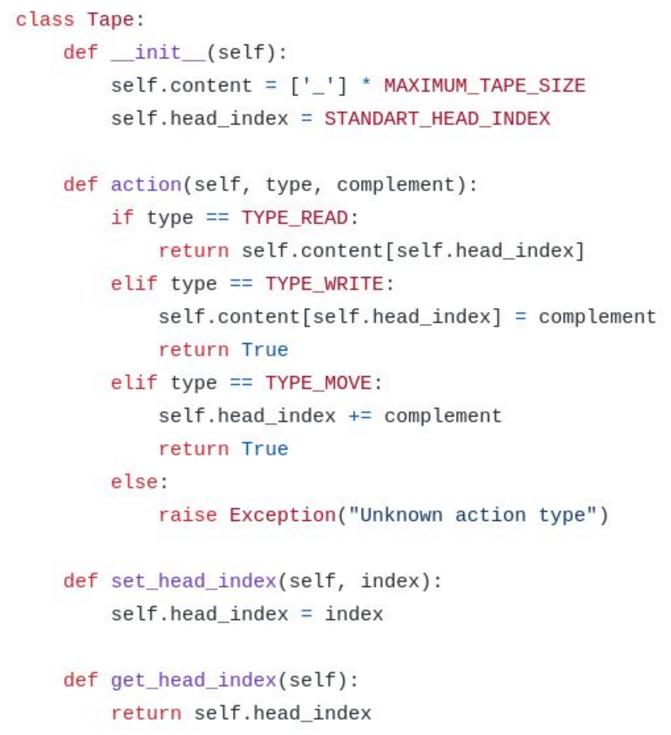
Reversor de Máquina de Turing

João Zucchi, Lorenzo Weber,
Luis Henrique Pozzebon, Raul Steinmetz

Processador de Máquina de Turing

A. Fita

Unidade básica que representa uma fita, em uma máquina existirão três. Nessa fita é possível ler, escrever e mover o cabeçote.

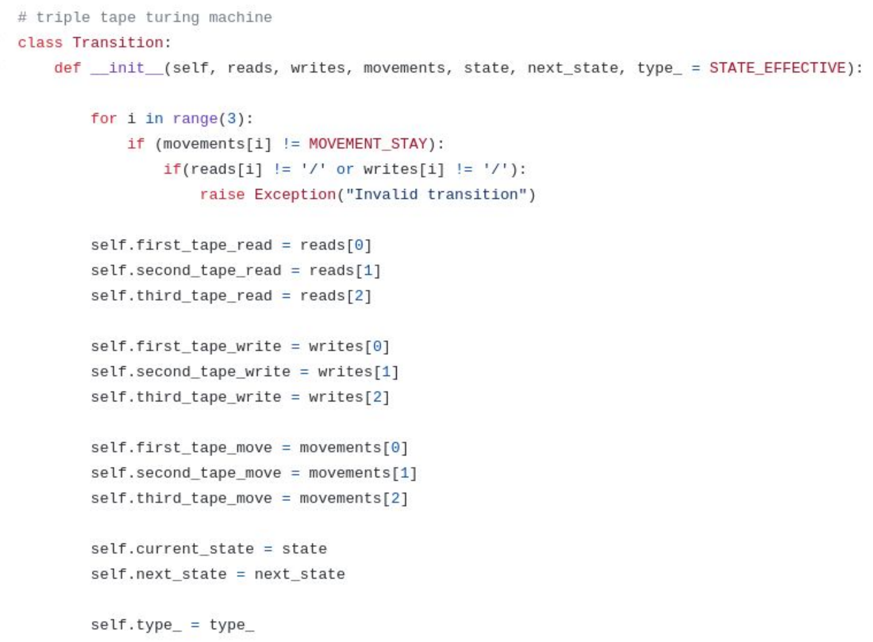


```
class Tape:
    def __init__(self):
        self.content = ['_'] * MAXIMUM_TAPE_SIZE
        self.head_index = STANDART_HEAD_INDEX

    def action(self, type, complement):
        if type == TYPE_READ:
            return self.content[self.head_index]
        elif type == TYPE_WRITE:
            self.content[self.head_index] = complement
            return True
        elif type == TYPE_MOVE:
            self.head_index += complement
            return True
        else:
            raise Exception("Unknown action type")

    def set_head_index(self, index):
        self.head_index = index

    def get_head_index(self):
        return self.head_index
```

Processador de Máquina de Turing

B. Transação

Como a máquina é quadrupla, só é possivel ler/escrever ou mover separadamente.

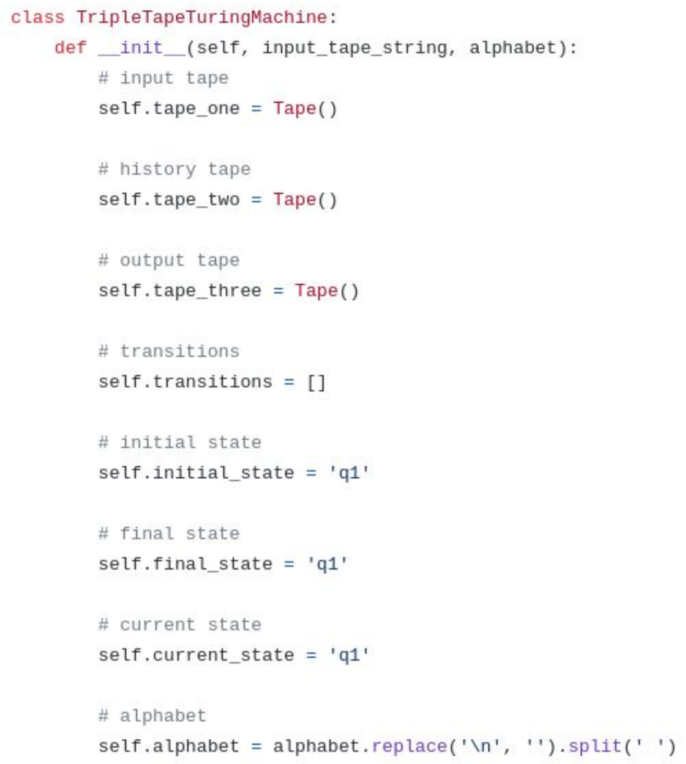


```
# triple tape turing machine
class Transition:
    def __init__(self, reads, writes, movements, state, next_state, type_ = STATE_EFFECTIVE):

        for i in range(3):
            if (movements[i] != MOVEMENT_STAY):
                if(reads[i] != '/' or writes[i] != '/'):
                    raise Exception("Invalid transition")

        self.first_tape_read = reads[0]
        self.second_tape_read = reads[1]
        self.third_tape_read = reads[2]

        self.first_tape_write = writes[0]
        self.second_tape_write = writes[1]
        self.third_tape_write = writes[2]

        self.first_tape_move = movements[0]
        self.second_tape_move = movements[1]
        self.third_tape_move = movements[2]

        self.current_state = state
        self.next_state = next_state

        self.type_ = type_
```

Processador de Máquina de Turing

C. Máquina tripla

A maquina de turing implementada possuí três fitas, transições ilimitadas e informações de estado. Além de algumas funções demonstradas nos próximos slides.

