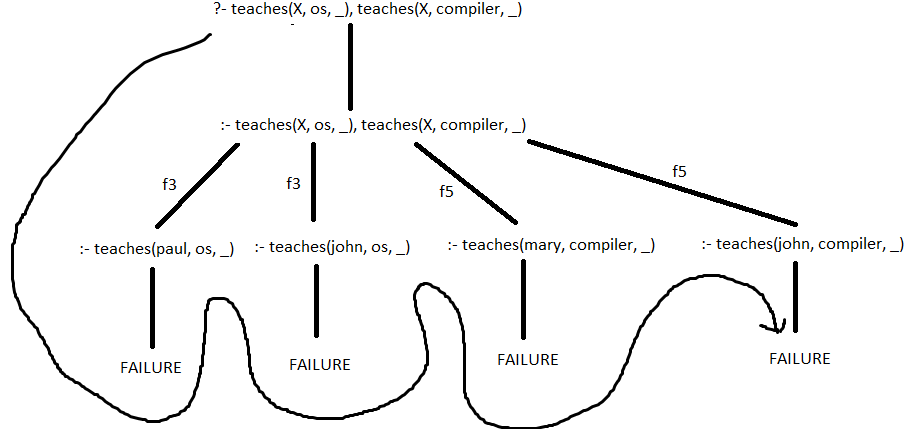
* 1. 

No contradiction

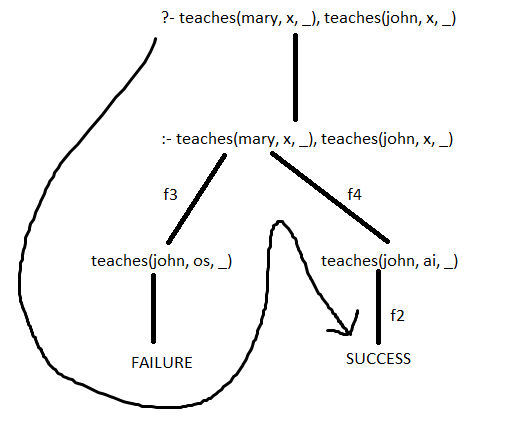
1. 

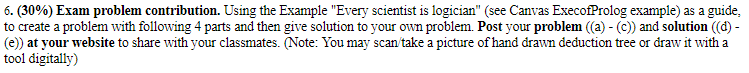


* 1. 



* 1. 



1. 
   1. 

%facts

cars('Maserati', gas, hers).

cars('Tesla Model S', electric, his).

cars('Honda Leaf', electric, noone).

cars('Toyota Avalon', gas, his).

cars('Ford Fiesta', gas, noone).

* 1. 

%rules

cool(X) :- cars(X, electric, his).

cool(X) :- cars(X, gas, hers).

isGas(X) :- cars(X, gas, \_).

isElectric(X) :- cars(X, electric, \_).

* 1. 

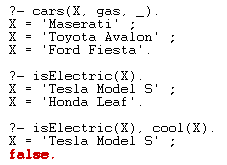
%queries

%?- cars(X, gas, \_).

%?- isElectric(X).

%?- isElectric(X), cool(X).

* 1. 



* 1. 

