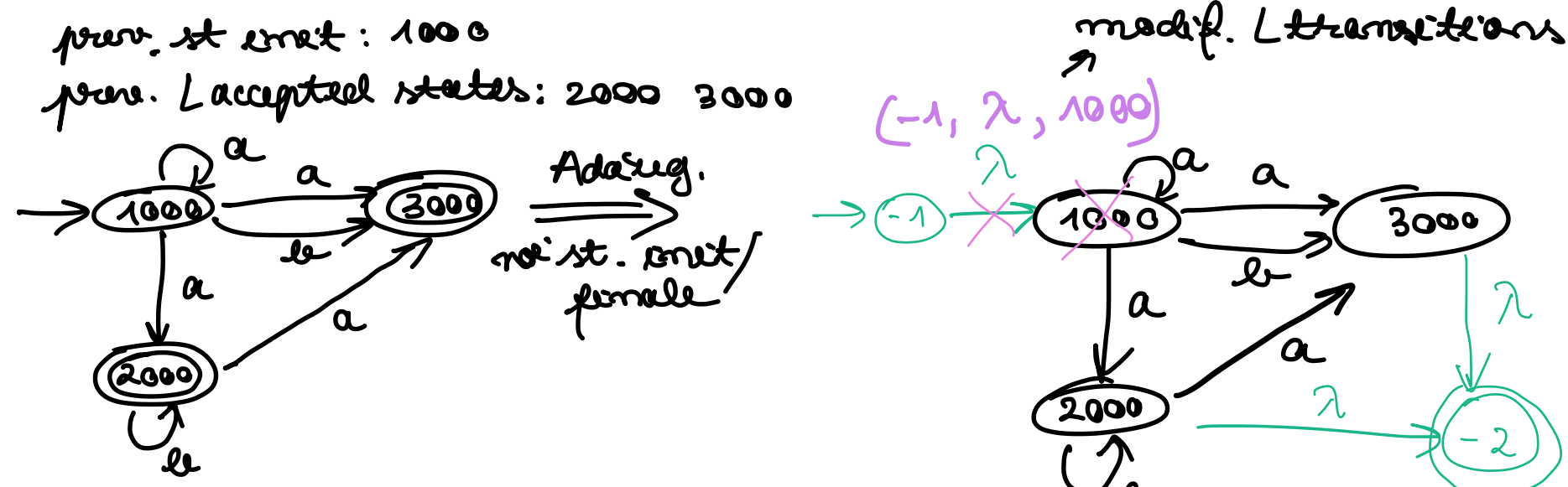# DFA to REGEX
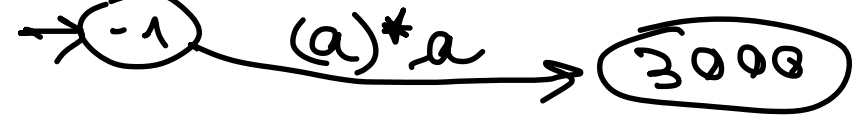
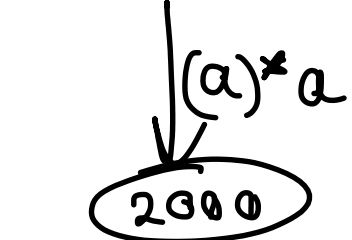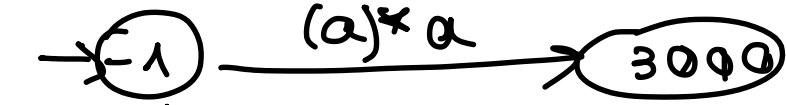Ex. din seminar (prelucrat altfel)

