Reinforcement Learning Project

Studente: Alessandro Mattei **Matricola:** 295441 **Email:** alessandro.mattei1@student.univaq.it

In [1]:
```
from agent import Agent
from state import State

from world import World
```

Introduzione

Il problema del cuoco

Consideriamo il caso in cui l'agente sia il vostro cuoco personale.

In particolare, l'agente (lo smiley sulla mappa) vuole cucinare la ricetta delle uova in base alle indicazioni dell'utente (strapazzate o in un budino).

Per cucinare la ricetta desiderata, l'agente deve prima raccogliere gli strumenti necessari (il frullino per le uova sulla mappa). Poi deve raggiungere i fornelli (la padella o il forno sulla mappa). Infine, può cucinare.

Non che ci siano due speciali celle interconnesse (contrassegnate con 1a G) che permettono all'agente di passare da un lato all'altro della mappa. Ma per farlo, l'agente deve esprimere la sua volontà di andare dall'altra parte.

Le celle in (4, 2) e (9, 3) sono i cancelli speciali. Permettono all'agente di passare da un lato all'altro della mappa. Queste due celle speciali sono collegate tra loro, ma l'agente deve esprimere la sua volontà di andare dall'altra parte.

Dato che siete molto affamati, è fondamentale che l'agente cucini le uova secondo i vostri gusti (strapazzate/pudding) il più velocemente possibile senza farvi aspettare più del necessario.

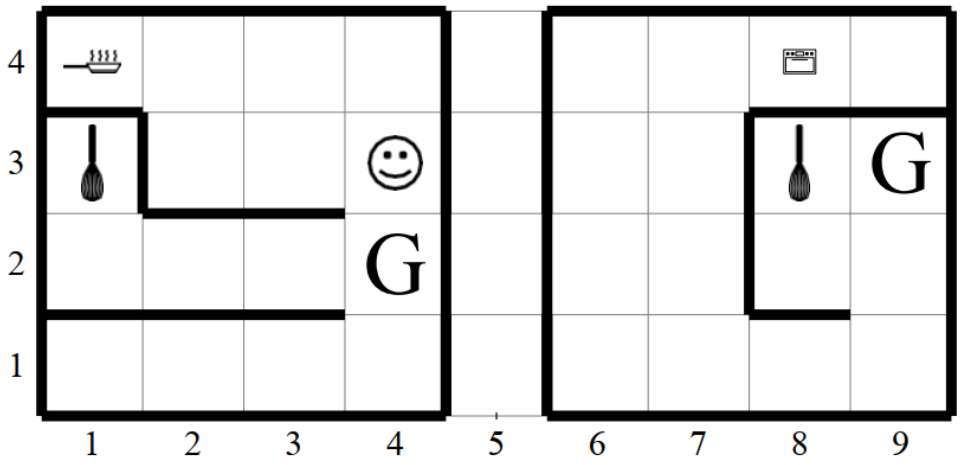

Figura 1: Un'istanza particolare del problema di cottura di Chef. L'obiettivo è che l'agente attualmente situato nello stato (4, 3) abbia una politica che porti sempre a cucinare le uova nella posizione (1, 4) o (8, 4). Le celle in (4, 2) e (9, 3) sono quelle del cancello speciale.

Consideriamo l'istanza mostrata nella Figura 1:

- Nella figura, l'agente si trova in (4, 3) (ma può iniziare in qualsiasi cella della griglia).
- L'agente ha bisogno di strumenti di cottura come il frullino per le uova in posizione (1, 3) e (8,3).
- Ci sono due obiettivi finali diversi, visualizzati come la padella in posizione (1, 4) e il forno in posizione (8, 4).
- Le celle in (4, 2) e (9, 3) sono i cancelli speciali. Permettono all'agente di passare da un lato all'afro della mappa. Queste due celle speciali sono collegate tra loro, ma l'agente deve esprimere la sua volontà di andare dall'altra parte.

- L'agente non può muoversi in diagonale.
- Le pareti sono rappresentate da linee nere spesse.
- L'agente non può muoversi attraverso i muri.
- Un episodio terminerà quando l'agente riuscirà a cucinare le uova strapazzate (vedi descrizione precedente).

Analisi

Per semplificare il problema consideriamo separatamente gli stati in cui l'agente non hanno in mano lo sbattitore da quelli che ha lo sbattitore (situato in posizione **(3,1)** e **(3.8)** visibile nelle foto in basso). Per far ciò ho duplicato la griglia.