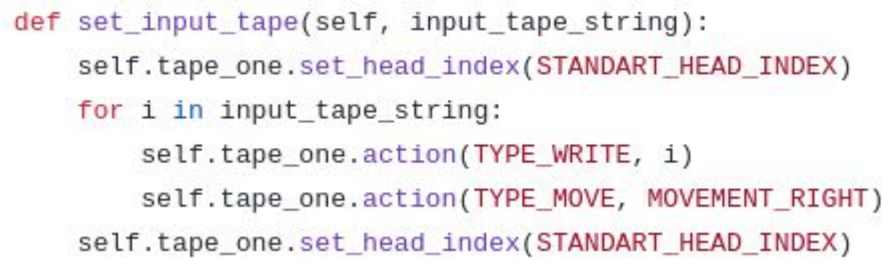


```python
class TripleTapeTuringMachine:
    def __init__(self, input_tape_string, alphabet):
        # input tape
        self.tape_one = Tape()

        # history tape
        self.tape_two = Tape()

        # output tape
        self.tape_three = Tape()

        # transitions
        self.transitions = []

        # initial state
        self.initial_state = 'q1'

        # final state
        self.final_state = 'q1'

        # current state
        self.current_state = 'q1'

        # alphabet
        self.alphabet = alphabet.replace('\n', '').split(' ')
```

Processador de Máquina de Turing

C. Máquina tripla

A função ao lado seta uma string de input na fita de número 1

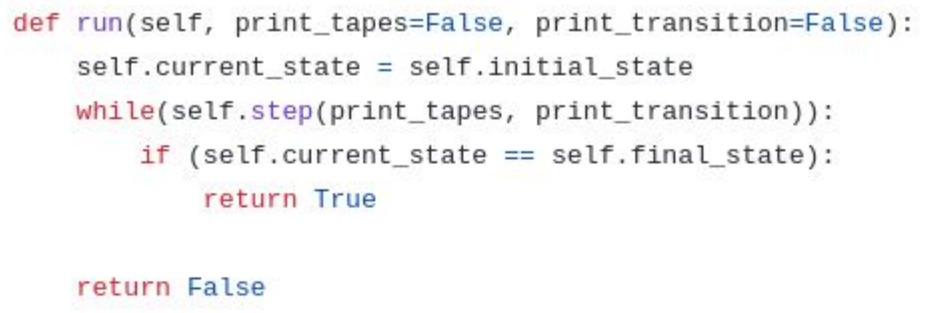


```
def set_input_tape(self, input_tape_string):
    self.tape_one.set_head_index(STANDART_HEAD_INDEX)
    for i in input_tape_string:
        self.tape_one.action(TYPE_WRITE, i)
        self.tape_one.action(TYPE_MOVE, MOVEMENT_RIGHT)
    self.tape_one.set_head_index(STANDART_HEAD_INDEX)
```

Processador de Máquina de Turing

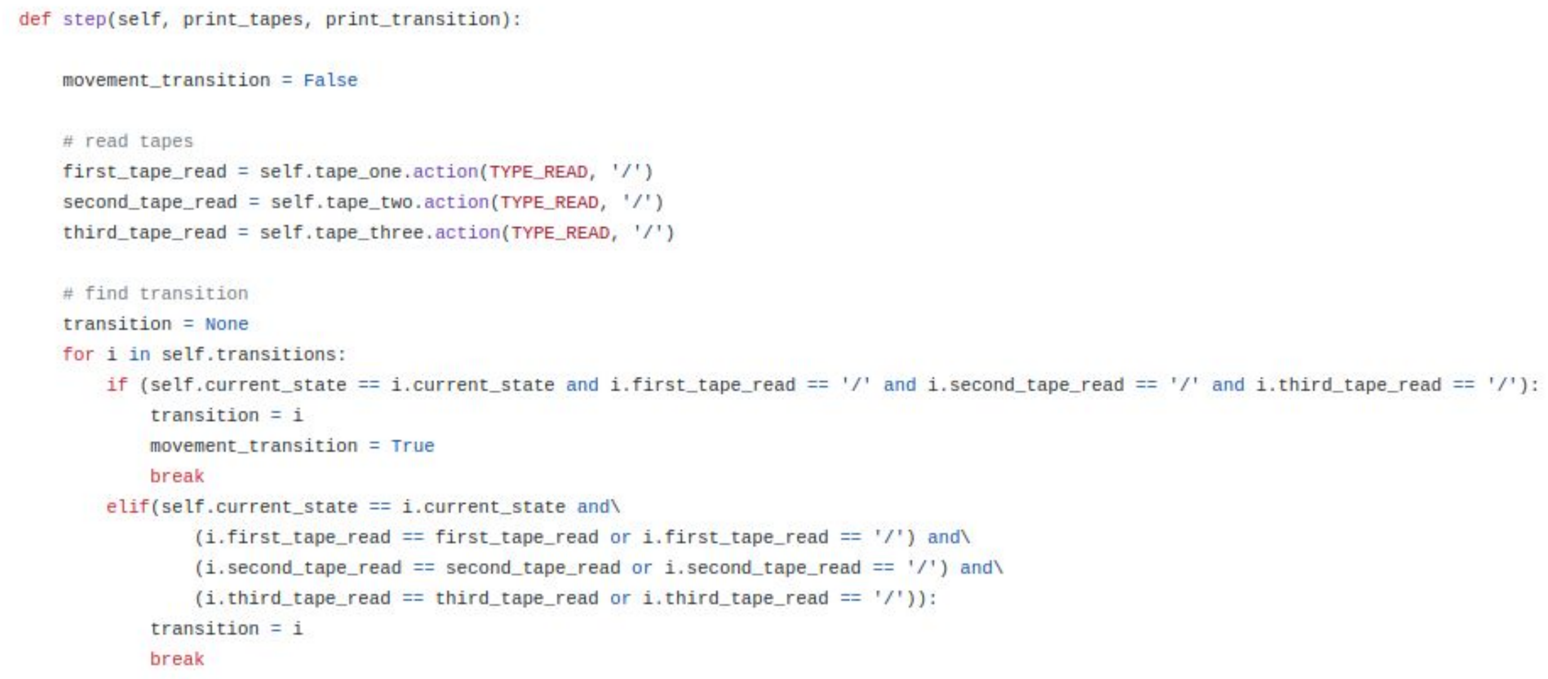
C. Máquina tripla

A função ao lado processa a máquina de turing, juntamente com a função step.



