| Variables | Upper case letters |
| --- | --- |
| Constants | Lower case letters or sequence of digits or double quoted strings |
| Predicate symbols | Start with a lowercase letter |
| Classical negation | “ - “ |
| NAF (Negation As Failure) | “ not “ |
| Disjunction | “ v “ or “ I “ |

**REMEMBER THE DOT ( . )**

musician(X) :- guitarist(X) .

-guitarist(bob) .

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

capital\_of(rome, italy) .

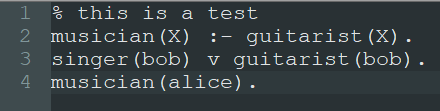
capital\_of(“Rome”, “Italy) .

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

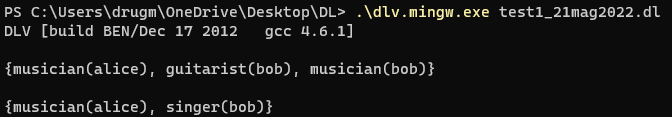
General options:

-brave perform brave reasoning

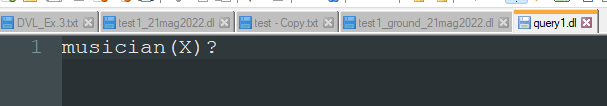
-cautious perform cautious reasoning



It returns



How to query





# ?? Exercise 1: Unification

Give a most general unifier for each of the following pairs of atoms, or explain why one does not exist. You can use the unification algorithm of Fig. 1.

* p(X, Y, a, b, W), p(E, c, F, G, F).
* p(Y, a, b, Y ), p(c, F, G, F).

# ?? Exercise 2: Bottom-up and top-down

Consider the following KB:

1. arc(a, b).
2. arc(b, a).
3. arc(b, c).
4. arc(c, d).
5. arc(d, e).
6. arc(e, c).
7. connected(X, Y) :- arc(X, Y).
8. connected(X, Z) :- arc(X, Y), connected(Y, Z).
9. strongly\_connected(X, Y) :- connected(X, Y), connected(Y, X).

Exercises:

1. Show that KB ` strongly\_connected(d, c) with the bottom-up proof procedure seen at lecture.
2. Compute the answers to the query strongly\_connected(X, c) with the top-down proof procedure seen at lecture.

strongly\_connected(d, c)

# Exercise 3: DLV - Music Knowledge Base

Utto:

musician(bocelli).

singer(eminem).

singer(drdre).

artist(daparezza).

artist(cattelan).

band(beatles).

band(dpg).

music\_work(lose\_your\_self).

music\_work(still\_dre).

subgenre(pop\_rock, rock).

subgenre(trap, rap).

genre(rap).

genre(rock).

music\_work(X):- artist\_of(Y, X).

genre(X):- music\_genre(X, Y), music\_work(Y).

genre(X):- subgenre(X,Y).

musician(X):- singer(X).

artist(X):- singer(X).

artist(X):- musician(X).

artist(X):- singer(X).

artist(X):- band(X).

transitive\_subgenre(X,Y):- subgenre(X,Z), subgenre(Z,Y).

recorded\_by(X,Y):- artist\_of(Y,X).

plays\_genre(X,Y):- artist\_of(X,Y), music\_genre(Z,Y).

io:

% ex.3 file 7

musician(bocelli).

singer(eminem).

singer(deandre).

artist(caparezza).

artist(cattelan).

band(beatles).

band(dpg).

music\_work(lose\_your\_self).

music\_work(still\_dre).

subgenre(pop\_rock, rock).

subgenre(trap, rap).

genre(rap).

genre(rock).

% X is a music work if Y is the artist of X.

music\_work(X) :- artist\_of(Y, X) .

% X is a genre if it is the genre of some music work.

genre(X) :- has\_genre(Y, X) .

% X is a genre if it is the subgenre/supergenre of some Y.

genre(X) :- has\_subgenre(X, Y) .

genre(X) :- has\_supergenre(X, Y) .

% If X is a singer then X is a musician and an artist.

musician(X) v artist(X) :- singer(X) .

% If X is a singer, a musician or a band, then X is an artist.

artist(X) :- singer(X) v musician(X) v band(X) .

% Introduce a new relation transitive\_subgenre(X, Y) that represents the transitive

% closure of the subgenre relation

% Introduce a relation recorded\_by(X, Y) as the inverse of the artist\_of(X, Y) relation.

recorded\_by(X, Y) :- artist\_of(Y, X) .

artist\_of(Y, X) :- recorded\_by(X, Y) .

% Infer artists’ genres from music genres. Define the relation plays\_genre(X, Y) that

% holds if an artist has music works of a certain genre, including all super-genres

plays\_genre(X, Y) :- artist(X), genre(Y), artist\_of(X, Z), has\_genre(Z, Y).