If x is an action and P a process then (x -> P) describes a process that initially engages in the action x and behaves exactly as described by P.

**Design in FSP a switch.**

Alphabet: {on, off}

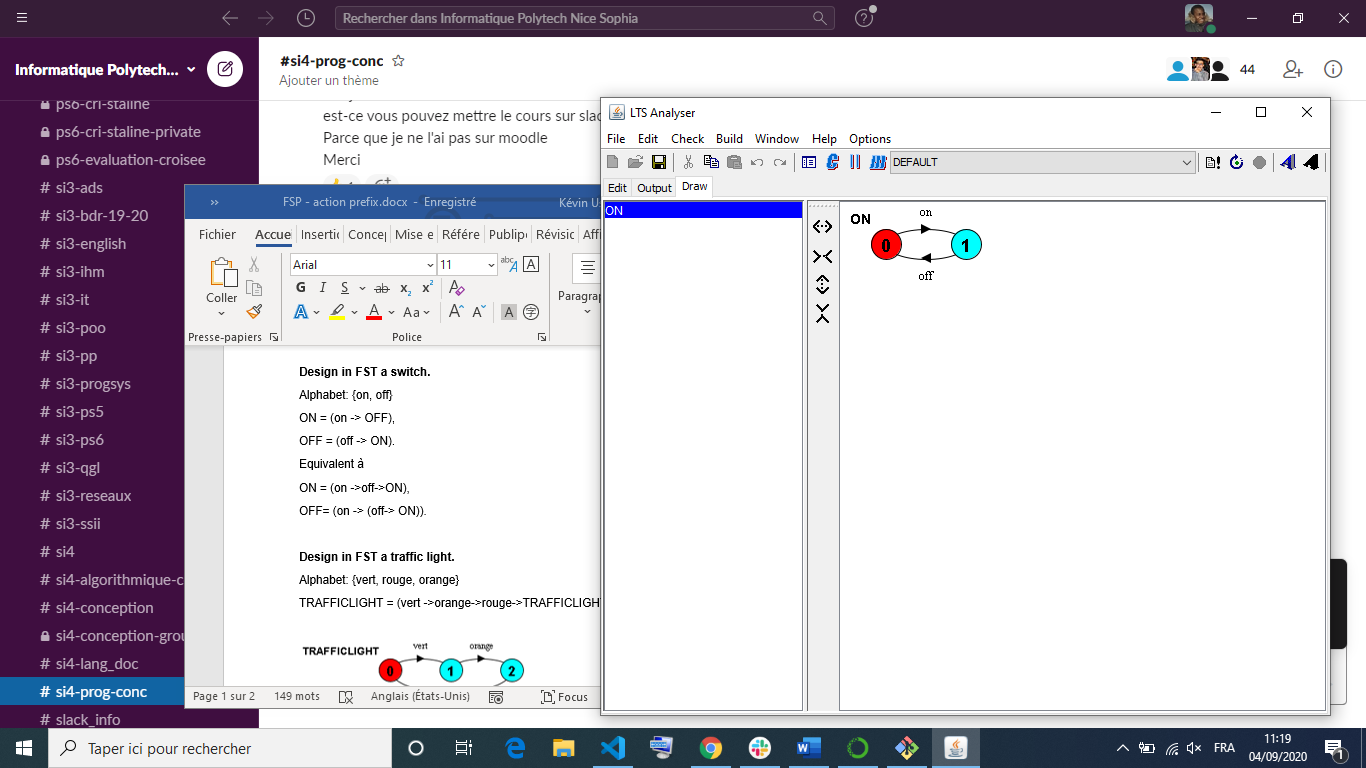
ON = (on -> OFF),

OFF = (off -> ON).