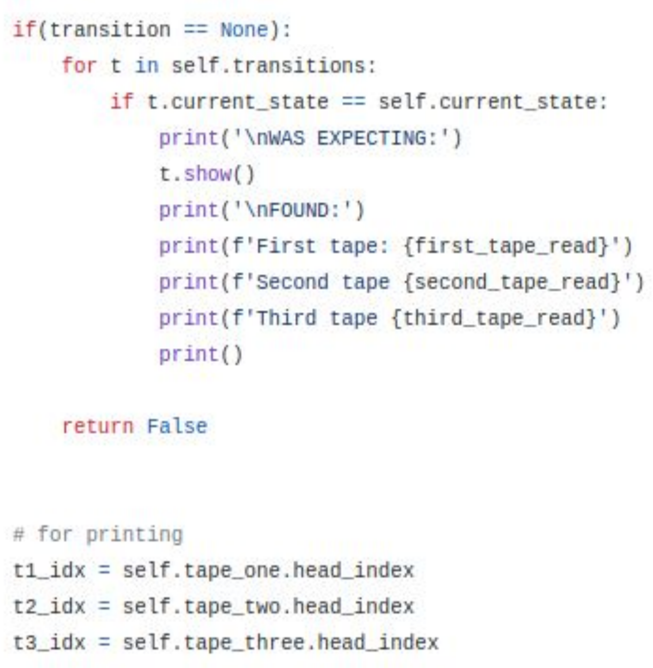
```
def run(self, print_tapes=False, print_transition=False):
    self.current_state = self.initial_state
    while(self.step(print_tapes, print_transition)):
        if (self.current_state == self.final_state):
            return True

    return False
```

Processador de Máquina de Turing

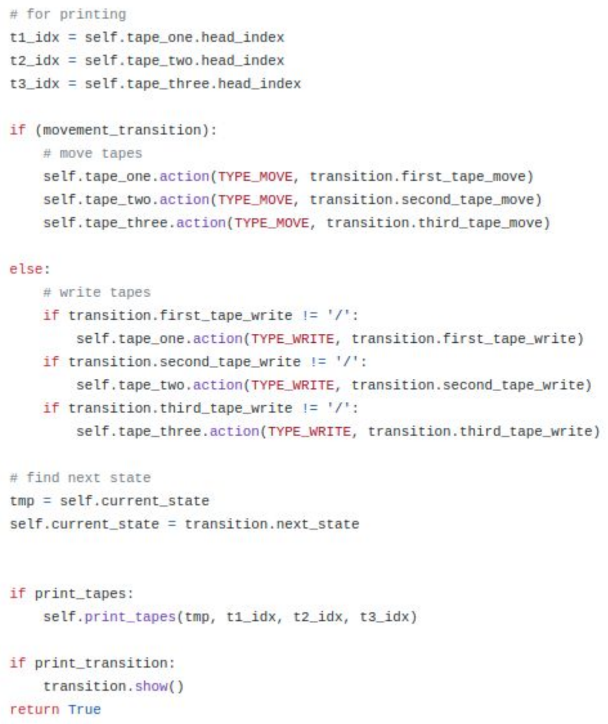


```
def step(self, print_tapes, print_transition):

    movement_transition = False

    # read tapes
    first_tape_read = self.tape_one.action(TYPE_READ, '/')
    second_tape_read = self.tape_two.action(TYPE_READ, '/')
    third_tape_read = self.tape_three.action(TYPE_READ, '/')

    # find transition
    transition = None
    for i in self.transitions:
        if (self.current_state == i.current_state and i.first_tape_read == '/' and i.second_tape_read == '/' and i.third_tape_read == '/'):
            transition = i
            movement_transition = True
            break
        elif(self.current_state == i.current_state and\
                (i.first_tape_read == first_tape_read or i.first_tape_read == '/') and\
                (i.second_tape_read == second_tape_read or i.second_tape_read == '/') and\
                (i.third_tape_read == third_tape_read or i.third_tape_read == '/')):
            transition = i
            break
```

Processador de Máquina de Turing



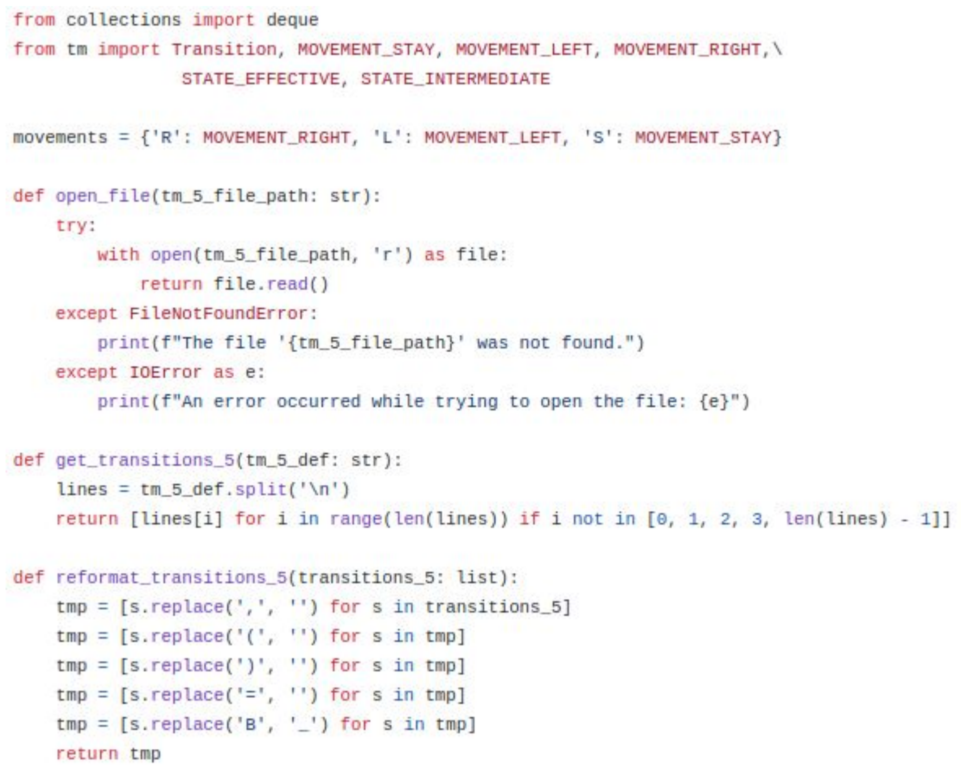
```python
if(transition == None):
    for t in self.transitions:
        if t.current_state == self.current_state:
            print('\nWAS EXPECTING:')
            t.show()
            print('\nFOUND:')
            print(f'First tape: {first_tape_read}')
            print(f'Second tape {second_tape_read}')
            print(f'Third tape {third_tape_read}')
            print()

    return False


# for printing
t1_idx = self.tape_one.head_index
t2_idx = self.tape_two.head_index
t3_idx = self.tape_three.head_index
```



```python
# for printing
t1_idx = self.tape_one.head_index
t2_idx = self.tape_two.head_index
t3_idx = self.tape_three.head_index

if (movement_transition):
    # move tapes
    self.tape_one.action(TYPE_MOVE, transition.first_tape_move)
    self.tape_two.action(TYPE_MOVE, transition.second_tape_move)
    self.tape_three.action(TYPE_MOVE, transition.third_tape_move)

else:
    # write tapes
    if transition.first_tape_write != '/':
        self.tape_one.action(TYPE_WRITE, transition.first_tape_write)
    if transition.second_tape_write != '/':
        self.tape_two.action(TYPE_WRITE, transition.second_tape_write)
    if transition.third_tape_write != '/':
        self.tape_three.action(TYPE_WRITE, transition.third_tape_write)

# find next state
tmp = self.current_state
self.current_state = transition.next_state


if print_tapes:
    self.print_tapes(tmp, t1_idx, t2_idx, t3_idx)

if print_transition:
    transition.show()
return True
```