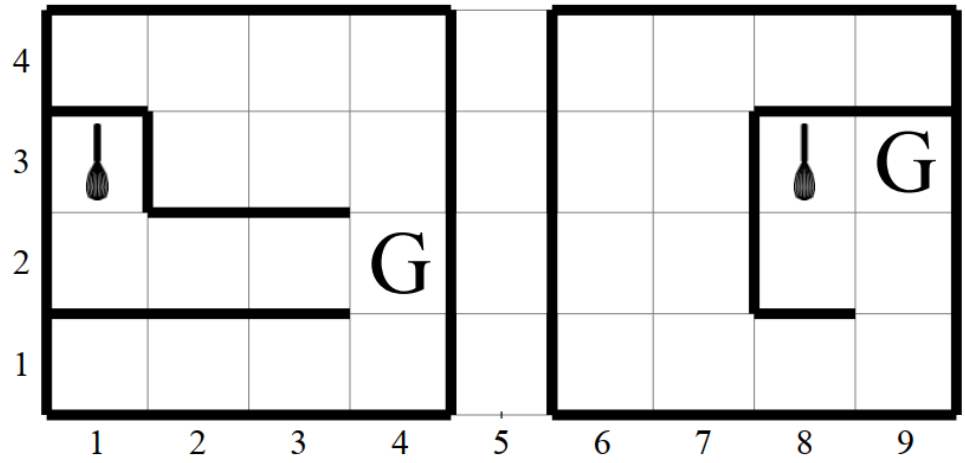

Figura 2: prima griglia che contiene solo gli sbattitori

Nella prima griglia (nella figura sopra), l'agente ha l'obiettivo di raggiungere uno dei due **sbattitori**.

