# DLV notation

← (implies) becomes :-

^ (and) becomes ,

V (or) becomes ;

¬ becomes -

Example: s(X) ← p(X) ^ ¬q(X) becomes s(X) :- p(X), -q(X).

* Variables: start with upper-case letters
* Constants: start with lower-case letters or can be sequences of digits or double-quoted strings
* Predicate symbols: start with a lower case letter
* Classical negation: is -
* (Classical) literal: is an atom *a* or its negation *-a*
* NAF (negation as failure) is *not*
* Disjunction is *v*
* Disjunctive rule: a formula *a1 v … v an :- b1 , … , bk , not bk+1, … not bm* where a1,...,an, b1,...,bm are classical literals

Examples:

musician(X) :- guitaris(X).

-guitarist(bob).

:- living(X), dead(X).

capital\_of(rome, italy).

capital\_of(“Rome”, “Italy).

living(X) v dead(X) :- human(X).

An answer set is a set *I* of literals (an interpretation) such that

* *I* is **consistent**: it does not contain both *a* and *-a* for any atom *a*
* Whenever all literals in the body of a clause *r € Ground(P)*are in *I*, then at least one literal in the head of *r* is *I*
* *I* is **minimal** among all interpretations that satisfy the previous two conditions

Variables and unifiers

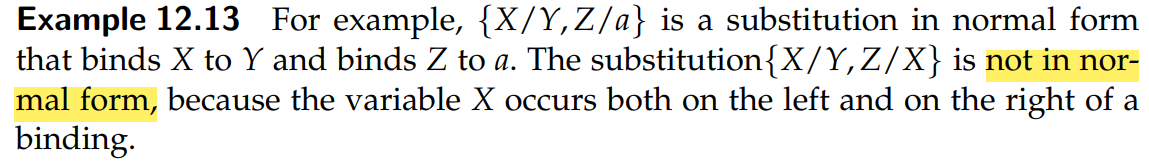
* An **instance** of an atom or a clause is obtained by uniformly substituting terms for variables
* A **substitution** is a finite set of the form { V1/t1, …, Vn/tn} where each Vi is a distinct variable and each ti is a term
* The **application** of a substitution σ = {V1/t1, … , Vn/tn} to an atom or clause *e*, written *e*σ, is the instance of *e* with every occurence of Vi replaced by ti

Substitutions:

σ1 = { X/A, Y/b, Z/C, D/e}

σ2 = {A/X, Y/b, C/Z, D/e}

σ3 = {A/V, X/V, Y/b, C/W, Z/W, D/e}



An expression is **ground**if it does not contain any variables

teaches(fred, cs342) is ground

teaches(Prof, cs342) is not ground

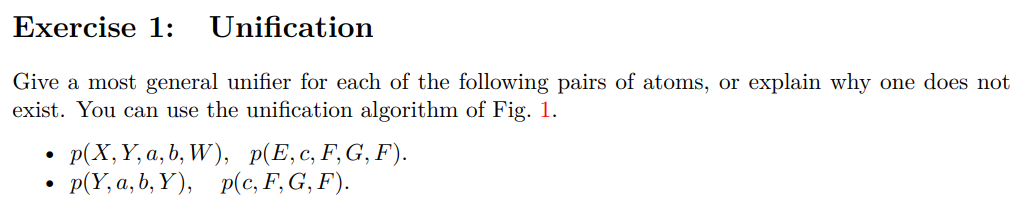
An interpretation is a triple *I* =〈 D, Φ, π 〉where,

* D is a non-empty set called the domain. Elements of D are individuals
* Φ is a mapping that assign to each constant an element of D
* π is a mapping that assign to each n-ary predicate symbols a function from Dn into {true, false}

Φ is a function from names into individuals in the world. The constant *c* is said to **denote** the individual Φ(c)

π(p) specifies whether the relation denoted by n-ary predicate symbol p is true or false for each n-tuple of individuals

# Ex.1



# Ex.2