Parser e Conversor 5-4

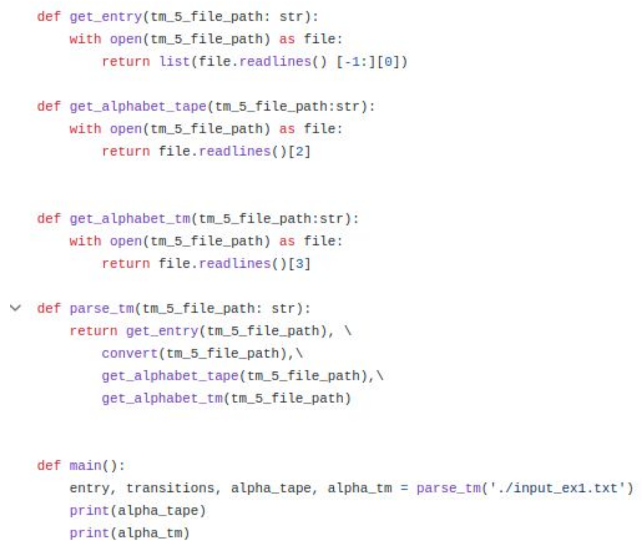
Parser da definição da máquina + conversor de quíntupla para quádrupla



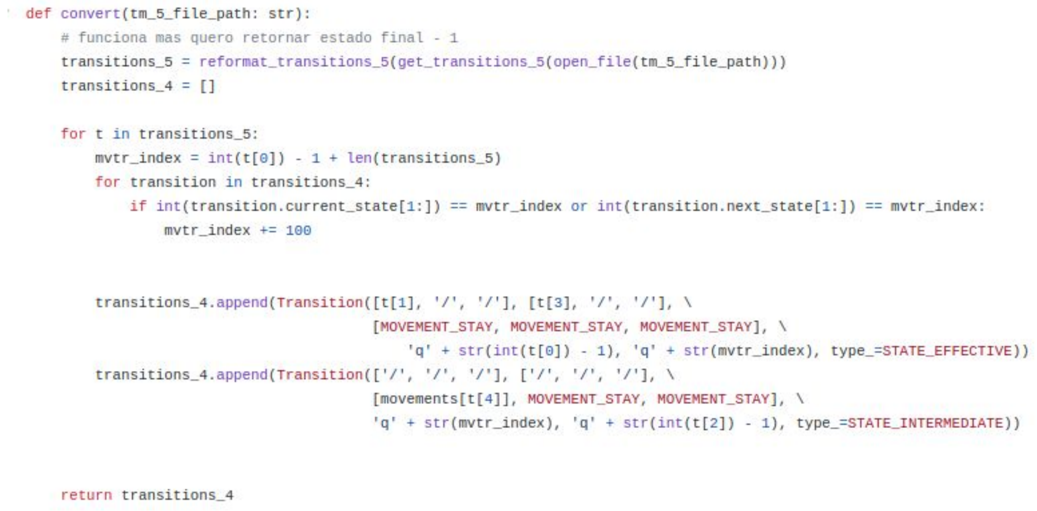
```python
from collections import deque
from tm import Transition, MOVEMENT_STAY, MOVEMENT_LEFT, MOVEMENT_RIGHT,\
                STATE_EFFECTIVE, STATE_INTERMEDIATE

movements = {'R': MOVEMENT_RIGHT, 'L': MOVEMENT_LEFT, 'S': MOVEMENT_STAY}

def open_file(tm_5_file_path: str):
    try:
        with open(tm_5_file_path, 'r') as file:
            return file.read()
    except FileNotFoundError:
        print(f"The file '{tm_5_file_path}' was not found.")
    except IOError as e:
        print(f"An error occurred while trying to open the file: {e}")

def get_transitions_5(tm_5_def: str):
    lines = tm_5_def.split('\n')
    return [lines[i] for i in range(len(lines)) if i not in [0, 1, 2, 3, len(lines) - 1]]

def reformat_transitions_5(transitions_5: list):
    tmp = [s.replace(',', '') for s in transitions_5]
    tmp = [s.replace('(', '') for s in tmp]
    tmp = [s.replace(')', '') for s in tmp]
    tmp = [s.replace('=', '') for s in tmp]
    tmp = [s.replace('B', '_') for s in tmp]
    return tmp
```

Parser e Conversor 5-4

Parser da definição da máquina + conversor de quíntupla para quádrupla



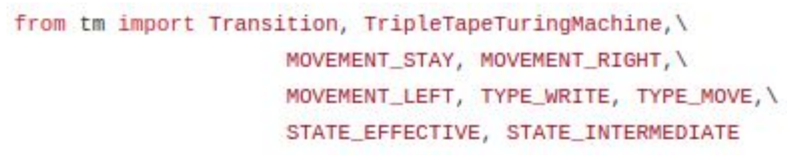
```python
def get_entry(tm_5_file_path: str):
    with open(tm_5_file_path) as file:
        return list(file.readlines() [-1:][0])

def get_alphabet_tape(tm_5_file_path:str):
    with open(tm_5_file_path) as file:
        return file.readlines()[2]


def get_alphabet_tm(tm_5_file_path:str):
    with open(tm_5_file_path) as file:
        return file.readlines()[3]

def parse_tm(tm_5_file_path: str):
    return get_entry(tm_5_file_path), \
        convert(tm_5_file_path),\
        get_alphabet_tape(tm_5_file_path),\
        get_alphabet_tm(tm_5_file_path)


def main():
    entry, transitions, alpha_tape, alpha_tm = parse_tm('./input_ex1.txt')
    print(alpha_tape)
    print(alpha_tm)
```



