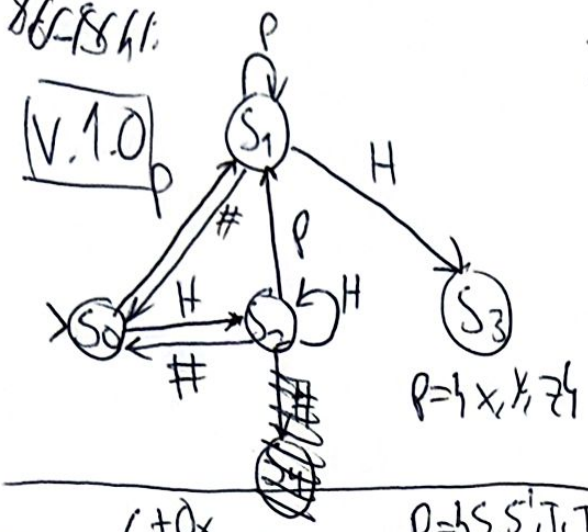
Sección:

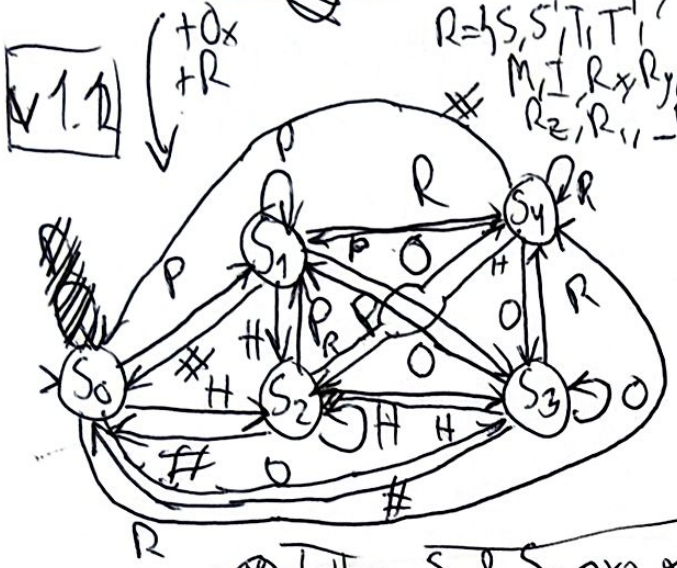
V.1.0



$\Sigma = \{P, H, \#\}$

$Q = \{$ S_0: Initial state. / New "Empty" Column
S_1: Initialization Pat. found,
S_2: Potential Superposition Pat.
S_3: Potential Entanglement Pat.
~~S_4: Superposition Pat. Found~~ $\}$