prev. st. init : 1000
prev. L acceptdel states: 2000 3000
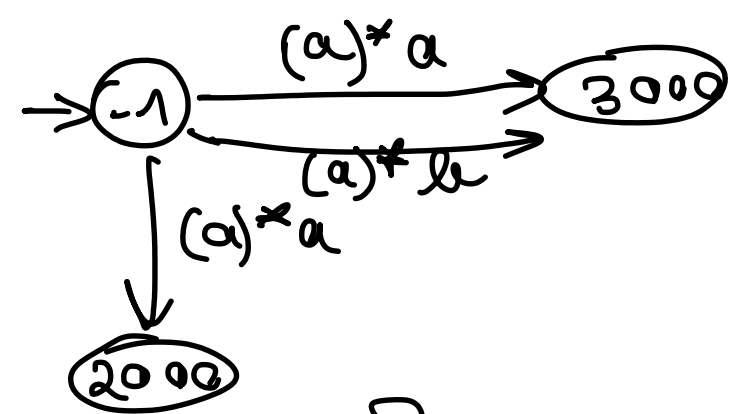

→ Adaug. noi esprii calculate

între (-1, 3000): (a)*a → 3000
între (-1, 2000): (a)*a → 3000 →(a)*a 2000
între (-1, 3000) ← încă o apariție din cauză că cele două tranziții cu simbolurile diferite
(a)*a → 3000 / (a)*b
(a)*a → 2000

Revenind la întregul automat:

---

modif. L transitions

(-1, λ, 1000)

(2000, λ, -2)
(3000, λ, -2)

Elim. stări în ordinea în care au fost enumerate în L states (fără a se considera st. init. și finale)

Eliminare 1000

exp prev, next = exp prev, curr · (exp curr, curr)* · exp curr, next

Stare curentă 1000 :
L prev : [(-1, λ)]
L self : [a]
L next : [(3000, a), (2000, a), (3000, b)]

Alg. meu modifică ÎNAINTE
L prev însă alese să șterg tranzițiile din L prev pt stările curentă verificate că să pot adăuga o nouă tranziție care va corecta prev_state cu un next_state

→ Cum arată dict - transitions după ce se adaugă noile tranziții:
{..., -1: {'(a)*a': [3000, 2000], '(a)*b': [3000]}}
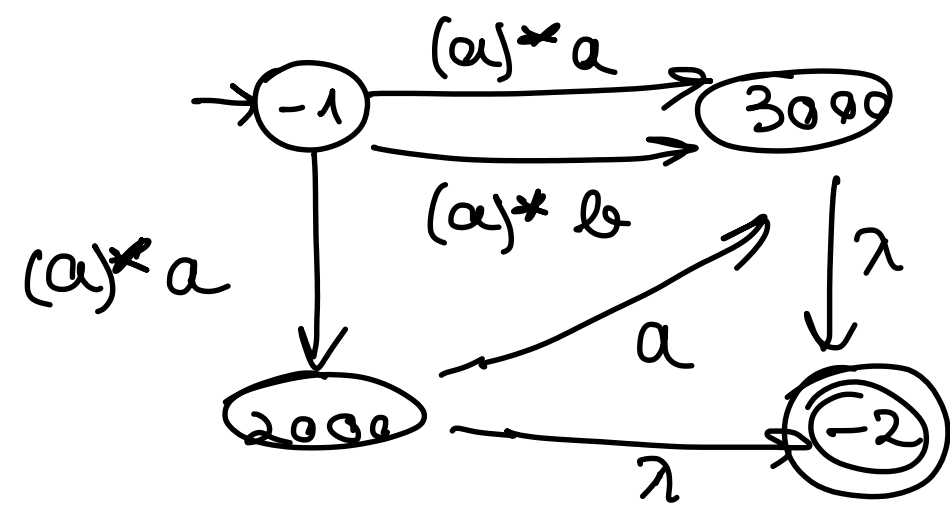
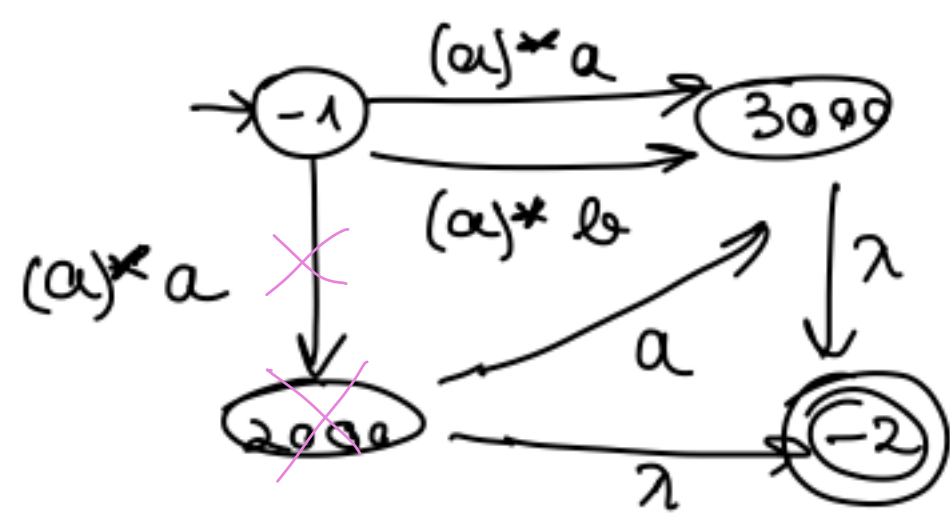Similar pt restul stărilor:

