

## UNIFICATION

unifies two terms

used for pattern matching and type

ex int \* x

ex int \* int

int \* bool

are not unifiable.

cinference.

are unifiable for x = book \* book

y \* ( bool \* bool)

y = int

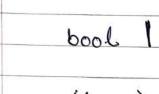


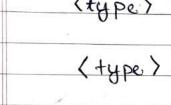
## &SUBSTICTUTION

(type) ::=

## b int 1

bloat 1

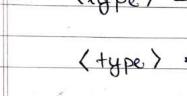


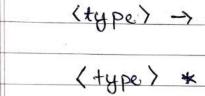


(term) :=

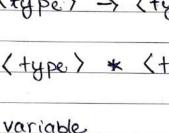
constant

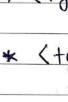
variable

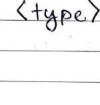




(type) -> (type) <type> \* <type> |

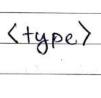




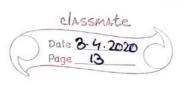




f ( (term), ..., (term))



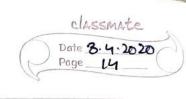




the essential task of unification is to find a substitution that makes the two terms equal.

the terms 
$$t_1$$
 and  $t_2$  are unificiable if there exists a substitution  $S$  such that  $t_1 S = t_2 S$ 

$$t_2 = (g(z), \omega)$$



MOST GENERAL UNIFIERS (MGU).

it is possible that no unifier for given two terms exist.

x / f(x) connot be unified.

there may be several unifiers.

 $t_2 = f(g(z), \omega)$ 

t, = 4 (20, g (y))

cz - 6 cyces, ws

 $S = \mathcal{E}_{x} \mapsto g(z), y \mapsto \omega, \omega \mapsto g(\omega) \dot{\mathcal{E}}$  $S' = \mathcal{E}_{x} \mapsto g(\mathcal{E}(a,b)),$ 

g +> f(b,a),

z 1-> f(a,b),

w Hg (f(b,a)) 3.



most general unifier (mgu) that is unique up to renaming.

- S is the most general unifier of t, and to

· ct is a unifier of t, t2.

· for every other unifier s' of t1, t2, there exists a refinement of s to give s'.

mgo can be efficiently computed.

 $t_1 = f(x_0, g(y)) = 1 = f(x_0)$   $t_2 = f(g(z), \omega)$ 

mgv = Extg(z), ytow, wtog(w)}

topacos' = & y in g(w) } v mgo

= &y rg(w), x rg(z), wrg(g(w))}