Equivalent à

ON = (on ->off->ON),

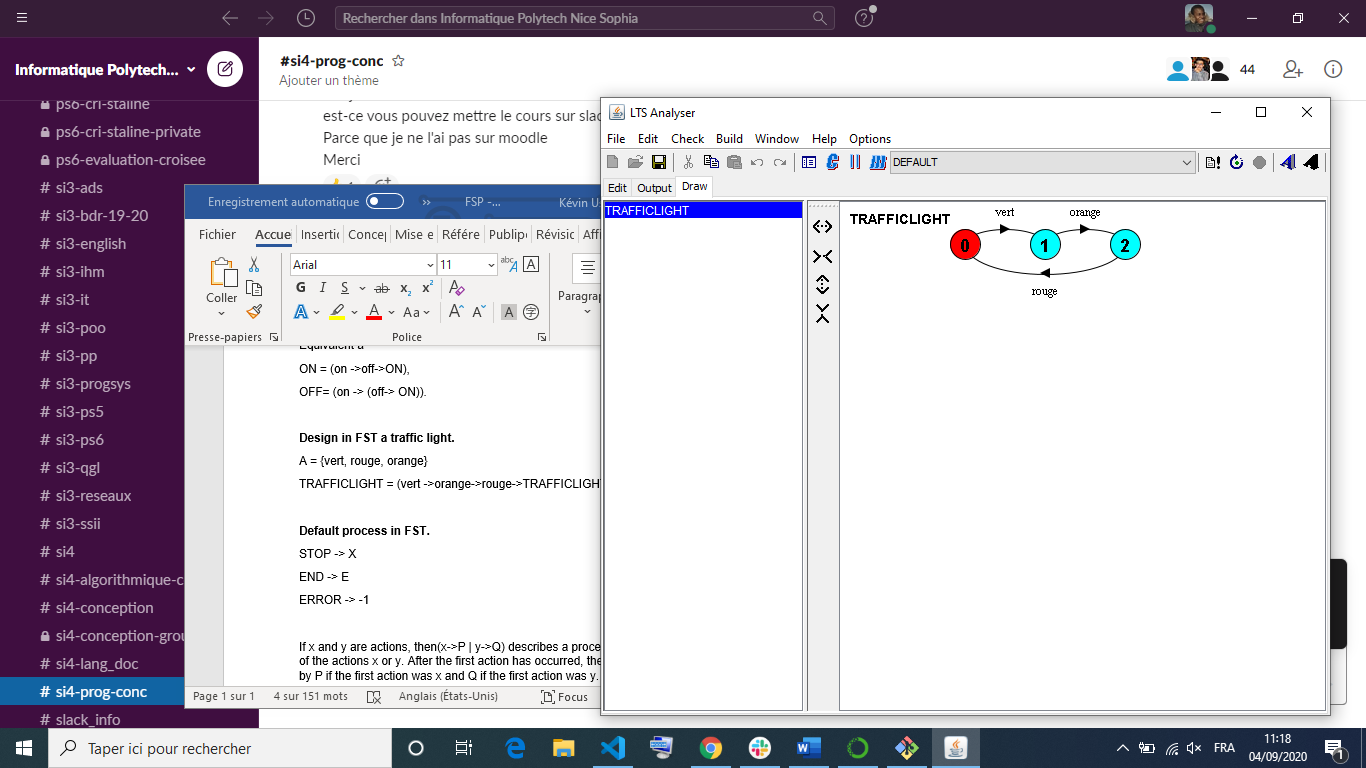
OFF= (on -> (off-> ON)).



**Design in FSP a traffic light.**

Alphabet: {vert, rouge, orange}

TRAFFICLIGHT = (vert ->orange->rouge->TRAFFICLIGHT).