V.1.1

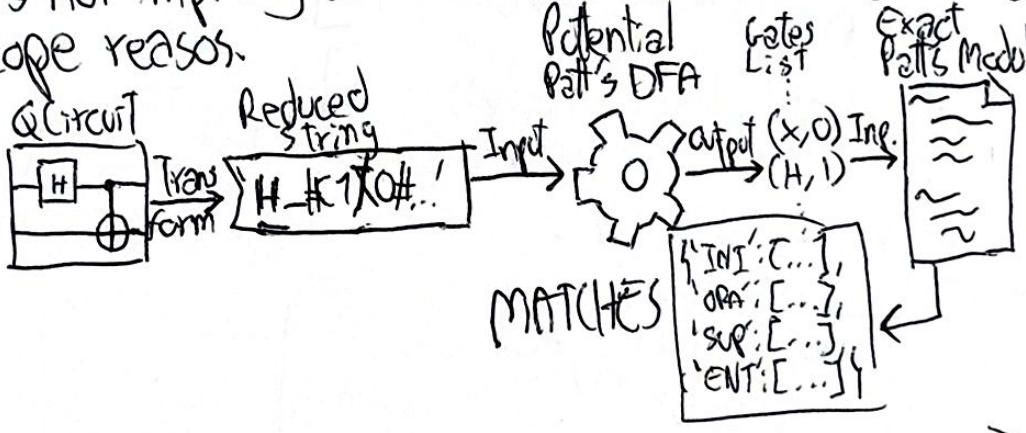


$R = \{S, S^\dagger, T, T^\dagger, M, I, R_x, R_y, R_z, R_1, -\}$

$\Sigma = \{P, H, \#, R, O\}$

$Q = \{$ S_0: New Column,
S_1: 'INI' potentially found, ⊛
S_2: 'SUP' potentially found, ⊛
S_3: 'CRA' found,
S_4: Remaining gate found [Neutral] $\}$

⊛ When S_1 & S_2 are reached, a method for checking whether there exists an exact match should be ran.
It's not implicitly described in the automaton (DFA) due to scope reasons.



	H	P	R	#	$	O
S_0	S_2	S_1	S_4	ERR	S_5	S_3
S_1	S_2	S_1	S_4	S_0	S_5	S_3
S_2	S_2	S_1	S_4	S_0	S_5	S_3
S_3	S_2	S_1	S_4	S_0	S_5	S_3
S_4	S_2	S_1	S_4	S_0	S_5	S_3

Tabla de Transiciones

S_5 ≡ ESTADO FINAL
↳ No codificado, sim. termina la ejecu. de forma correcta

State Definition

S_i

```
def _s_i(self):
    #print#

    # State task
    self.checker.check_wtf(...).

    #Move for next state (gate)
    self.gate = self._next()

    # Transition
    switch(self.gate):
        case 0:
        :
        case N:
```

V0

!!! One gate is skipped. switch checks +1.

S_i

```
def _s_i(self):
    #print
    print(f'state S_i -> Gate {}')..

    #State task (if needed).
    self.checker.whatever(...).

    # Transition
    switch(self.gate):
        case 0:
        :
        case j:
            self._s_j() <- return..
        :
        case N:
            Block
```

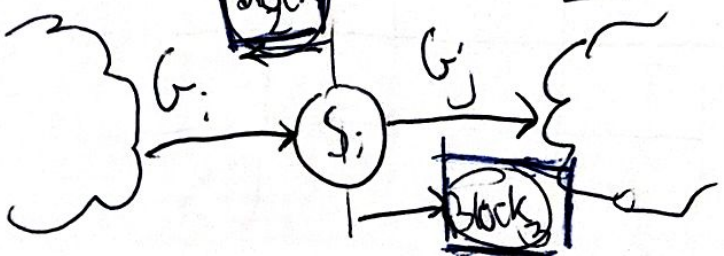
Block 1 (optional) *

Block 2 (Depends on i).

Block 1 (Optional) *

self.gate=self._next()

Block 3 (Compulsory)



* Depends on the State Task

Next Char Function

...#XYZ#_02_#... ← Input String (compressed circuit).

