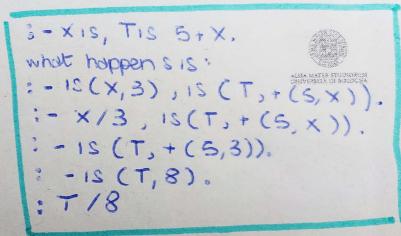
```
prod(X, 0, 0).
prod(X, s(Y), Z):- prod(X, Y, W), sum(X, W, Z).
```

· Non practicable...



## **Evaluation of expressions**

However, when we write an <u>expression</u>, we would like to have it <u>evaluated</u>...

· Special pre-defined predicat is.

T is Expr (is(T,Expr))

2020/6/12 10:51

T can be a numerical atom or a variable

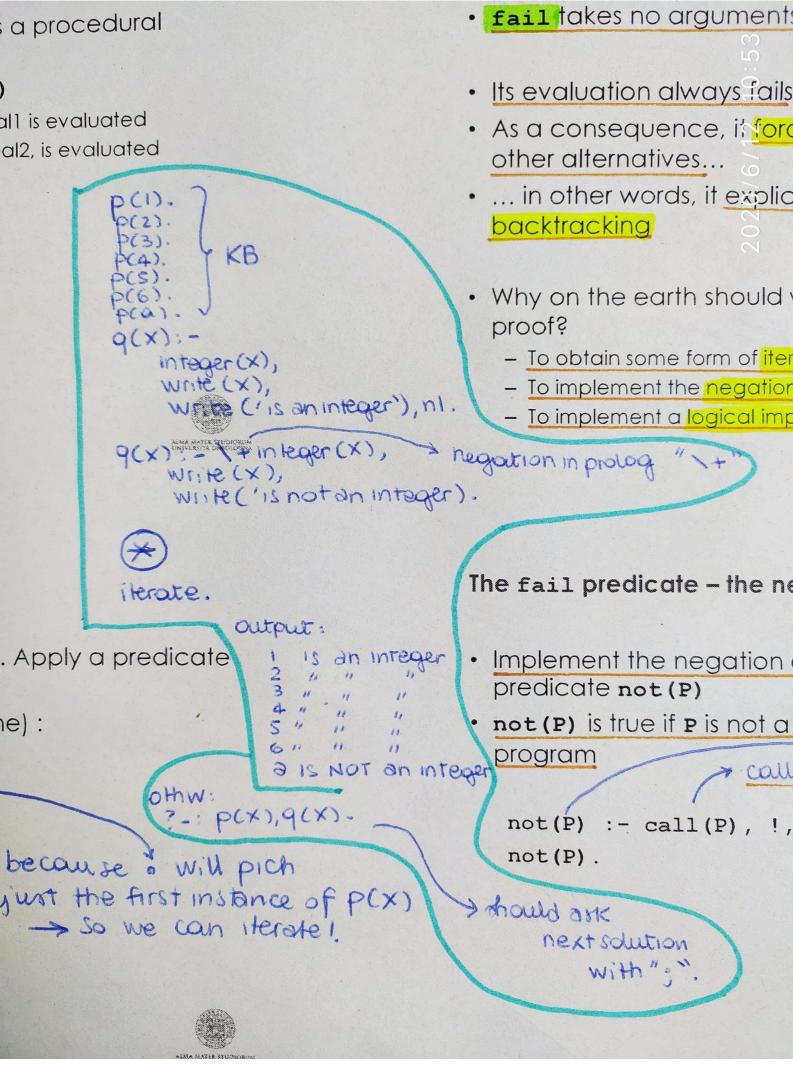
- Expr1 and Expr2 are evaluated; Mind it! They both have to be completely instantiated.
- · The results are then compared on the basis of REL

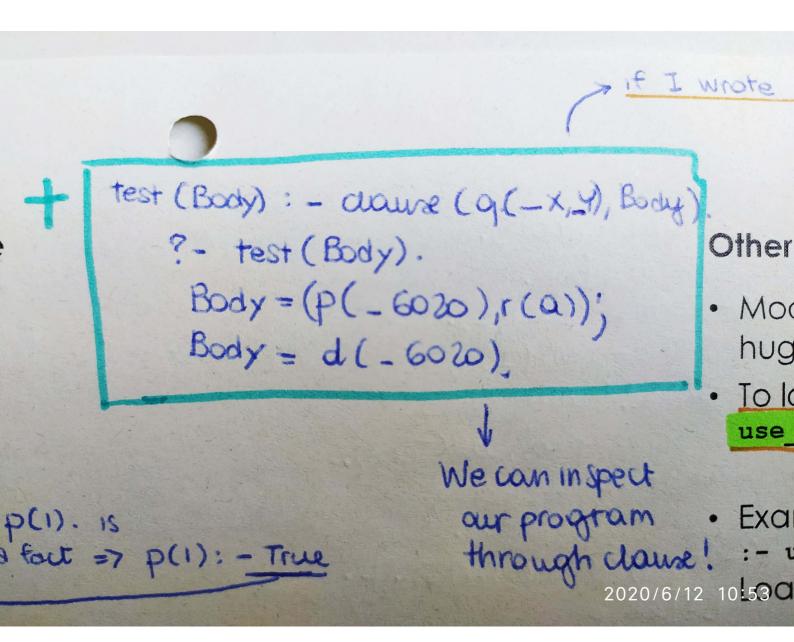
Do a program that receives N as input, and prints N times "Hello world!" > recurrive calls chesani (0.) newline chesani (N): print ('Hello world'), n NI 45 N-1, chesani (Ni). I last = D tail - recursive this works also write