**Default process in FSP.**

STOP -> X

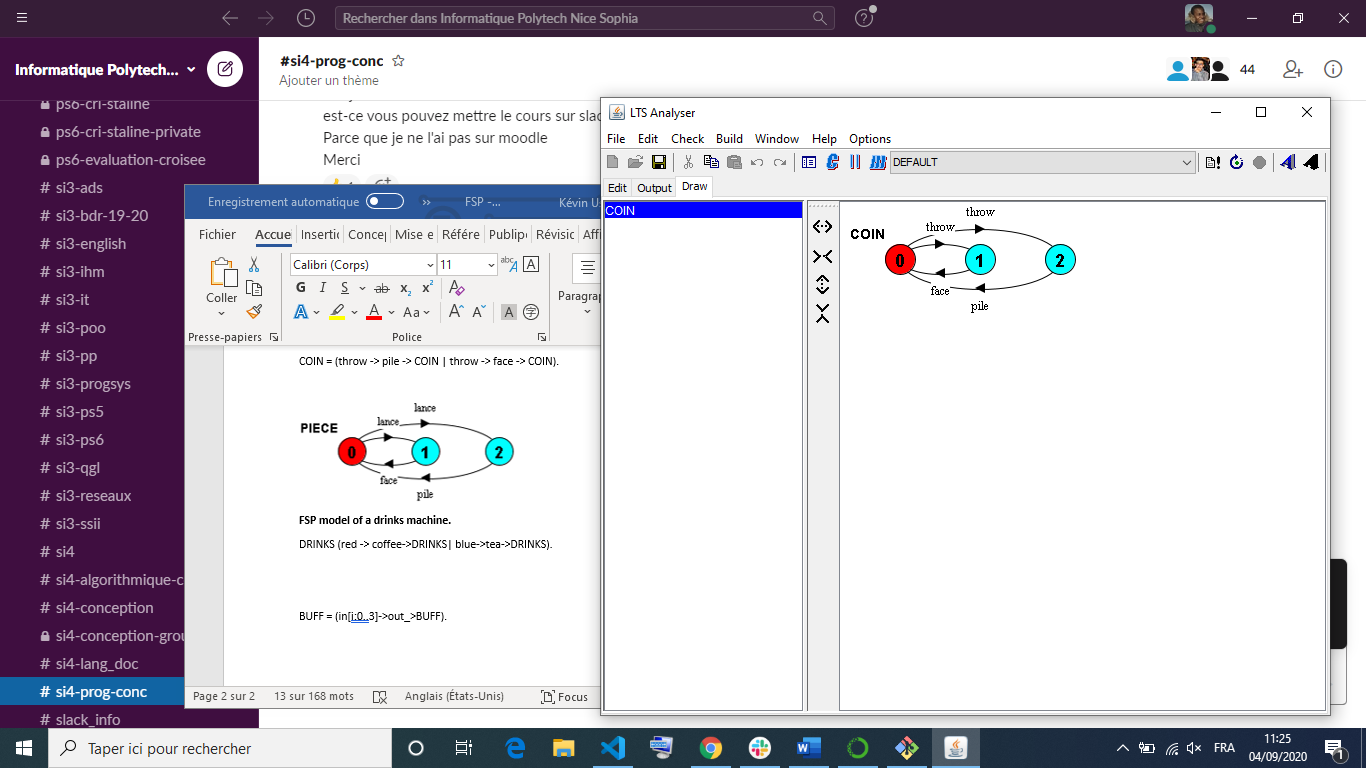
END -> E

ERROR -> -1

If x and y are actions, then(x->P | y->Q) describes a process which initially engages in either of the actions x or y. After the first action has occurred, the subsequent behavior is described by P if the first action was x and Q if the first action was y.

**FSP model of a coin**.

COIN = (throw -> pile -> COIN | throw -> face -> COIN).



**FSP model of a drinks machine.**

DRINKS (red -> coffee->DRINKS| blue->tea->DRINKS).

Single shot buffer that inputs a value in the range 0 to 3 and the outputs that value:

