Assignment - 6

Name :- Sumit Kumar Yadav   Roll No. 18CS30042

Ans 1 :-

(a) ((six equals (two plus one)) → one [] (three minus one)) plus two

((six equals three) → one [] (three minus one)) plus two

= (false → one [ ] (three minus one)) plus two

= (three minus one) plus two

= two plus two

= four

(b) (two equals (true → one [] two)) and true

= (two equals one) and true

= false and true

= false

(c) not (false) → not (true) [] not (true)

= true → not (true) [] not (true)

= not (true)

= false.

Ans 2 :- Dynamic array algebra with upper & lower bounds :-

Domain :-

Array = (Nat → A) × Nat × Nat + Error , where A is the domain with error element . return error.

Error is a unit domain used to

during an update and contains only 1
value : error array

Operations:-

new array = Nat X Nat → Array

new array = λl. λu. (λn. error , l, u)

new array represents an empty array and maps all index arguments between lower bound(l) & upper bound (u) to error.

access:- Nat X Array → A

access = λn. λ(r, l, u). r greater than u → error
[ ] n less than l → error [ ] r(n))

access checks if index lies between upper bound & lower bound if not then it returns error element.

Now, update : Nat X A X Array → Array

update = λn. λv. λ(r, l, u). n greater than u → error array
[ ] n less than l → error array
[ ] ([n → v] r, l, u)

update basically checks if index lies between upper bound and lower bound. if yes then it performs normal array update else it returns error array.

Sumit Kumar Yadav (18CS30042)

Ans 3:- Abstract Syntactic domains :-
program, operator, numeral, expr. sequence, expression,
answer, digit

abstract production rules:

program ::= Expr. sequence

expr. Sequence ::= expression | expression expr. sequence

expression ::= Numeral | M^R | clear | Expression answer
| expression operator expression