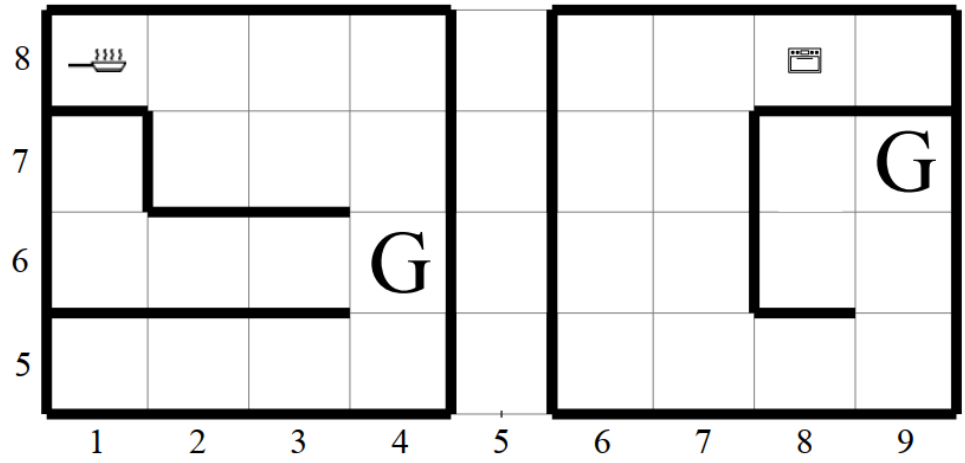

Figura 3: seconda griglia che contiene solo le postazioni di cottura

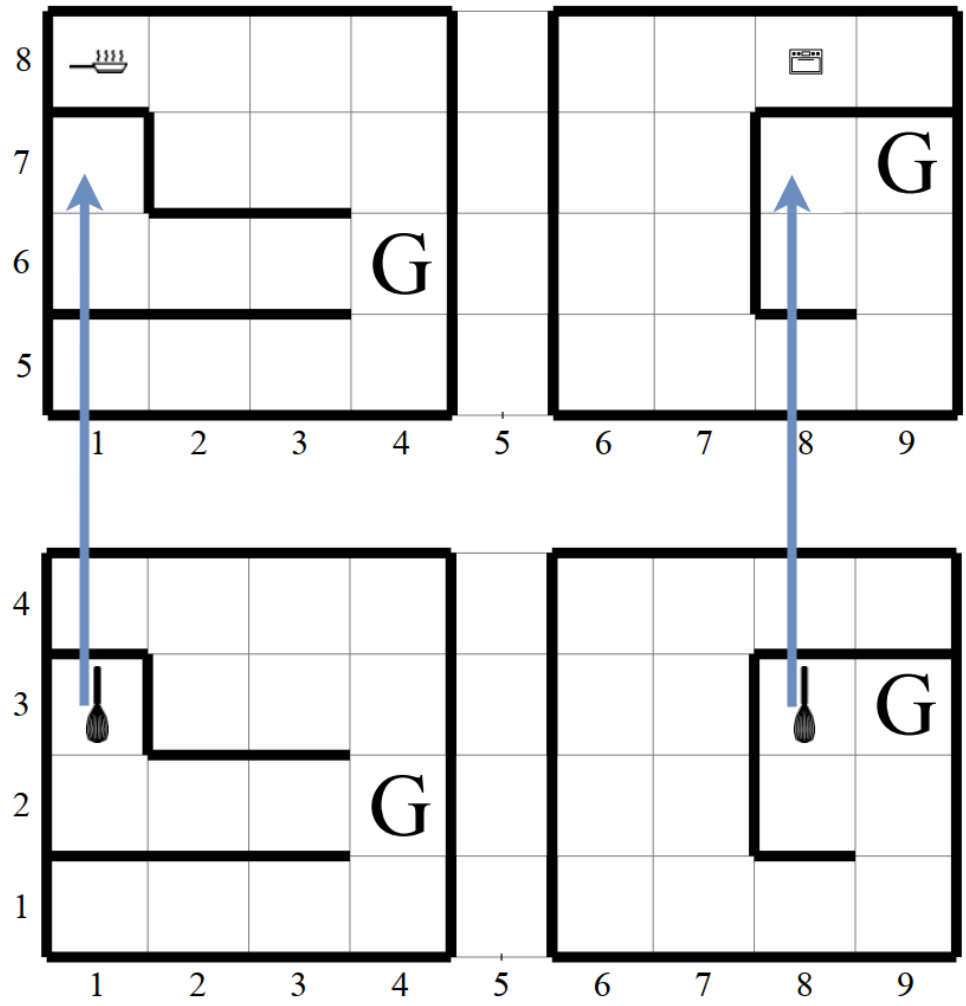

Figura 4: griglia complessiva. Una volta che l'agente avrà preso uno sbattitore potrà andare nella griglia 2 per cucinare (come indicato dalle frecce).

Non appena l'agente esegue l'azione **take_whiskr** da una delle due posizioni contenenti uno **sbattitore**, esso viene portato nella seconda griglia seguendo le frecce mostrate nella **figura 4**:

- Se l'agente prende lo sbattitore in posizione **(3, 1)** nella griglia in basso, esso viene portato nella griglia in alto in posizione **(7, 1)**
- Se l'agente prende lo sbattitore in posizione **(3, 8)** nella griglia in basso, esso viene portato nella griglia in alto in posizione **(7, 8)**

Dopo aver raggiunto e **preso uno sbattitore**, l'agente deve raggiungere una delle due postazioni di cucina che sono nella griglia superiore (situati nelle posizioni **(8, 1)** e **(8, 8)** nella griglia in alto) per cucinare le uova (la postazione di cucina da raggiungere dipende dalla ricetta che l'agente deve cucinare)

L'agente dovrà percorrere entrambe le griglie per raggiungere l'obiettivo finale. Dalla prima griglia alla seconda griglia.

Nella seconda griglia (griglia situata in alto), l'agente ha l'obiettivo di raggiungere una delle due postazioni di cucina e di eseguire quindi l'azione **cook**:

- la postazione di cucina per il piatto **scrambled eggs** è situato in posizione **(8, 1)**
- la postazione di cucina per il piatto **pudding eggs** è situato in posizione **(8, 8)**

In entrambe le griglie sono situati dei **Gate** che hanno la funzione di trasportare l'agente da una stanza all'altra delle stessa griglia, l'agente quando si trova su uno di questi può usare il **Gate** tramite l'azione **go_rigth** o **go_left**:

- la postazione di **Gate** situato in posizione **(2, 4)** trasporta l'agente in posizione **(3, 9)** se effettua l'azione **go_right**

- la postazione di **Gate** situato in posizione **(3, 9)** trasporta l'agente in posizione **(2, 4)** se effettua l'azione **go_left**
- la postazione di **Gate** situato in posizione **(6, 4)** trasporta l'agente in posizione **(7, 9)** se effettua l'azione **go_right**
- la postazione di **Gate** situato in posizione **(7, 9)** trasporta l'agente in posizione **(6, 4)** se effettua l'azione **go_left**

Definiamo tutti i possibili stati

Ogni stato del sistema corrisponde a una posizione possibile che l'agente può assumere nella griglia.

Tutte le posizione della **colonna 5** che non possono essere fisicamente raggiunte non saranno presenti tra gli stati possibili del problema (posizioni impossibili).

È stata creata una classe denominata **World** per modellare il mondo.

Vediamo una rappresentazione in Tupla posizionale del mondo.

In [2]:
```
world = World()
world.print_states(True)
```

```
(8, 1)  (8, 2)  (8, 3)  (8, 4)      (8, 6)  (8, 7)  (8, 8)  (8, 9)
(7, 1)  (7, 2)  (7, 3)  (7, 4)      (7, 6)  (7, 7)  (7, 8)  (7, 9)
(6, 1)  (6, 2)  (6, 3)  (6, 4)      (6, 6)  (6, 7)  (6, 8)  (6, 9)
(5, 1)  (5, 2)  (5, 3)  (5, 4)      (5, 6)  (5, 7)  (5, 8)  (5, 9)


(4, 1)  (4, 2)  (4, 3)  (4, 4)      (4, 6)  (4, 7)  (4, 8)  (4, 9)
(3, 1)  (3, 2)  (3, 3)  (3, 4)      (3, 6)  (3, 7)  (3, 8)  (3, 9)
(2, 1)  (2, 2)  (2, 3)  (2, 4)      (2, 6)  (2, 7)  (2, 8)  (2, 9)
(1, 1)  (1, 2)  (1, 3)  (1, 4)      (1, 6)  (1, 7)  (1, 8)  (1, 9)
```

Vediamo una rappresentazione dei stati in versione numerica.