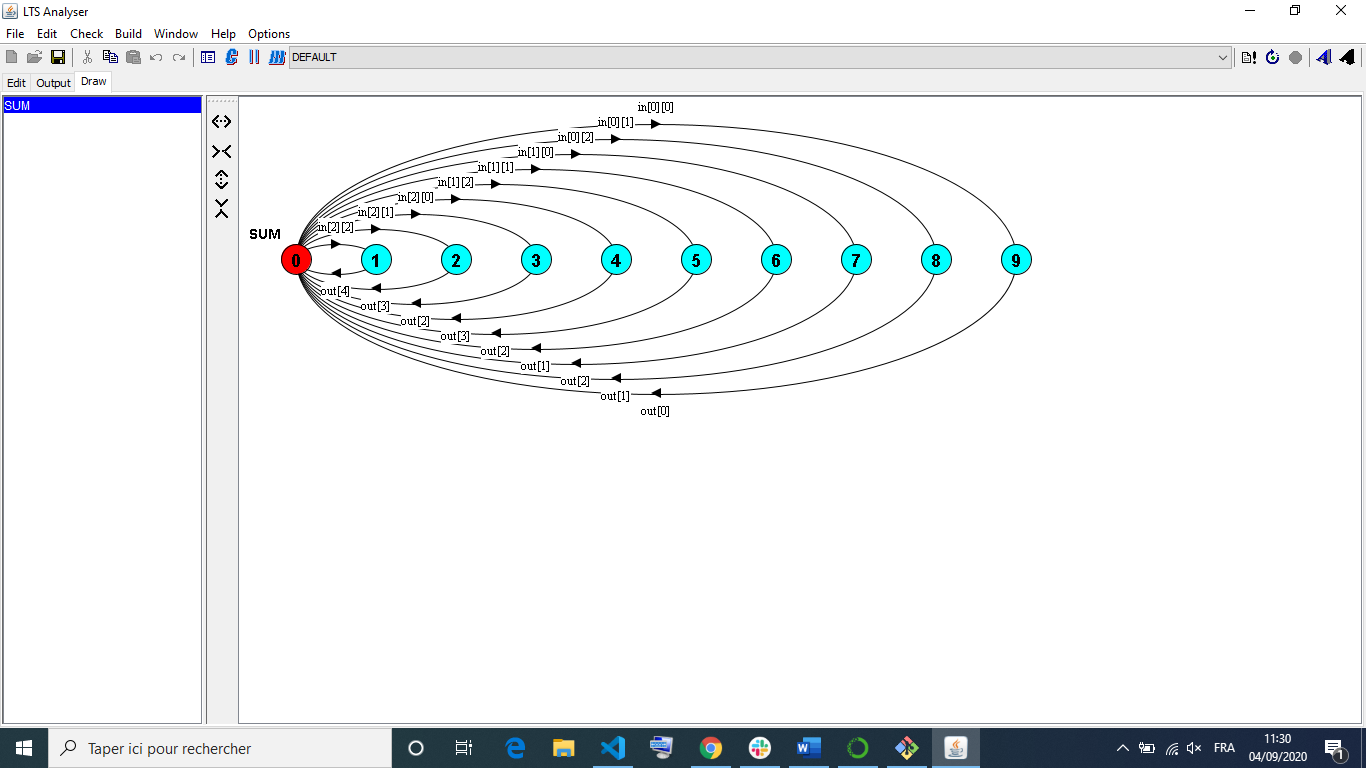
BUFF = (in[i:0..3]->out\_>BUFF).

Equivalent à

BUFF = (in[0]->out[0]->BUFF | in[1]->out[1]->BUFF | in[2]->out[2]->BUFF | in[3]->out[3]->BUFF ).

**FSP model of a sum.**

SUM = (in[a:0..2][b:0..2] -> out[a+b] -> SUM).



Equivalent à

const N=2

SUM = (in[a:0..N][b:0..N] -> out[a+b] -> SUM).

Equivalent à

const N= 2

range T = 0..N

range R =0..2\*N

SUM (in[a:T][b:T]->TOTAL[a+b]),

TOTAL[s:R] = (out[s]->PROD).