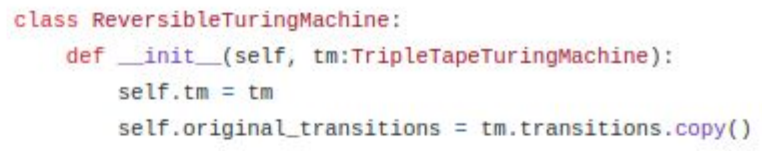
```python
def convert(tm_5_file_path: str):
    # funciona mas quero retornar estado final - 1
    transitions_5 = reformat_transitions_5(get_transitions_5(open_file(tm_5_file_path)))
    transitions_4 = []

    for t in transitions_5:
        mvtr_index = int(t[0]) - 1 + len(transitions_5)
        for transition in transitions_4:
            if int(transition.current_state[1:]) == mvtr_index or int(transition.next_state[1:]) == mvtr_index:
                mvtr_index += 100


        transitions_4.append(Transition([t[1], '/', '/'], [t[3], '/', '/'], \
                                        [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], \
                                            'q' + str(int(t[0]) - 1), 'q' + str(mvtr_index), type_=STATE_EFFECTIVE))
        transitions_4.append(Transition(['/', '/', '/'], ['/', '/', '/'], \
                                        [movements[t[4]], MOVEMENT_STAY, MOVEMENT_STAY], \
                                        'q' + str(mvtr_index), 'q' + str(int(t[2]) - 1), type_=STATE_INTERMEDIATE))


    return transitions_4
```

Máquina Reversível

Classe que recebe uma máquina de turing e tem autofunções que convertem em máquina reversível



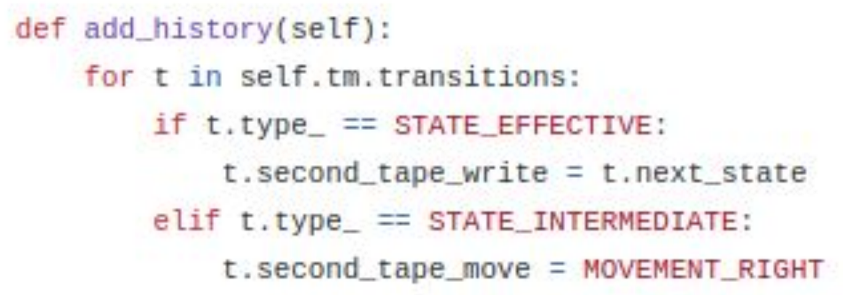
```
from tm import Transition, TripleTapeTuringMachine,\
                    MOVEMENT_STAY, MOVEMENT_RIGHT,\
                    MOVEMENT_LEFT, TYPE_WRITE, TYPE_MOVE,\
                    STATE_EFFECTIVE, STATE_INTERMEDIATE
```



```
class ReversibleTuringMachine:
    def __init__(self, tm:TripleTapeTuringMachine):
        self.tm = tm
        self.original_transitions = tm.transitions.copy()
```

Máquina Reversível

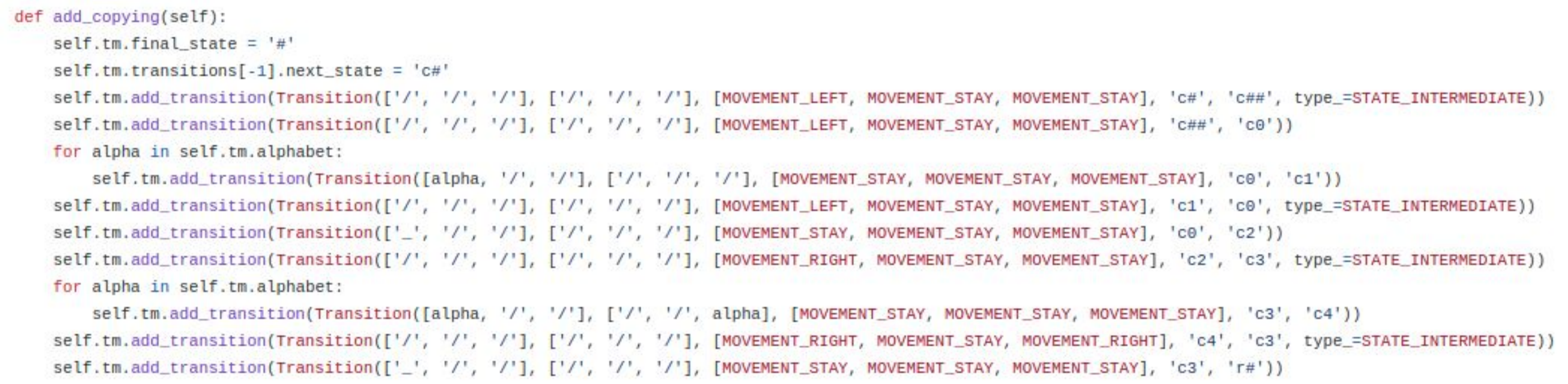
Função para adicionar histórico às transições da máquina



```python
def add_history(self):
    for t in self.tm.transitions:
        if t.type_ == STATE_EFFECTIVE:
            t.second_tape_write = t.next_state
        elif t.type_ == STATE_INTERMEDIATE:
            t.second_tape_move = MOVEMENT_RIGHT
```