In [3]:
```
world.print_states(False)
```

```
56  57  58  59      60  61  62  63
48  49  50  51      52  53  54  55
40  41  42  43      44  45  46  47
32  33  34  35      36  37  38  39


24  25  26  27      28  29  30  31
16  17  18  19      20  21  22  23
8   9   10  11      12  13  14  15
0   1   2   3       4   5   6   7
```

Per poter lavorare più agevolmente è stato creato un Oggetto **State** che modella uno stato del mondo.

```
class State:
    def __init__(self, number, position):
        self.number = number
        self.position = position

    def __str__(self):
        return f"State{{number: {self.number}, position: {self.position}}}"

    def __repr__(self):
        return self.__str__()

    def __eq__(self, other):
        if not isinstance(other, State):
            return False
        return self.number == other.number and self.position == other.position
```

Il numero totale di Stati del mondo è 64.

In [4]:
```
print(len(world.S))
world.S
```

```
64
```

Out[4]:
```
[State{number: 0, position: (1, 1)},
 State{number: 1, position: (1, 2)},
 State{number: 2, position: (1, 3)},
 State{number: 3, position: (1, 4)},
 State{number: 4, position: (1, 6)},
 State{number: 5, position: (1, 7)},
 State{number: 6, position: (1, 8)},
 State{number: 7, position: (1, 9)},
 State{number: 8, position: (2, 1)},
 State{number: 9, position: (2, 2)},
 State{number: 10, position: (2, 3)},
 State{number: 11, position: (2, 4)},
 State{number: 12, position: (2, 6)},
 State{number: 13, position: (2, 7)},
 State{number: 14, position: (2, 8)},
 State{number: 15, position: (2, 9)},
 State{number: 16, position: (3, 1)},
 State{number: 17, position: (3, 2)},
 State{number: 18, position: (3, 3)},
 State{number: 19, position: (3, 4)},
 State{number: 20, position: (3, 6)},
 State{number: 21, position: (3, 7)},
 State{number: 22, position: (3, 8)},
 State{number: 23, position: (3, 9)},
 State{number: 24, position: (4, 1)},
 State{number: 25, position: (4, 2)},
 State{number: 26, position: (4, 3)},
 State{number: 27, position: (4, 4)},
 State{number: 28, position: (4, 6)},
 State{number: 29, position: (4, 7)},
 State{number: 30, position: (4, 8)},
 State{number: 31, position: (4, 9)},
 State{number: 32, position: (5, 1)},
 State{number: 33, position: (5, 2)},
 State{number: 34, position: (5, 3)},
 State{number: 35, position: (5, 4)},
 State{number: 36, position: (5, 6)},
 State{number: 37, position: (5, 7)},
 State{number: 38, position: (5, 8)},
 State{number: 39, position: (5, 9)},
 State{number: 40, position: (6, 1)},
 State{number: 41, position: (6, 2)},
 State{number: 42, position: (6, 3)},
 State{number: 43, position: (6, 4)},
 State{number: 44, position: (6, 6)},
 State{number: 45, position: (6, 7)},
 State{number: 46, position: (6, 8)},
 State{number: 47, position: (6, 9)},
 State{number: 48, position: (7, 1)},
 State{number: 49, position: (7, 2)},
 State{number: 50, position: (7, 3)},
 State{number: 51, position: (7, 4)},
 State{number: 52, position: (7, 6)},
 State{number: 53, position: (7, 7)},
 State{number: 54, position: (7, 8)},
 State{number: 55, position: (7, 9)},
 State{number: 56, position: (8, 1)},
 State{number: 57, position: (8, 2)},
 State{number: 58, position: (8, 3)},
 State{number: 59, position: (8, 4)},
 State{number: 60, position: (8, 6)},
 State{number: 61, position: (8, 7)},
 State{number: 62, position: (8, 8)},
 State{number: 63, position: (8, 9)}]
```

Per poter generare tutti e 64 gli stati sono stati implementate la seguente funzione:

```python
def __build_states(self):
    states = []
    count = 0
    for i in range(1, 9):
        for j in [x for x in range(1, 10) if x != 5]:
            states.append(State(count, (i, j)))
```

```
            count += 1
    return states
```

Definizioni delle azioni Possibili

L'agente può effettuare le **azioni di movimento** (**left**, **right**, **up**, **down**) e inoltre può effettuare alcune azioni specifiche per specifichi stati:

- L'azione **take_whiskr** corrisponde all'azione di prendere lo sbattitore. A seguito della sua applicazione, l'agente viene portato nella seconda griglia.
- L'azione **go_right** corrisponde all'azione di utilizzo del Gate posizionato nella stanza di sinistra. A seguito della sua applicazione, l'agente viene portato nella seconda stanza dalle prima stanza.
- L'azione **go_left** corrisponde all'azione di utilizzo del Gate posizionato nella stanza di destra. A seguito della sua applicazione, l'agente viene portato nella prima stanza dalla seconda stanza.
- L'azione **cook** corrisponde all'azione di cucinare le uova. In questo caso, l'applicazione di questa azione porta l'agente a rimanere fermo (absorbing state). L'esecuzione di questa azione rappresenta l'obiettivo finale dell'agente

L'insieme **A** è il seguente:

- azione 0: **left** ←
- azione 1: **right** →
- azione 2: **up** ↑
- azione 3: **down** ↓
- azione 4: **take_whiskr T**
- azione 5: **go_right R**
- azione 6: **go_left L**
- azione 7: **cook C**

Per poter lavorare più agevolmente è stato creato un Oggetto **Action** che modella un azione possibile nel sistema

```
class Action:
    def __init__(self, name, number, function, ascii):
        if not callable(function):
            raise ValueError("La funzione deve essere callable")
        self.name = name
        self.number = number
        self.function = function
        self.ascii = ascii

    def __str__(self):
        function_name = self.function.__name__ if hasattr(self.function, '__name__') else
'funzione anonima'
        return f"Action{{name: {self.name}, number: {self.number}, function: {function_name},
ascii: {self.ascii}}}"

    def __repr__(self):
        return self.__str__()
```

Ogni azione ha le seguenti funzioni che prendo in input un **State** e ritorna uno **State**

```
def action_left(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 0 in self.am[s.number]:
        title = (state.position[0], state.position[1] - 1)
    return self.state_position_dict[title]

def action_right(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
```

```
    if 1 in self.am[s.number]:
        title = (state.position[0], state.position[1] + 1)
    return self.state_position_dict[title]

def action_up(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 2 in self.am[s.number]:
        title = (state.position[0] + 1, state.position[1])
    return self.state_position_dict[title]

def action_down(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 3 in self.am[s.number]:
        title = (state.position[0] - 1, state.position[1])
    return self.state_position_dict[title]

def action_take_whiskr(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 4 in self.am[s.number]:
        if s.number == 16:
            title = self.state_dict[48].position
        elif s.number == 22:
            title = self.state_dict[54].position
    return self.state_position_dict[title]

def action_go_right(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 5 in self.am[s.number]:
        if s.number == 11:
            title = self.state_dict[23].position
        elif s.number == 43:
            title = self.state_dict[55].position
    return self.state_position_dict[title]

def action_go_left(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 6 in self.am[s.number]:
        if s.number == 23:
            title = self.state_dict[11].position
        elif s.number == 55:
            title = self.state_dict[43].position
    return self.state_position_dict[title]

def action_cook(self, s: State):
    state = self.state_dict[s.number]
    if state is None:
        raise RuntimeError("Non trovato")
    title = state.position
    if 7 in self.am[s.number]:
        title = state.position
    return self.state_position_dict[title]
```

Di seguito la funzione che genera gli oggetti Azione:

```
def __build_actions(self):
    actions = [Action('left', 0, self.action_left, '←'),
               Action('right', 1, self.action_right, '→'),
               Action('up', 2, self.action_up, '↑'),
               Action('down', 3, self.action_down, '↓'),
               Action('take_whiskr', 4, self.action_take_whiskr, 'T'),
               Action('go_right', 5, self.action_go_right, 'R'),
               Action('go_left', 6, self.action_go_left, 'L'),
               Action('cook', 7, self.action_cook, 'C')]
    return actions
```

In [5]:
```
world.A
```

Out[5]:
```
[Action{name: left, number: 0, function: action_left, ascii: ←},
 Action{name: right, number: 1, function: action_right, ascii: →},
 Action{name: up, number: 2, function: action_up, ascii: ↑},
 Action{name: down, number: 3, function: action_down, ascii: ↓},
 Action{name: take_whiskr, number: 4, function: action_take_whiskr, ascii: T},
 Action{name: go_right, number: 5, function: action_go_right, ascii: R},
 Action{name: go_left, number: 6, function: action_go_left, ascii: L},
 Action{name: cook, number: 7, function: action_cook, ascii: C}]
```

Costruiamo ora una mappa delle azioni possibili per ogni stato. In questo caso modelliamo questa mappa per permettere entrambe le ricette contemporaneamente, ma in seguito se decidiamo una ricetta elimineremo l'azione di cucinare in funzione della ricetta:

- Se si decide di cucinare **scrambled eggs (8, 1)** allora l'azione **cook** allo stato **62 (8,8)** verrà eliminata dalla mappa
- Se si decide di cucinare **pudding eggs (8,8)** allora l'azione **cook** allo stato **56 (8,1)** verrà eliminata dalla mappa

Un'altra alternativa alla rimozione dell'azione **cook** è quella di impostare una ricompensa estremamente negativa e svantaggiosa, così da non farla prediligere rispetto alle altre azioni possibili.