self.pos. ↑ (+1) => Next method

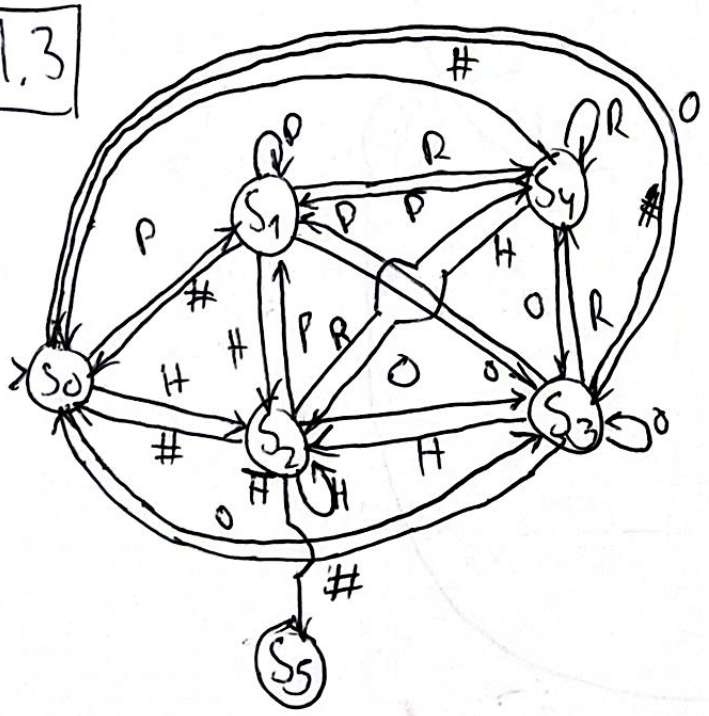
(*) self.real_pos goes apart.

```
def _next(self):
    self.pos += 1
    try:
        next_char = self.circuit[self.pos]
    except IndexError:  #End of circuit assumed in error case.
        return '$'

    #specific cases.
    if next_char == '0': #Grade length adding.
    elif next_char == '#': #Next column jump.
        ...

    return next_char.
```

V1.3



$\Sigma = \{P, H, \#, R, 0, \top, C\}$

$Q = \{$ S_0: New column
S_1: 'INI' potentially found.
S_2: 'SUP' —"— .
S_3: 'ORA' —"— .
S_4: Remaining gate found.
S_5: 1st part of 'ENT' found. [Neutral]

$\overline{H}$ ≡ There is not a uniform superposition.