Máquina Reversível

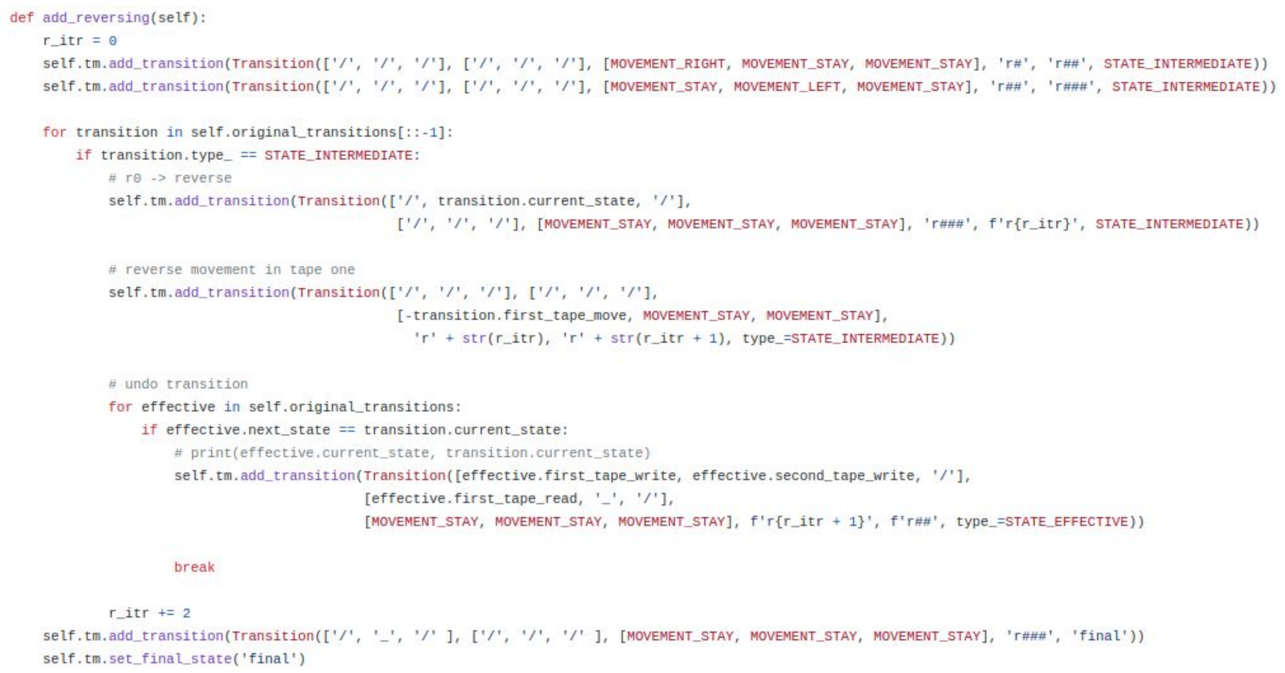
Função que acopla máquina de turing que copia a fita 1 na fita 3



```
def add_copying(self):
    self.tm.final_state = '#'
    self.tm.transitions[-1].next_state = 'c#'
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_LEFT, MOVEMENT_STAY, MOVEMENT_STAY], 'c#', 'c##', type_=STATE_INTERMEDIATE))
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_LEFT, MOVEMENT_STAY, MOVEMENT_STAY], 'c##', 'c0'))
    for alpha in self.tm.alphabet:
        self.tm.add_transition(Transition([alpha, '/', '/'], ['/', '/', '/'], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'c0', 'c1'))
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_LEFT, MOVEMENT_STAY, MOVEMENT_STAY], 'c1', 'c0', type_=STATE_INTERMEDIATE))
    self.tm.add_transition(Transition(['_', '/', '/'], ['/', '/', '/'], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'c0', 'c2'))
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_RIGHT, MOVEMENT_STAY, MOVEMENT_STAY], 'c2', 'c3', type_=STATE_INTERMEDIATE))
    for alpha in self.tm.alphabet:
        self.tm.add_transition(Transition([alpha, '/', '/'], ['/', '/', alpha], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'c3', 'c4'))
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_RIGHT, MOVEMENT_STAY, MOVEMENT_RIGHT], 'c4', 'c3', type_=STATE_INTERMEDIATE))
    self.tm.add_transition(Transition(['_', '/', '/'], ['/', '/', '/'], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'c3', 'r#'))
```

Máquina Reversível

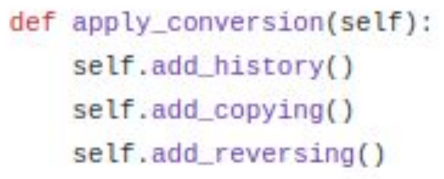
Função que acopla máquina inversora



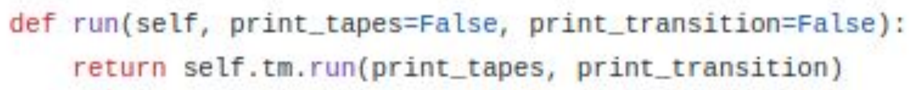
```
def add_reversing(self):
    r_itr = 0
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_RIGHT, MOVEMENT_STAY, MOVEMENT_STAY], 'r#', 'r##', STATE_INTERMEDIATE))
    self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'], [MOVEMENT_STAY, MOVEMENT_LEFT, MOVEMENT_STAY], 'r##', 'r###', STATE_INTERMEDIATE))

    for transition in self.original_transitions[::-1]:
        if transition.type_ == STATE_INTERMEDIATE:
            # r0 -> reverse
            self.tm.add_transition(Transition(['/', transition.current_state, '/'],
                                               ['/', '/', '/'], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'r###', f'r{r_itr}', STATE_INTERMEDIATE))

            # reverse movement in tape one
            self.tm.add_transition(Transition(['/', '/', '/'], ['/', '/', '/'],
                                               [-transition.first_tape_move, MOVEMENT_STAY, MOVEMENT_STAY],
                                                 'r' + str(r_itr), 'r' + str(r_itr + 1), type_=STATE_INTERMEDIATE))

            # undo transition
            for effective in self.original_transitions:
                if effective.next_state == transition.current_state:
                    # print(effective.current_state, transition.current_state)
                    self.tm.add_transition(Transition([effective.first_tape_write, effective.second_tape_write, '/'],
                                                       [effective.first_tape_read, '_', '/'],
                                                       [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], f'r{r_itr + 1}', f'r##', type_=STATE_EFFECTIVE))

                    break

            r_itr += 2
    self.tm.add_transition(Transition(['/', '_', '/' ], ['/', '/', '/' ], [MOVEMENT_STAY, MOVEMENT_STAY, MOVEMENT_STAY], 'r###', 'final'))
    self.tm.set_final_state('final')
```