Vediamo le azioni possibili per ogni stato del sistema.

In [6]:
```
world.am
```

Out[6]:
```
{0: [1],
 1: [0, 1],
 2: [0, 1],
 3: [0, 2],
 4: [1, 2],
 5: [0, 1, 2],
 6: [0, 1],
 7: [0, 2],
 8: [1, 2],
 9: [0, 1],
 10: [0, 1],
 11: [0, 2, 3, 5],
 12: [1, 2, 3],
 13: [0, 2, 3],
 14: [1, 2],
 15: [0, 2, 3],
 16: [3, 4],
 17: [1, 2],
 18: [0, 1, 2],
 19: [0, 2, 3],
 20: [1, 2, 3],
 21: [0, 2, 3],
 22: [1, 3, 4],
 23: [0, 3, 6],
 24: [1],
 25: [0, 1, 3],
 26: [0, 1, 3],
 27: [0, 3],
 28: [1, 3],
 29: [0, 1, 3],
 30: [0, 1],
 31: [0],
 32: [1],
 33: [0, 1],
 34: [0, 1],
 35: [0, 2],
 36: [1, 2],
 37: [0, 1, 2],
 38: [0, 1],
 39: [0, 2],
 40: [1, 2],
 41: [0, 1],
 42: [0, 1],
 43: [0, 2, 3, 5],
 44: [1, 2, 3],
 45: [0, 2, 3],
 46: [1, 2],
 47: [0, 2, 3],
 48: [3],
 49: [1, 2],
 50: [0, 1, 2],
 51: [0, 2, 3],
 52: [1, 2, 3],
 53: [0, 2, 3],
 54: [1, 3],
 55: [0, 3, 6],
 56: [1, 7],
 57: [0, 1, 3],
 58: [0, 1, 3],
 59: [0, 3],
 60: [1, 3],
 61: [0, 1, 3],
 62: [0, 1, 7],
 63: [0]}
```

Creazione insieme di Transizioni P

d) Report the transition function P for an state s and action a in a tabular format.

Creiamo la matrice a 3 dimensioni di transizione $P[A, S, S']$:

- A: sono le Azioni

- S: sono gli stati sorgente
- S': sono gli stati di destinazione

Modelliamo il problema assumendo che non ci sono effetti imprevisti a seguito dell'applicazione delle azioni:

da uno stato **sorgente** applicando l'**azione**, abbiamo soltanto **uno stato destinazione** (stato raggiungibile con probabilità **1**)