⟹ Eliminare 2000 :
L prev : [(-1, (a)*a)]
L self : [b]
L next : [(3000, a), (-2, λ)]

⟹ Adaug noi esprii calculate

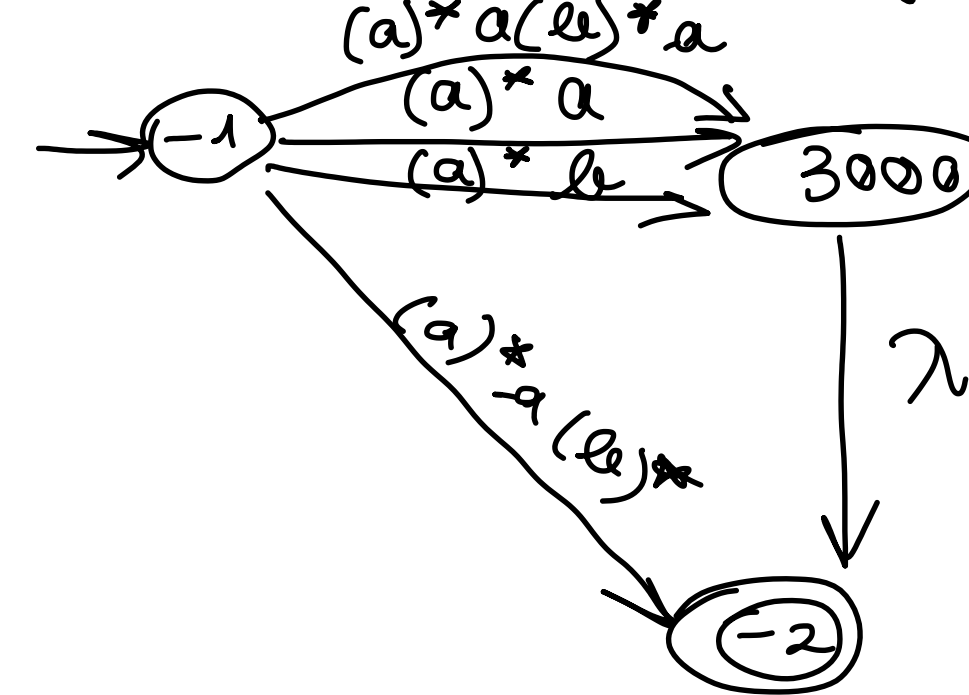între (-1, 3000):
(a)*a(b)*a → 3000
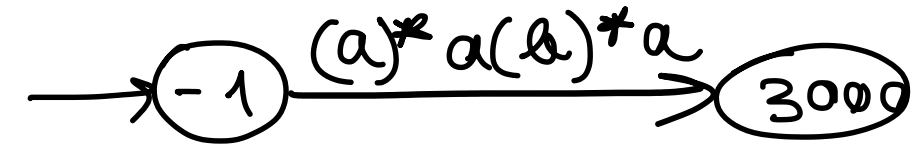
între (-1, -2):
(a)*a(b)*a → 3000
(a)*a(b)* → -2
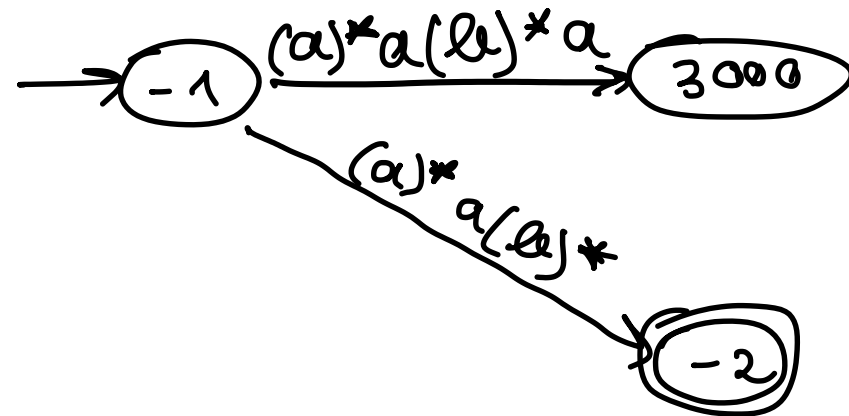
⟹ Revenind la întregul automat