operator ::= + | − | x

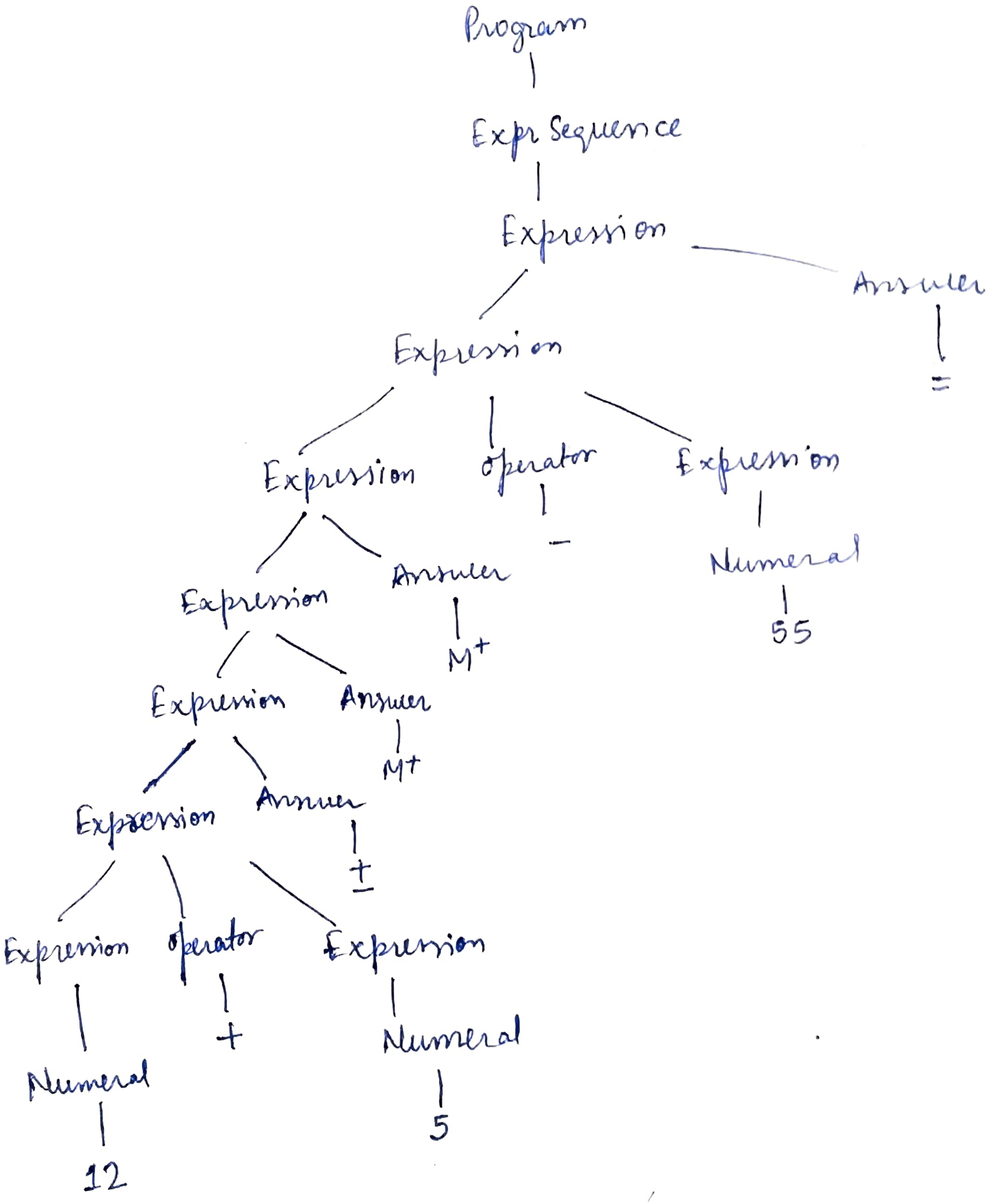
answer ::= M^+ | = | ±

numerial ::= Digit | Numeral digit

Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Sumit (18CS30042)

12 + 5 ± M^+ M^+ −55 =



Abstract Syntax Tree

Sumit (18CS30042)

Ans 4:- (a)

(i) update-payrate : Rat × Payroll-rec → Payroll-rec

update-payrate (pay, employee)
= (employee ↓1, (cases (employee ↓2) of isDay (dwage) →
inDay (pay) [] isNight (nwage) → inNight (pay) end),
employee ↓3)

(ii) update-hours : Rat × Payroll-rec → Payroll-rec

update-hours (hours, employee)
= (employee ↓1, employee ↓2, hours add rat employee ↓3)

(b) i) jdoe = newemp ("Jane Doe")
= ("game Doe", inDay (minimum-wage), 0)

(ii) jdoe-night = move-to-gnightshift (jdoe)
= (jdoe ↓1, (cases (jdoe ↓2) of is Day (dwage) →
inNight (dwage) [] is Night (nwage) → inNight (nwage) end),
jdob ↓3)

= ("Jane Doe", (cases (inDay (minimum-wge) of
~~isDay~~ isDay (dwage) → inNight (dwage) [] isNight (nwage)
→ inNight (nwage) end), 0)

= ("Jane Doe", inNight (minimum-wage), 0)

Sumit (18CS30042)

(iii) jdoe-hours = update_hours (makerat (38, 1), jdoe-night)

= (jdoe-night ↓1, jdoe-night ↓2, makerat (38, 1)
addrat jdoe-night ↓3)

= ("Jane Doe", inNight (minimum_wage), makerat (38,1)
addrat 0)

= ("Jane Doe", inNight (minimum_wage), makerat (38,1))

(iv) jdoe-pay = update_payrate (makerat (9,1), jdoe-hours)

= (jdoe_hours ↓1, (cases (jdoe-hours ↓2) of isDay (dwage)
→ inDay (makerat (9,1)) [ ] isNight (nwage) →
inNight (makerat (9,1))) end), jdoe-hours ↓3)

= ("Jane.Doe", (cases (inNight (minimum_wage)) of
isDay (dwage) → inDay (makerat (9,1)) [ ] isNight (nwage)
→ inNight (makerat (9,1)) end), makerat (38,1))

= ("Jane Doe", inNight (makerat (9,1)), makerat (38,1))

Sumit (18CS30042)

compute-pay (jdoe-pay)

= (cases jdoe-pay ↓2 of isDay (dwage) → dwage
multrat (jdoe-pay ↓3) [] is Night (nwage) → (nwage
multrat 1.5) multrat (jdoe-pay ↓3) end)

= (cases in Night (makerat (9,1)) of isDay (dwage) → dwage
multrat makerat (38,1) [] is Night (nwage) →
(nwage multrat 1.5) multrat (makerat (38,1) end)

= (in Night (makerat (9,1)) multrat 1.5) multrat makerat(38,1)