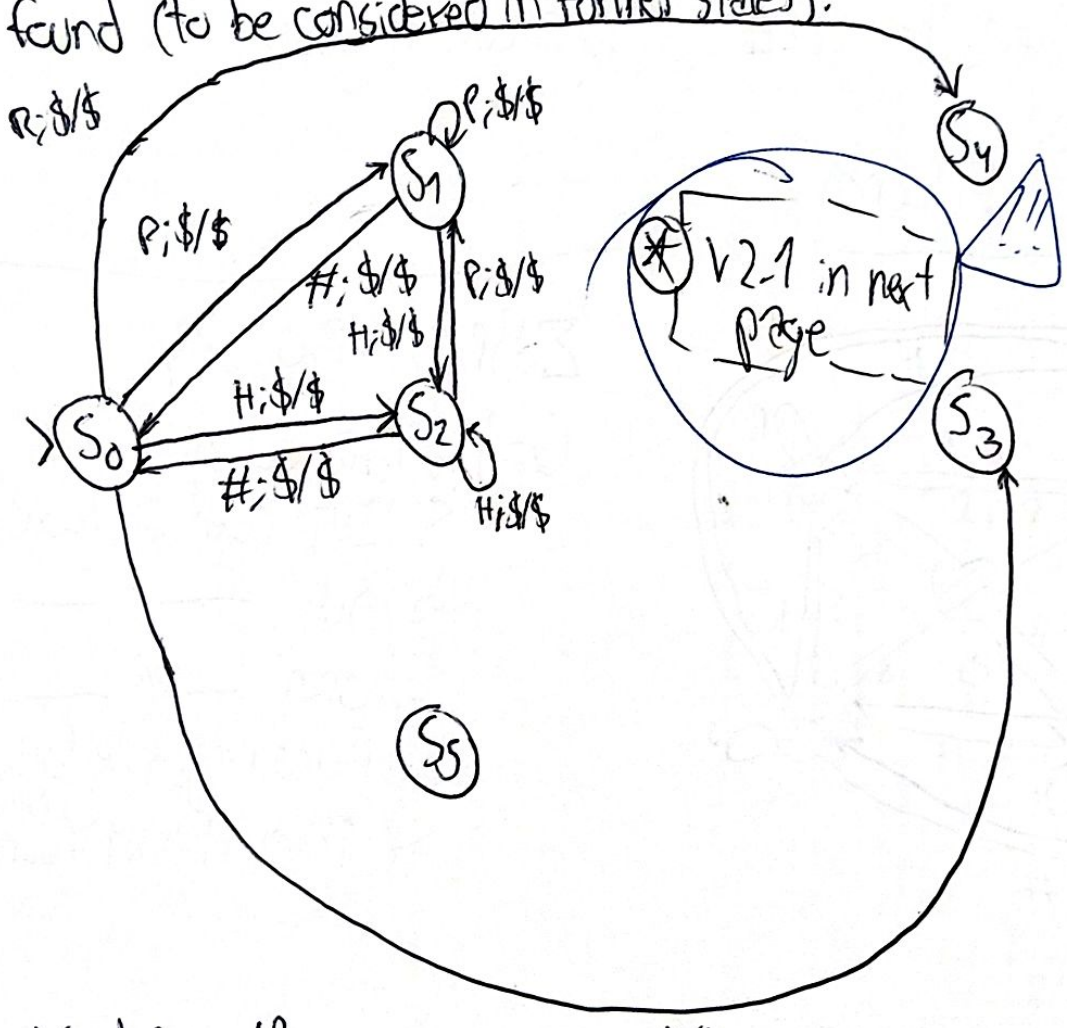
V2.1

For this new version, we will split a pattern into parts depending on its atomic behavior, such that:

- 'INI': 1 part (composed by Pauli gates).
- 'SUP': 1 part (composed by Hadamard gates).
- 'ORA': 1 part (composed by the Oracle itself).
- 'ENT': 2 parts { 1) Superposition of control qubit (ENT_0). 2) CNOT for both qubits (ENT_1).

To model the atomic-division of those patterns for its recognition, we will adapt the DFA v1.3 to a PUSHDOWN AUTOMATON (PA) v.2.1.

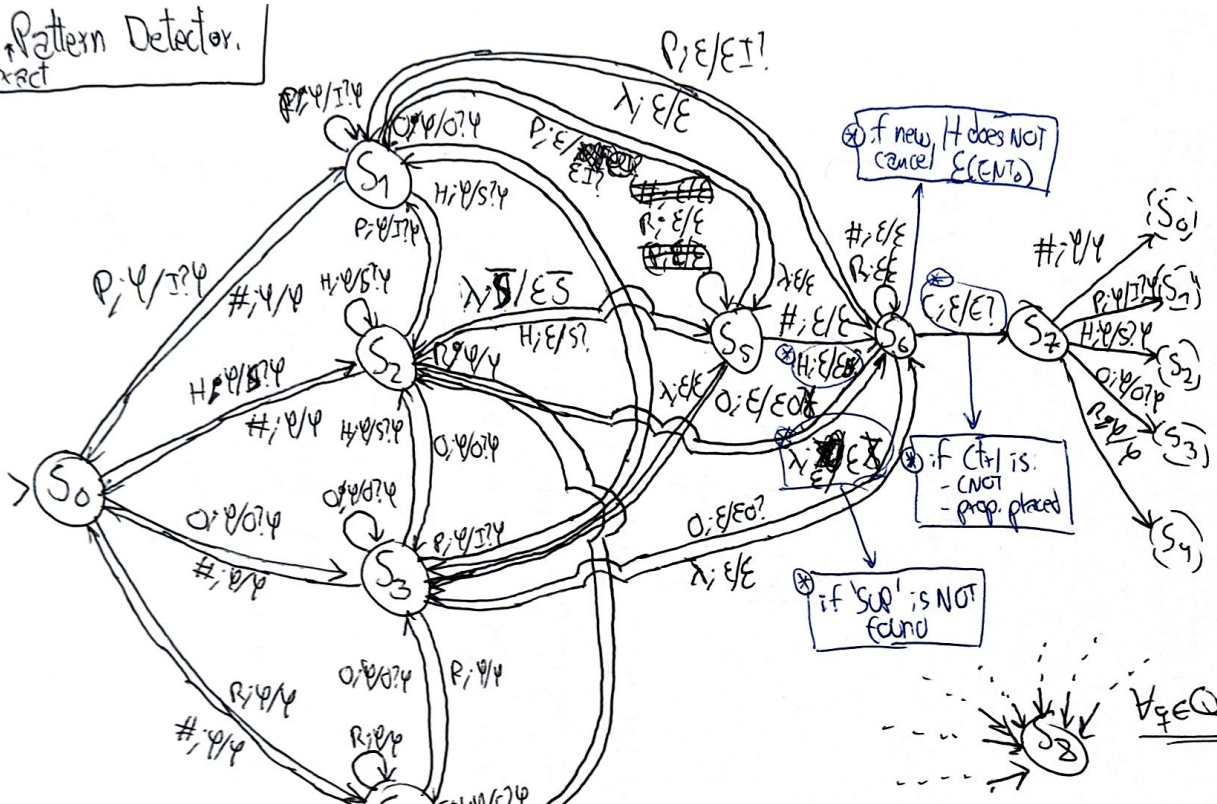
In the PA's stack we will store whether an atomic-part of the pattern is found (to be considered in further states).