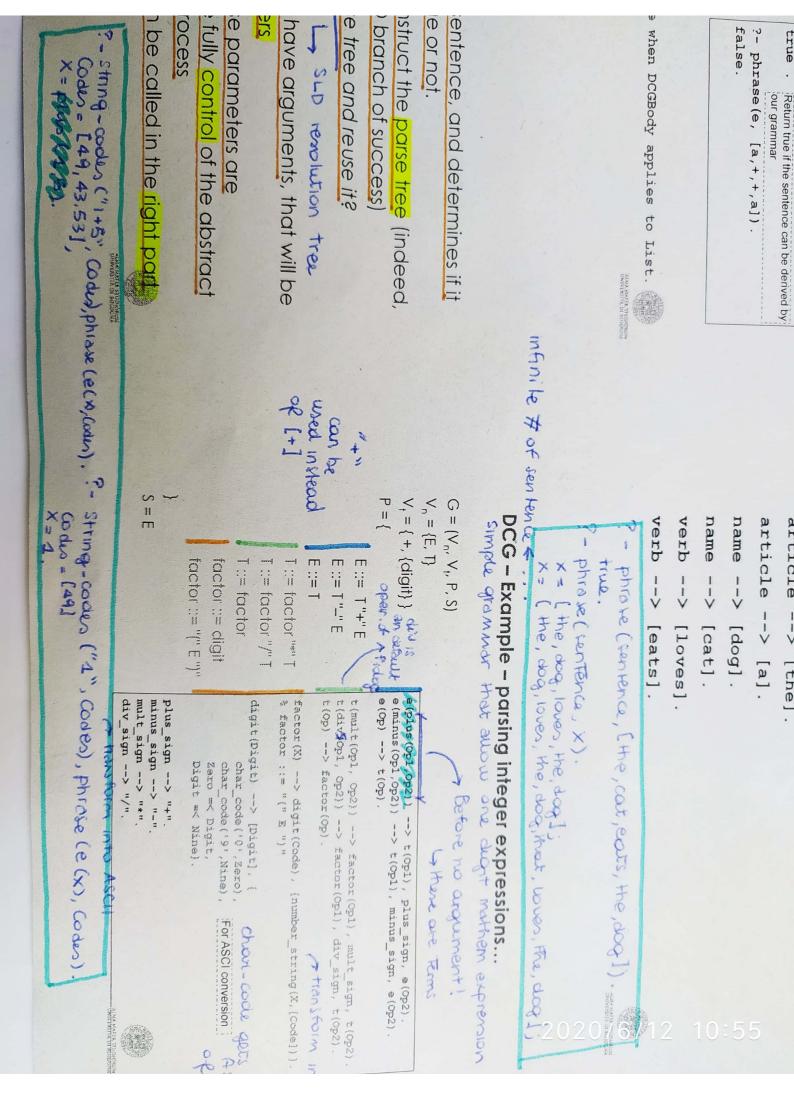
Iteration and recursion in Prolog

In Prolog there is no iteration (no while, for, repeat...)

exchanged and exploited in diff There are several pre-defined pr to deal with these structures, an JKB p(father (federico, francesco)). p (father (federico, chiara)). female (chiara) test(x):father in p(fother (X,Y)), female (Y) tather (X,Y): - Son (Y,X). son (francesco, feder, co). test\_if\_x\_is\_father (x):-P(father (x, Y)) call (son (x, y)). dere passed as a term test2(X):e wi P(X), call(X). evo grar call all the term unitied with x from, is in HIIGIPICION p(a). When exec







```
? - Term = .. [boh].
                           Accessir
    Term = boh.
? - Term = .. [boh, 1].
     Term = boh (1).
                          • Term =
? - Term = .. (father, mario,
                             [SWI do
             abbl, call (Term). - List is remo
  Term = father (morrio,
                              - Eithe
                aldo).
?-boh(e, q(f(r(1))))=...L.
                          ?- foo(}
   L=[boh, P,
                          List =
        g(fcrcinn)].
                           ?- Term
```

Term = h