⟹ Eliminare 3000 :
L prev : [(-1, (a)*a), (-1, (a)*b), (-1, (a)*a(b)*a)]
L self : []
L next : [(-2, λ)]
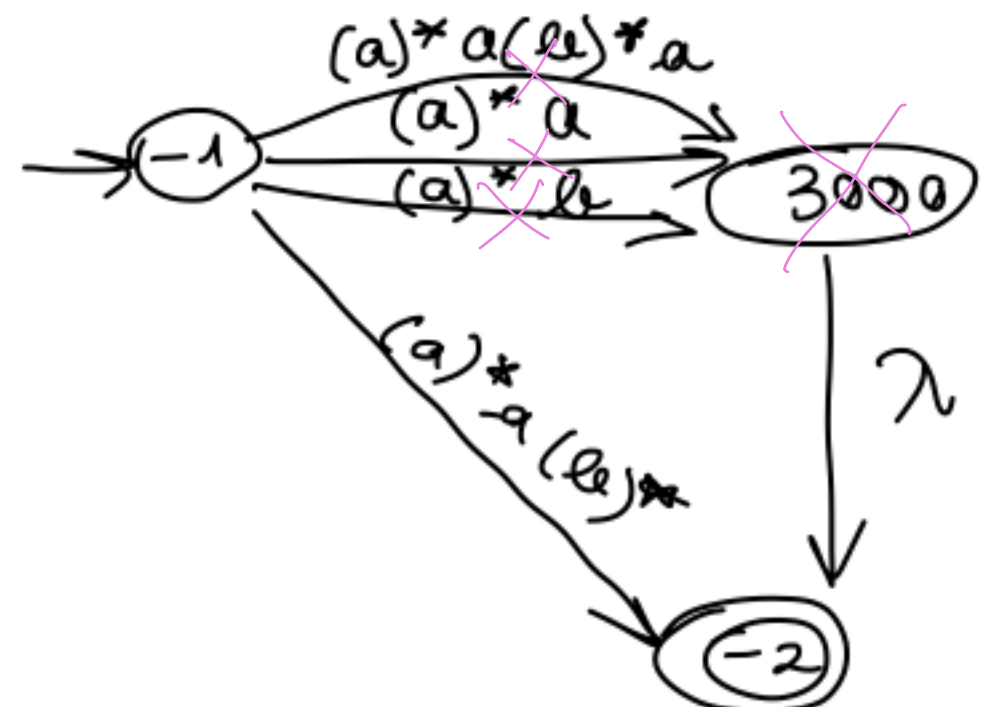
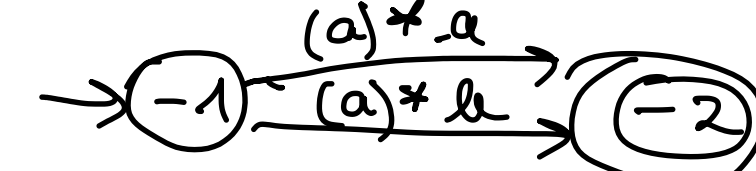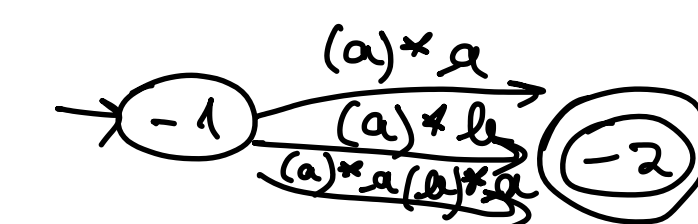⟹ Adaug noi esprii calculate