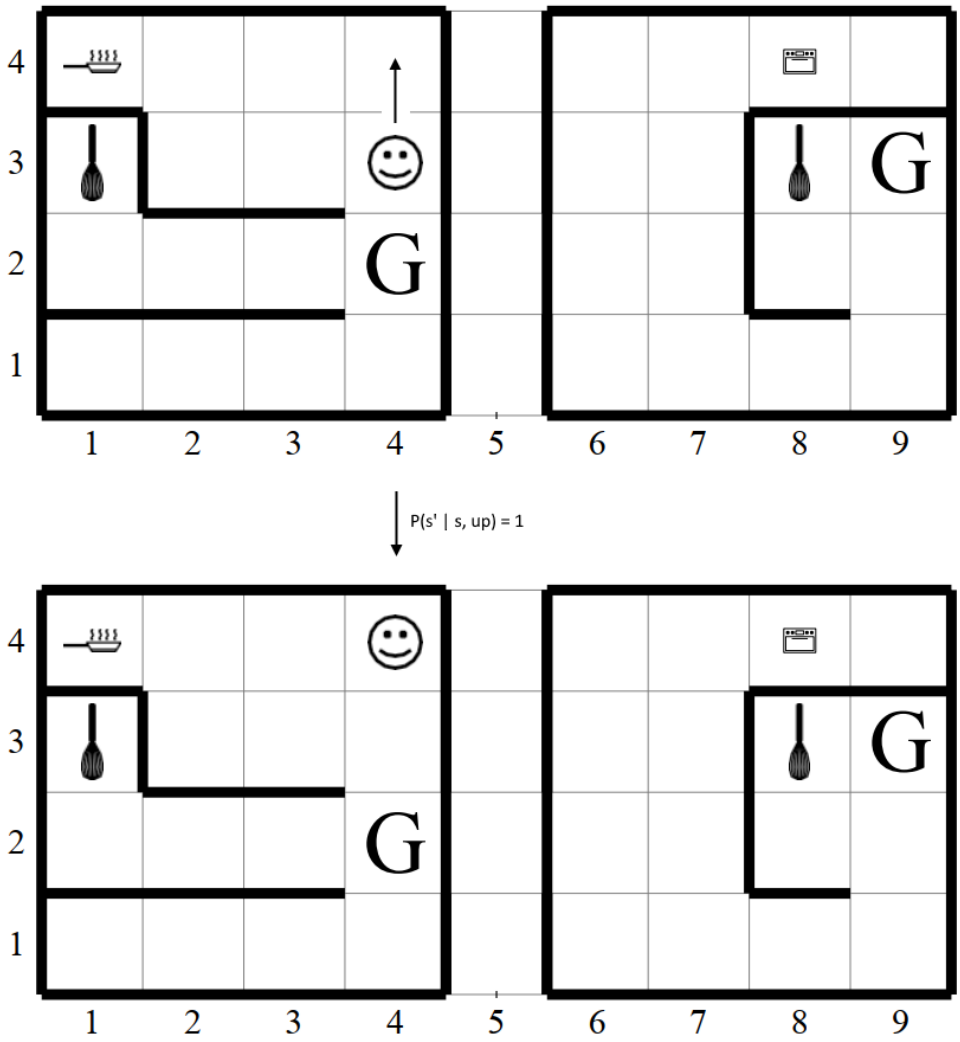

Figura 5: Esempio di comportamento dell'agente a seguito dell'applicazione dell'azione **up**

Vediamo il codice usato per generare la funzione di Transizione. La funzione generatrice è **__generate_transitions()** contenuta nella classe **World**

```
def __generate_transitions(self):
    s,  a, mapping = self.__convert_s_a_mapping_to_numpy_arrays()

    transitions = np.zeros(
        shape=(len(self.A), len(self.S), len(self.S)), dtype=np.float64
    )

    for state in self.S:
        for action in self.__get_possible_actions_from_state(state.number, mapping):
            if mapping[state.number, action.number] > 0.0:
                # l'azione `action` è possibile dallo stato `state`
                # non ci sono effetti imprevisti, c'è solo uno stato destinazione
                # raggiungibile con probabilità 1
                transitions[action.number, state.number, action.function(state).number] = 1

    return transitions

def __get_possible_actions_from_state(self, s: int, mapping: np.ndarray):
```

```
        actions = []
        for a in self.A:
            if mapping[s, a.number] > 0.0:
                actions.append(a)
        return actions

    def __convert_to_numpy_array(self, mapping: dict, s: np.ndarray, a: np.ndarray):
        count = 0
        for i in sorted(list(mapping.keys())):
            if i != count:
                raise RuntimeError("Struttura non valida")
            count += 1
        array = np.zeros(shape=(len(s), len(a)), dtype=np.float64)
        for s, actions in mapping.items():
            prob = 1 / len(actions)
            for a in actions:
                array[s, a] = prob
        return array

    def __convert_s_a_mapping_to_numpy_arrays(self):
        s = np.array(list(self.state_dict.keys()), dtype=np.int64)
        a = np.array([0, 1, 2, 3, 4, 5, 6, 7], dtype=np.int64)
        mapping = self.__convert_to_numpy_array(self.am, s, a)
        return s, a, mapping
```

In [7]:
```
world.P
```

Out[7]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [1., 0., 0., ..., 0., 0., 0.],
        [0., 1., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 1., 0., 0.],
        [0., 0., 0., ..., 0., 1., 0.]],

       [[0., 1., 0., ..., 0., 0., 0.],
        [0., 0., 1., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 1., 0.],
        [0., 0., 0., ..., 0., 0., 1.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 1., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]])
```

Costruiamo le varie matrici di transizione P per ogni azione specificata.

Stampiamo una tabella che descrive una matrice SXS' per l'azione scelta. $P[left, S, S']$

In [8]:
```
transition_left = world.print_transitions_matrix(action='left')
transition_left.to_csv("transitions/transition_left.csv")
transition_left
```

Out[8]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

64 rows × 64 columns

$P[right, S, S']$

In [9]:
```
transition_right = world.print_transitions_matrix(action='right')
transition_right.to_csv("transitions/transition_right.csv")
transition_right
```

Out[9]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[up, S, S']$

In [10]:
```
transition_up = world.print_transitions_matrix(action='up')
transition_up.to_csv("transitions/transition_up.csv")
transition_up
```

Out[10]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[down, S, S']$

In [11]:
```
transition_down = world.print_transitions_matrix(action="down")
transition_down.to_csv("transitions/transition_down.csv")
transition_down
```

Out[11]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[take_whiskr, S, S']$

In [12]:
```
transition_take_whiskr = world.print_transitions_matrix(action='take_whiskr')
transition_take_whiskr.to_csv("transitions/transition_take_whiskr.csv")
transition_take_whiskr
```

Out[12]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[go_right, S, S']$

In [13]:
```
transition_go_right = world.print_transitions_matrix(action='go_right')
transition_go_right.to_csv("transitions/transition_go_right.csv")
transition_go_right
```

Out[13]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[go_left, S, S']$

In [14]:
```
transition_go_left = world.print_transitions_matrix(action='go_left')
transition_go_left.to_csv("transitions/transition_go_left.csv")
transition_go_left
```

Out[14]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

$P[cook, S, S']$

In [15]:
```
transition_cook = world.print_transitions_matrix(action='cook')
transition_cook.to_csv("transitions/transition_cook.csv")
transition_cook
```

Out[15]:

Stati	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	...	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
s0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
s59	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s61	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
s63	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

64 rows × 64 columns

Creazione Insieme di reward R

e) Describe a reward function $R : S \times A \times S$ and a value of γ that will lead to an optimal policy

Il nostro obbiettivo è cucinare il piatto **pudding eggs**.

Un possibile approccio alla scelta dei rewards da assegnare all'agente per guidarlo nella risoluzione del problema è il seguente:

- Premio per l'obbiettivo $+100$ cucinare il piatto **pudding eggs** allo stato 62
- Premio per l'obbiettivo intermedio $+50$ prendere uno sbattitore posizionato allo stato **16** o **22**

- Premio per l'utilizzo del gate che porta a destra (verso il goal) $+1$
- Penalizzazione per un azione non valida -1
- nei casi rimanenti il Premio sarà 0

In [16]:
```
# caso cook (goal)
print(world.R[62, 7, 62])
# caso cook sbagliato
print(world.R[56, 7, 56])
# caso take_whiskr
print(world.R[16, 4, 48])
# caso take_whiskr
print(world.R[22, 4, 54])
# caso non valido
# lo stato 22 ha le seguenti azioni: [1, 3, 4]
world.R[22, 2, 22]
```