**FSP model of a multiplicator.**

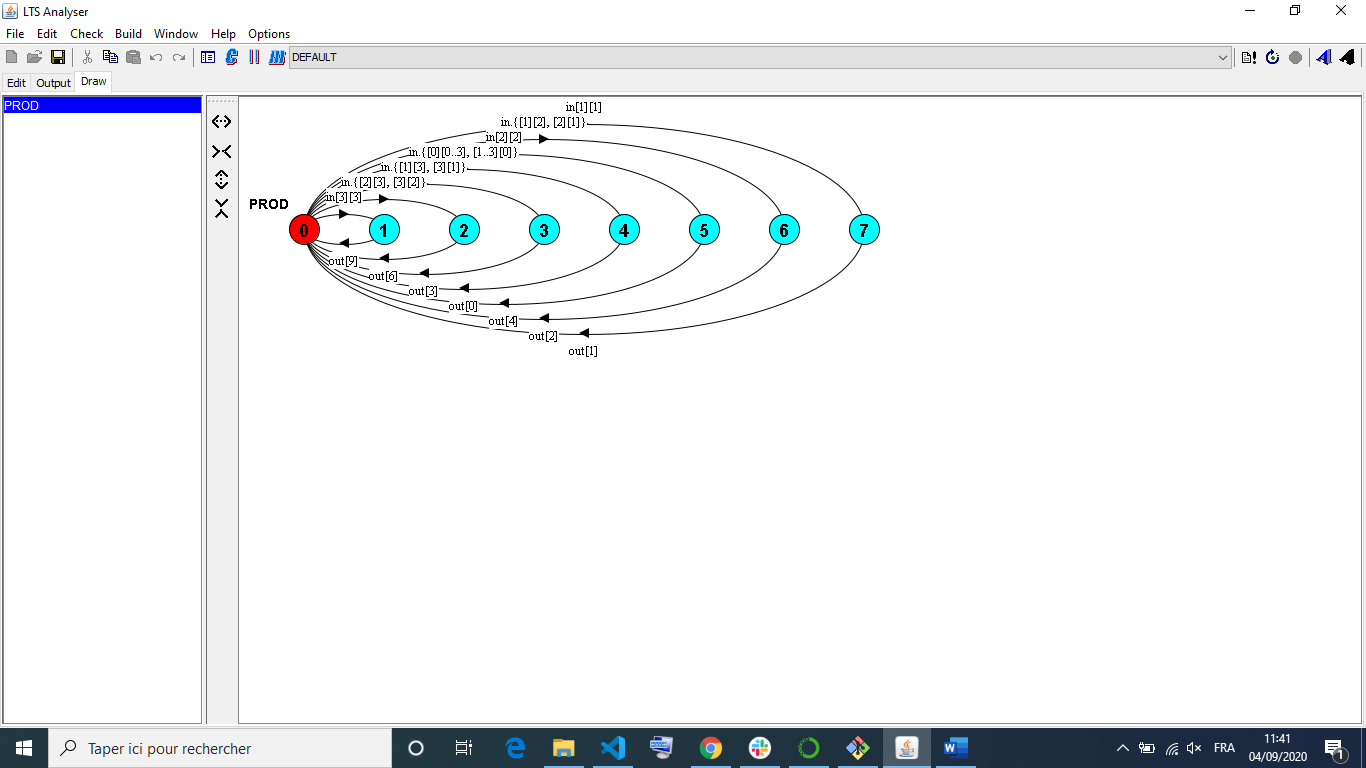
const N= 3

range T = 0..N

range R = 0..N\*N

PROD =(in[a:T][b:T]->TOTAL[a\*b]),

TOTAL[s:R] = (out[s]->PROD).



The choice (when B x->P | y->Q) means that when B is true then the actions x and y are both eligible to be chosen, otherwise if B is false then the action x cannot be chosen.

COUNT (N=3) = COUNT[0],

COUNT[i:0..N] = (when(i<N) inc->COUNT[i+1]

|when(i>0) dec->COUNT[i-1]

).

**FSP coffee machine.**

MACHINE1 = METTRE30,

METTRE30 = (mettre10 -> METTRE20

|mettre20 -> METTRE10

|mettre50 -> RENDRE20),

METTRE20 = (mettre10 -> METTRE10

|mettre20 -> CHOISIRBOISSON

|mettre50 -> RENDRE30),

METTRE10 = (mettre10 -> CHOISIRBOISSON

|mettre20 -> RENDRE10

|mettre50 -> RENDRE40),

RENDRE10 = (rendre10 -> CHOISIRBOISSON),

RENDRE20 = (rendre20 -> CHOISIRBOISSON

| rendre10 -> RENDRE10),

RENDRE30 = (rendre10 -> RENDRE20

| rendre20 -> RENDRE10),

RENDRE40 = (rendre10 -> RENDRE30

|rendre20 -> RENDRE20),

CHOISIRBOISSON = (the -> MACHINE1

|cafe -> MACHINE1

|chocolat -> MACHINE1).

MACHINE2 = M[0],

M[i:0..7] = (when (i<3) mettre10 -> M[i+1]

| when (i<3) mettre20 -> M[i+2]

| when (i<3) mettre50 -> M[i+5]

| when (i>3) rendre10 -> M[i-1]

| when (i>4) rendre20 -> M[i-2]

| when (i==3) the -> MACHINE2

| when (i==3) cafe -> MACHINE2

| when (i==3) chocolat -> MACHINE2).