între (-1, -2):
(a)*a → -2
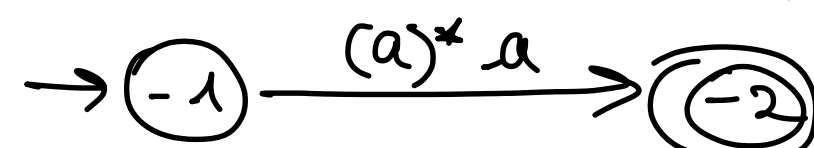
între (-1, -2):
(a)*a / (a)*b → -2

între (-1, -2):
(a)*a / (a)*b / (a)*a(b)*a → -2

⟹ Revenind la întregul automat
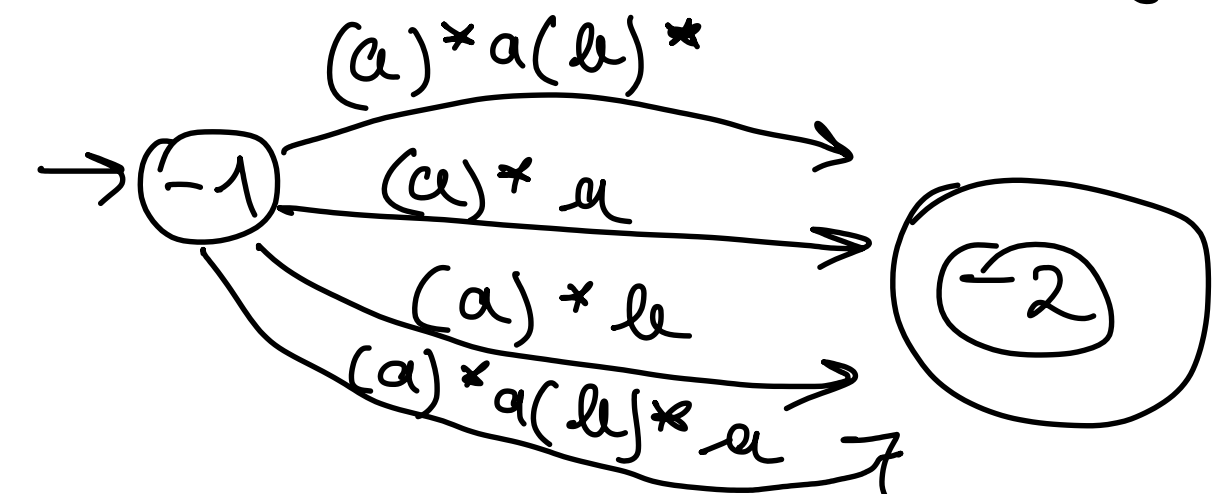
Regex-ul va fi reuniunea tuturor exp de pe muchii

regex = (a)*a(b)* | (a)*a | (a)*b | (a)* a(b)* a