Vending Machine NFA:

q0 = {Standby**}**

Q = {Standby, Resupply, Show Menu, Clear Cart, Order, P1, P2, P3, P4, P5, Enter Quantity, Check, Confirm Order, Show Cart, Payment, Payment Failed, Payment Successful, Reduce Quantity, Eject**}**

F = {Standby**}**

Σ **= {**a, b, c, d, e, f**}**

δ (Standby, b**) = {**Resupply**}**

δ (Resupply, b**) = {**Standby**}**

δ (Standby, a**) = {**Show Menu**}**

δ (Show Menu, a**) = {**Order**}**

δ (Order, a**) = {**Product1**}**

δ (Order, b**) = {**Product2**}**

δ (Order, c**) = {**Product3**}**

δ (Order, d**) = {**Product4**}**

δ (Order, e**) = {**Product5**}**

δ (Product1, a**) = {**Enter Quantity**}**

δ (Product2, b**) = {**Enter Quantity**}**

δ (Product3, c**) = {**Enter Quantity**}**

δ (Product4, d**) = {**Enter Quantity**}**

δ (Product5, e**) = {**Enter Quantity**}**

δ (Enter Quantity, a**) = {**Check**}**

δ (Enter Quantity, d**) = {**Clear Cart**}**

δ (Check, a**) = {**Confirm Order**}**

δ (Check, d**) = {**Clear Cart**}**

δ (Confirm Order, b**) = {**Order**}**

δ (Confirm Order, d**) = {**Clear Cart**}**

δ (Confirm Order, a**) = {**Show Cart**}**

δ (Show Cart, b**) = {**Order**}**

δ (Show Cart, d**) = {**Clear Cart**}**

δ (Show Cart, a**) = {**Payment**}**

δ (Payment, a**) = {**Payment Successful**}**

δ (Payment, e**) = {**Payment Failed**}**

δ (Payment Failed, d**) = {**Clear Cart **}**

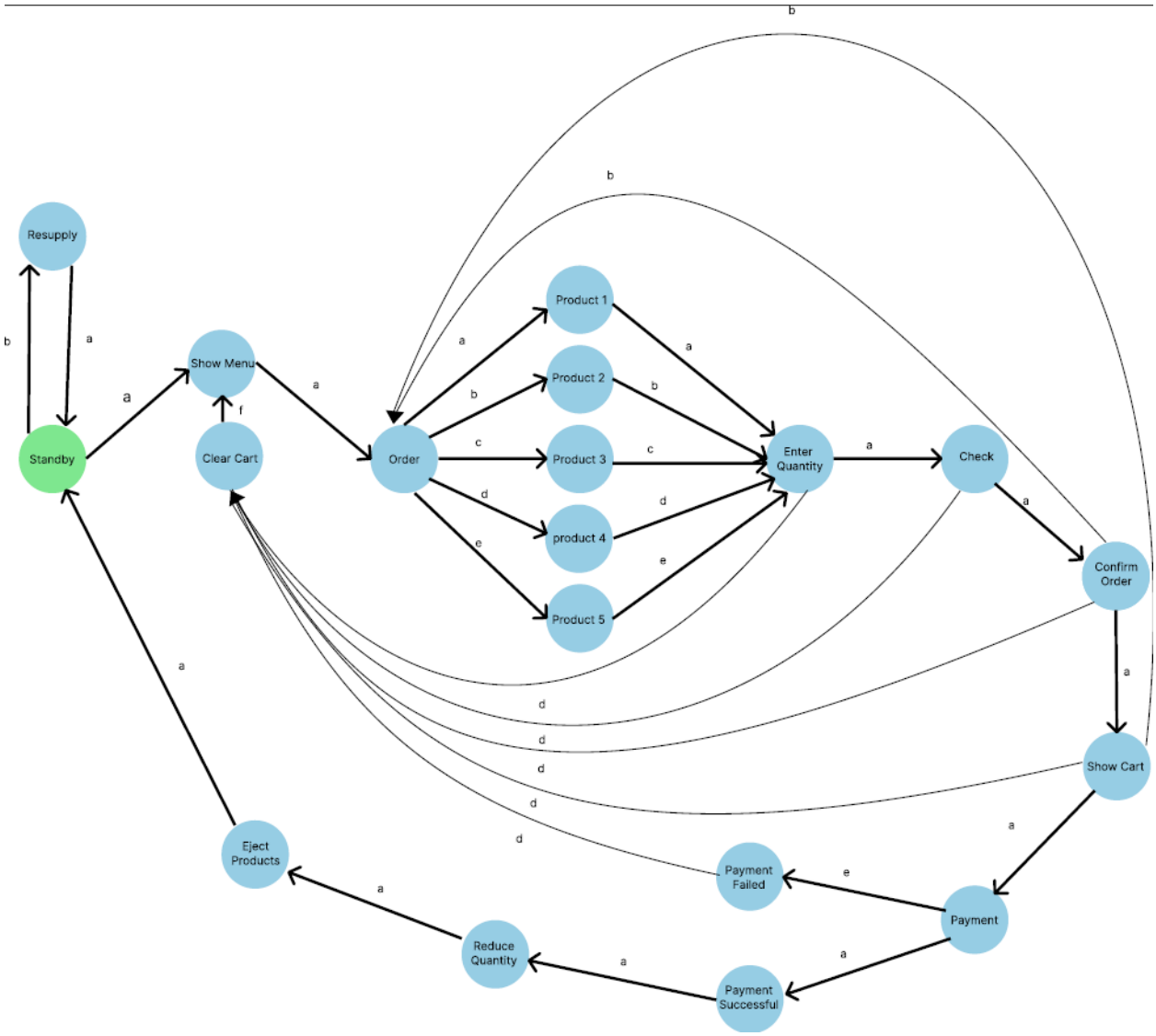
δ (Clear Cart, f**) = {**Order**}**

δ (Payment Successful, a**) = {**Reduce Quantity**}**

δ (Reduce Quantity, a**) = {**Eject**}**

δ (Eject, a**) = {**Standby**}**

Non-Deterministic Finite Automata



Code Explanation (vending.py):

There is a list of states (Q) and a string generated by action from NFA. Starting from **standby state**, options are show to user and based on options chosen by the user, string will up-dated and current state is altered. Any input string that there is no transition for it, results HALT. At any moment, input 0 prints the string generated by NFA based on action taken so far. Also, after input 0, Code checks that if it is in an Accept State or not. Then code prints result string and its condition (Accept/Not Accept).

Code Explanation (Scheck.py):

In this code, check function gets an input string, because all of NFA's transition are length 1, we iterate through characters of given string. When reading input finishes, done function is called to show the result (judging based on index of states in list). There are three results:

1. **String Accepted:** if string ends on a accept state.
2. **String Not Accepted:** if string ends on a non-accept state.
3. **HALT:** if there is no transition from specific state with given input.