Máquina Reversível

Função que transforma a máquina de turing com ajuda das outras e método run



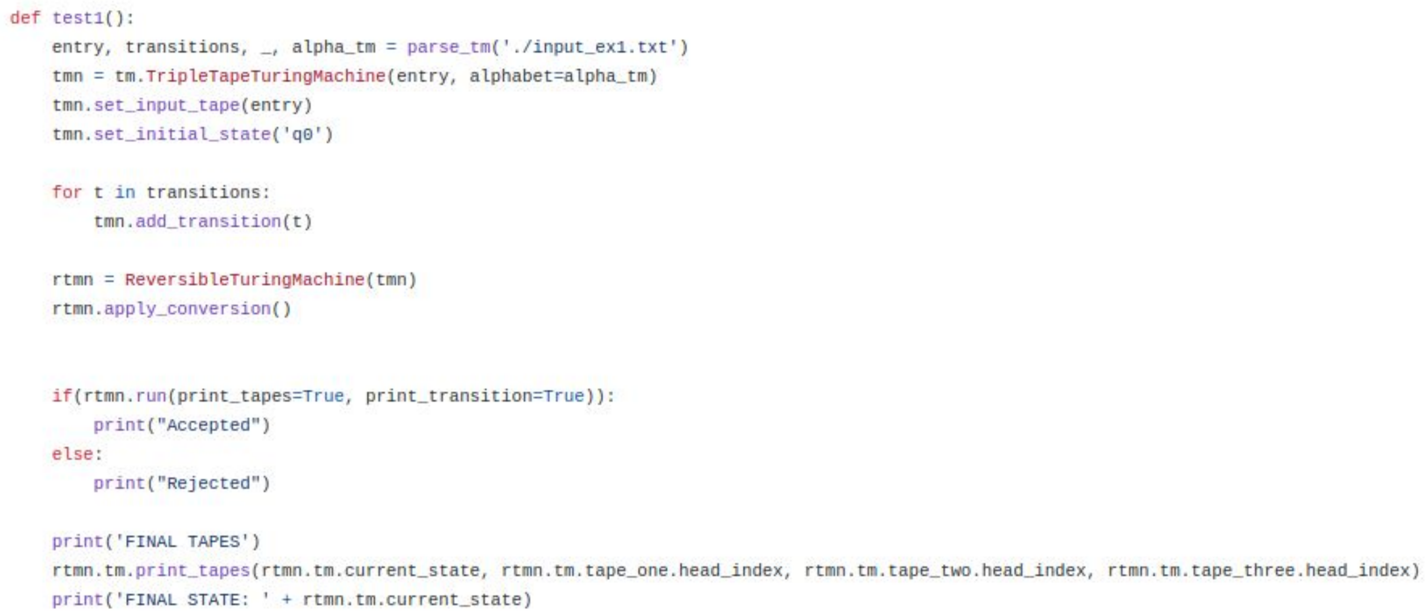
```python
def apply_conversion(self):
    self.add_history()
    self.add_copying()
    self.add_reversing()
```



```python
def run(self, print_tapes=False, print_transition=False):
    return self.tm.run(print_tapes, print_transition)
```

Rodando o código

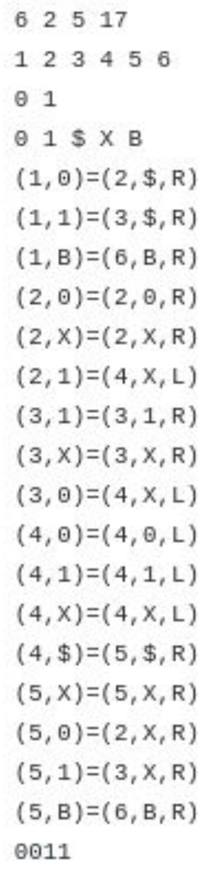
Rodando tudo



```python
def test1():
    entry, transitions, _, alpha_tm = parse_tm('./input_ex1.txt')
    tmn = tm.TripleTapeTuringMachine(entry, alphabet=alpha_tm)
    tmn.set_input_tape(entry)
    tmn.set_initial_state('q0')

    for t in transitions:
        tmn.add_transition(t)

    rtmn = ReversibleTuringMachine(tmn)
    rtmn.apply_conversion()


    if(rtmn.run(print_tapes=True, print_transition=True)):
        print("Accepted")
    else:
        print("Rejected")

    print('FINAL TAPES')
    rtmn.tm.print_tapes(rtmn.tm.current_state, rtmn.tm.tape_one.head_index, rtmn.tm.tape_two.head_index, rtmn.tm.tape_three.head_index)
    print('FINAL STATE: ' + rtmn.tm.current_state)
```

Input + Output

- Input



```
6 2 5 17
1 2 3 4 5 6
0 1
0 1 $ X B
(1,0)=(2,$,R)
(1,1)=(3,$,R)
(1,B)=(6,B,R)
(2,0)=(2,0,R)
(2,X)=(2,X,R)
(2,1)=(4,X,L)
(3,1)=(3,1,R)
(3,X)=(3,X,R)
(3,0)=(4,X,L)
(4,0)=(4,0,L)
(4,1)=(4,1,L)
(4,X)=(4,X,L)
(4,$)=(5,$,R)
(5,X)=(5,X,R)
(5,0)=(2,X,R)
(5,1)=(3,X,R)
(5,B)=(6,B,R)
0011
```

```
T1 = ['_', '_', '_', '_', '_'](q0)['$', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](q0)['q17', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q0)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = 0 $ 0
Second tape = / q17 0
Third tape = / / 0
Current state = q0
Next state = q17
Type = 0

T1 = ['_', '_', '_', '_', '_'](q17)['0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](q17)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q17)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 1
Third tape = / / 0
Current state = q17
Next state = q1
Type = 1

T1 = ['_', '_', '_', '_', '_', '$'](q1)['0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17'](q1)['q18', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q1)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = 0 0 0
Second tape = / q18 0
Third tape = / / 0
Current state = q1
Next state = q18
Type = 0
```

```
T1 = ['_', '_', '_', '_', '_', '$'](q18)['1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17'](q18)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q18)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 1
Third tape = / / 0
Current state = q18
Next state = q1
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', '0'](q1)['X', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18'](q1)['q218', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q1)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = 1 X 0
Second tape = / q218 0
Third tape = / / 0
Current state = q1
Next state = q218
Type = 0

T1 = ['_', '_', '_', '_', '_', '$', '0'](q218)['0', 'X', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18'](q218)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q218)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / -1
Second tape = / / 1
Third tape = / / 0
Current state = q218
Next state = q3
Type = 1
```

```
T1 = ['_', '_', '_', '_', '_', '$', 'X'](q1)['X', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121'](q1)['q118', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q1)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = X X 0
Second tape = / q118 0
Third tape = / / 0
Current state = q1
Next state = q118
Type = 0

T1 = ['_', '_', '_', '_', '_', '$', 'X'](q118)['1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121'](q118)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q118)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 1
Third tape = / / 0
Current state = q118
Next state = q1
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X'](q1)['X', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118'](q1)['q218', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q1)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = 1 X 0
Second tape = / q218 0
Third tape = / / 0
Current state = q1
Next state = q218
Type = 0
```