Let's define Ψ as: 0;$/$

- Any symbol in Γ (stack alphabet) excepting those used in other transitions from that state.
- I/.../#s(?) means that could be found or not.

PDA - Potential Pattern Detector. Exact



Donde/Where:

- ψ ≡ Any symbol in Γ (stack alphabet) excepting those used in other transition/s from that state.
- I/.../S (?) = Γ (!) means that could be written in the stack or not (depends on the Exact Pattern Detector mod.).
- $\overline{I}/\ldots/\overline{S} = \overline{\Gamma}$ means any symbol except the one written under the upper line.

/// → Code/Part of the EPDM

- Formal definition of PEPDM-PD-λ:

A pushdown automateaw with λ-transitions defined as M=(Q, Σ, Γ, q_0, z, δ, F) where:

$q_0 = S_0$

$z = \$$

$F = \{S_8\}$

$\Sigma = \{P, R, H, T, C, \#, 0, \text{EoC}\}$ (End of Circuit (EoC))

$Q = \{$
- S_0: New Column (no atomic-part detected). → No pattern in progress
- S_1: 'INI' potentially found.
- S_2: 'SUP' —"— .
- S_3: 'ORA' —"— .
- S_4: Remaining gate found [Neutral] (no atomic-part detected).
- S_5: Atomic Part 'ENT$_0$' (ε) found potentially. Pattern in progress
- S_6: Transitioning between 'ENT$_0$' → 'ENT$_1$' (atomic-part detected).
- S_7: 'ENT' potentially found.
- S_8: Search has finished.

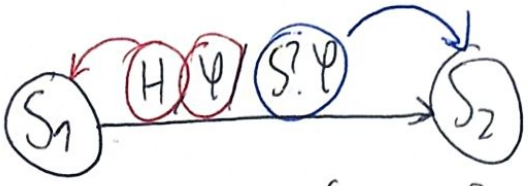
$\}$

$\Gamma = \{I, S, O, \epsilon, \varepsilon, \$\}$

$\delta =$ //Check prev page 44 PDA-λ.

-Code Translation for Transition

Let's take as example the following transition:



Let's now give the pseudocode for S_2 & S_1:

```
def self._s1():
    #Inform
    ...
    #Task
    ...
    #Transition
    self.gate = self._next()
    if PAULI_R.search(self.gate) is not None:
        ...
        elif self.gate='H':
            return self._s2().
    ...
def _s2():
    #Inform
    ...
    #Task
    if checker.check_sup(gate_tup):
        self.stack.pop()
        self.stack.append('s')
    #Transition
    ...
```

(q, col, str)

depends on transition

!!! IMPORTANT !!!

Note the transition-tasks division:

- The condition regarding the stack top is done in S_1 (left-most state).
- The stack replacing is done in S_2 (right state).