Right Linear Grammar Generated by NFA:

S → aA (S: Standby / A: Show Menu)

S → bZ (Z: Resupply)

Z → aS

S →lambda

A → aB (B: Order)

B → aC (C: Order Product1)

C → aH (H: Enter Quantity)

B → bD (D: Order Product2)

D → bH

B → cE (E: Order Product3)

E → cH

B → dF (F: Order Product4)

F → dH

B → eG (G: Order Product5)

G → eH

H → dK (K: Clear Cart)

K → fA

H → aI (I: Check)

I → dK

I → aJ (J: Confirm Order)

J → bB

J → dK

J → aL (L: Show Cart)

L → dK

L → bB

L → aP

P → eM (M: Payment Failed)

M → dK

P → eN (N: Payment Successful)

N → aQ (Q: Reduce Quantity)

Q → aW (W: Eject Product)

W → aS

Regular Expression of NFA (With One Product)

(Product 2 to 5 not in grammar)

(b(ab)*aaa+aa)aad(faaad)*faaaa+(b(ab)*aaa+aa)aaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae(d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae)*d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae+(b(ab)*aaa+aa)aad(faaad)*faaaa+(b(ab)*aaa+aa)aaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)aeaa((ab(ab)*aaa+aaa)aad(faaad)*faaaa+(ab(ab)*aaa+aaa)aaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae(d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae)*d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae+(ab(ab)*aaa+aaa)aad(faaad)*faaaa+(ab(ab)*aaa+aaa)aaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*aeaa)*(ab(ab)*aλ+aλ)+b(ab)*aλ+λ

A Bit Simplified Regular Expression

((b(ab)*aa+a)a)(a(a((d(faaad)*faaaa+a(d(faaad)*faaaa)*a((((((baad+d)(faaad)*fa+b)a)a)a)(d(faaad)*faaaa)*a)*a(((((baad+d)(faaad)*fa+b)a)a)a)(d(faaad)*faaaa)*a((((((baad+d)(faaad)*fa+b)a)a)a)(d(faaad)*faaaa)*a)*a)*ae(d(faaad)*faaaa(d(faaad)*faaaa)*a((((((baad+d)(faaad)*fa+b)a)a)a)(d(faaad)*faaaa)*a)*a((((((baad+d)(faaad)*fa+b)a)a)a)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae)*d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae+a(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*aeaa((a(b(ab)*aaa+aa))(a(a((d(faaad)*faaaa+a(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae(d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae)*d(faaad)*faaaa(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*ae)+a(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a(((baad+d)(faaad)*faaaa+baaa)(d(faaad)*faaaa)*a)*a)*aeaa))))*(a(b(ab)*aλ+λ)))))+b(ab)*aλ+λ

Test Cases #1 (User Interface)

After ordering some products and canceling

Out:

Not Accepted baaaddaaaaaaaaeedf

A successful Payment after buying product

Out:

Accepted aaccaabeeaaaaaa

No Transition with given input

program halted

Test Cases #2 (String Check)

Input String is Accepted (aaccaabeeaaaaaa**)**

Accepted

Input String Not Accepted: (aaa)

Not Accepted

Input String Causes HALT: (aax)

program halted, Not Accepted