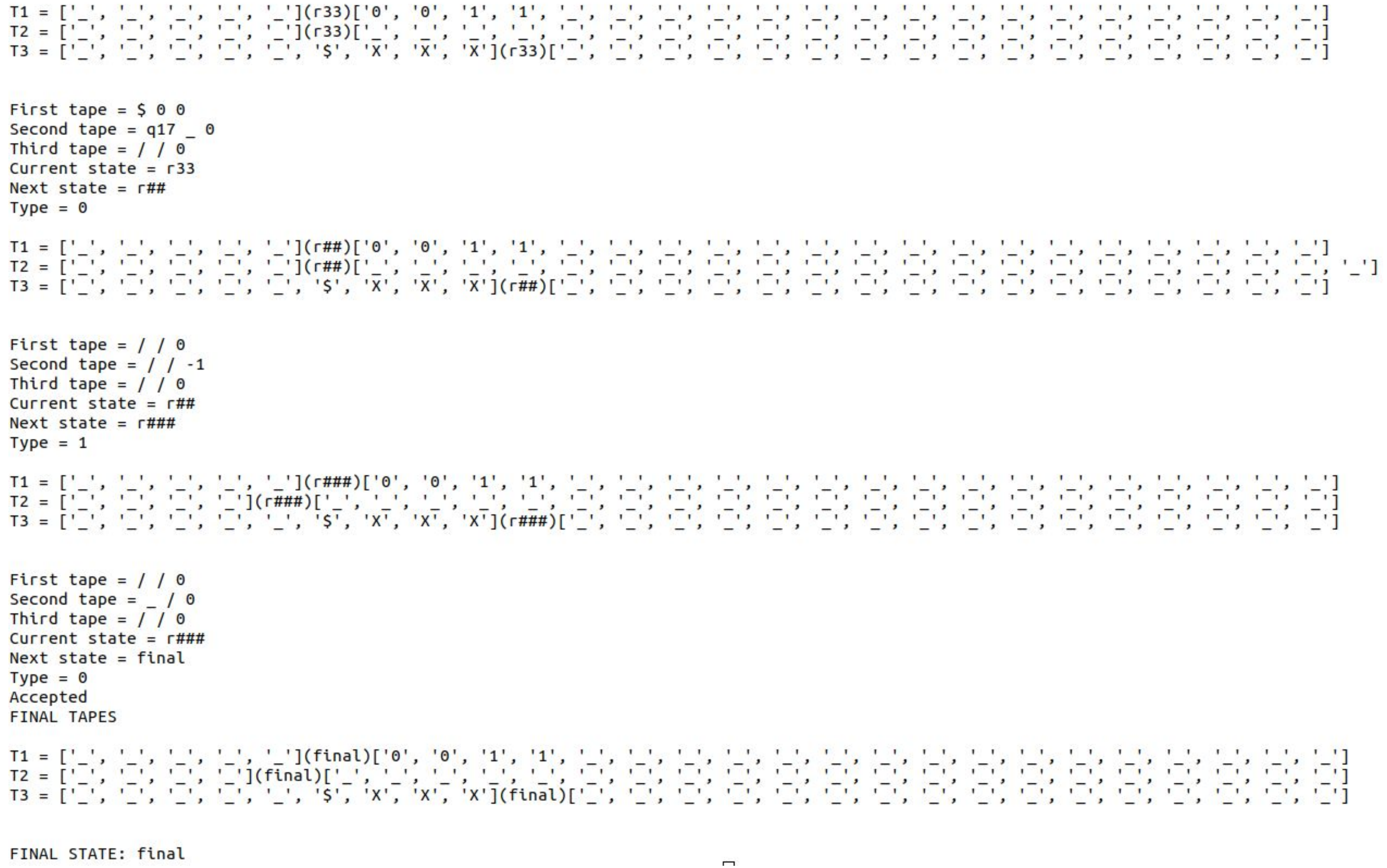
```
T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](q4)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21'](q4)['q321', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q4)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = _ _ 0
Second tape = / q321 0
Third tape = / / 0
Current state = q4
Next state = q321
Type = 0

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](q321)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21'](q321)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](q321)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 1
Third tape = / / 0
Current state = q321
Next state = c#
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X', '_'](c#)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c#)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](c#)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / -1
Second tape = / / 0
Third tape = / / 0
Current state = c#
Next state = c##
Type = 1
```

```
T1 = ['_', '_', '_', '_', '_'](c3)['$', 'X', 'X', 'X', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c3)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](c3)['$', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = $ / 0
Second tape = / / 0
Third tape = / $ 0
Current state = c3
Next state = c4
Type = 0

T1 = ['_', '_', '_', '_', '_'](c4)['X', 'X', 'X', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c4)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_'](c4)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 0
Third tape = / / 1
Current state = c4
Next state = c3
Type = 1

T1 = ['_', '_', '_', '_', '_', '$'](c3)['X', 'X', 'X', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c3)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$'](c3)['X', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = X / 0
Second tape = / / 0
Third tape = / X 0
Current state = c3
Next state = c4
Type = 0
```

```
T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X'](c4)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c4)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X'](c4)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 0
Third tape = / / 1
Current state = c4
Next state = c3
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](c3)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](c3)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](c3)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = _ / 0
Second tape = / / 0
Third tape = / / 0
Current state = c3
Next state = r#
Type = 0

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r#)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121', 'q118', 'q218', 'q220', 'q220', 'q320', 'q21', 'q21', 'q21', 'q321'](r#)['_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r#)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 1
Second tape = / / 0
Third tape = / / 0
Current state = r#
Next state = r##
Type = 1
```

```
T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X'](r###)['1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121'](r###)['q118', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r###)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 0
Second tape = q118 / 0
Third tape = / / 0
Current state = r###
Next state = r24
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', 'X', 'X'](r24)['X', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121'](r24)['q118', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r24)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / -1
Second tape = / / 0
Third tape = / / 0
Current state = r24
Next state = r25
Type = 1

T1 = ['_', '_', '_', '_', '_', '$', 'X'](r25)['X', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17', 'q18', 'q218', 'q20', 'q320', 'q121'](r25)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r25)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = X X 0
Second tape = q118 _ 0
Third tape = / / 0
Current state = r25
Next state = r##
Type = 0
```

```
T1 = ['_', '_', '_', '_', '_', '$'](r##)['0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_', 'q17'](r##)['q17', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r##)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 0
Second tape = / / -1
Third tape = / / 0
Current state = r##
Next state = r###
Type = 1

T1 = ['_', '_', '_', '_', '_', '$'](r###)['0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](r###)['q17', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r###)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 0
Second tape = q17 / 0
Third tape = / / 0
Current state = r###
Next state = r32
Type = 1

T1 = ['_', '_', '_', '_', '_', '$'](r32)['$', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](r32)['q17', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r32)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / -1
Second tape = / / 0
Third tape = / / 0
Current state = r32
Next state = r33
Type = 1
```



```
T1 = ['_', '_', '_', '_', '_'](r33)['0', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](r33)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r33)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = $ 0 0
Second tape = q17 _ 0
Third tape = / / 0
Current state = r33
Next state = r##
Type = 0

T1 = ['_', '_', '_', '_', '_'](r##)['0', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_', '_'](r##)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r##)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 0
Second tape = / / -1
Third tape = / / 0
Current state = r##
Next state = r###
Type = 1

T1 = ['_', '_', '_', '_', '_'](r###)['0', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_'](r###)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](r###)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


First tape = / / 0
Second tape = _ / 0
Third tape = / / 0
Current state = r###
Next state = final
Type = 0
Accepted
FINAL TAPES

T1 = ['_', '_', '_', '_', '_'](final)['0', '0', '1', '1', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T2 = ['_', '_', '_', '_'](final)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
T3 = ['_', '_', '_', '_', '_', '$', 'X', 'X', 'X'](final)['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']


FINAL STATE: final
```