```
100.0
-10.0
50.0
50.0
```

Out[16]: -1.0

Vediamo il codice usato per generare le reward R. La funzione generatrice è **__generate_transitions()** contenuta nella classe **World**

```
def __generate_rewards(self):
    s, a, mapping = self.__convert_s_a_mapping_to_numpy_arrays()
    rewards = np.zeros(shape=(len(self.S), len(self.A), len(self.S)), dtype=np.float64)
    rewards[62, self.__get_number_action('cook'), 62] = 1000
    rewards[56, self.__get_number_action('cook'), 56] = -10

    rewards[16, self.__get_number_action('take_whiskr'), 48] = 500
    rewards[22, self.__get_number_action('take_whiskr'), 54] = 500

    rewards[11, self.__get_number_action('go_right'), 23] = 1
    rewards[43, self.__get_number_action('go_right'), 55] = 1

    for s in self.S:
        for a in self.A:
            # settiamo il reward a -1 se l'azione `a` non è valida
            # nello stato `s`
            # (ovvero quando non è possibile eseguire l'azione
            # `a` dallo stato `s`)
            if self.__get_probs(mapping, a.number, s.number) == 0.0:
                rewards[s.number, a.number, s.number] = -10
    return rewards
```

implementazione scenario Model free

h) Now, considering the problem as a model-free scenario, provide a program (written in Python, possibly based on the labs) that can compute the optimal policy for this world by solely considering the pudding eggs scenario. Draw the computed policy in the grid by putting the optimal action in each cell. If multiple actions are possible, include the probability of each arrow. There may be multiple optimal policies; pick one to show it. Note that the model is not available for computation but must be encoded to be used as the "real-world" environment.

In questo caso, non sarà possibile accedere al **transition model** e al **reward model** del problema.

Per la specifica **Model-free**, consideriamo soltanto lo scenario **pudding eggs**.

Definiamo una funzione di policy iniziale insieme a una funzione di generazione di episodi. Useremo e adatteremo il codice sviluppato durante il laboratorio.

In [17]:
```
def simple_policy(s: State):
    return world.am[s.number][0]
```

Definiamo una funzione che ci genererà gli episodi

```
def generate_episode(self, length: int, return_actions=False, first_state=None,
first_action=None,
                     random_first_state=False):
    next_state = self.starting_state if first_state == None else first_state
    if random_first_state:
        while True:
            next_state = np.random.choice(self.states)
            if 0 <= next_state.number <= 32:
                break
    actions = []
    episode = [next_state.number]

    if first_action != None:
        actions.append(first_action)
        next_state = first_action.function(next_state)
        episode.append(next_state.number)

    while len(episode) < length:
        if next_state.number not in [s.number for s in self.states]:
            self.states.append(next_state)
            for action in self.actions:
                self.policy[(next_state.number, action.name)] = 0
            self.policy[(next_state.number,
self.actions_number_dict[self.policy_function(next_state)].name)] = 1

        action_probabilities = [self.policy[(next_state.number, action.name)] for action in
self.actions]
        action = np.random.choice(list(self.actions), p=action_probabilities)
        actions.append(action)
        next_state = action.function(next_state)
        episode.append(next_state.number)

        if next_state.number == 62 and action.number == 7:
            break

    actions.append(self.step())
    self.reset_state()

    state_action = []
    for i in range(len(episode)):
        state_action.append({'s': episode[i], 'a': actions[i]})

    return episode if not return_actions else state_action
```

Definiamo una funzione che ci calcola una policy ottima. Utilizzeremo la funzione vista in laboratorio opportunamente modificata, come richiesto.

```
def compute_returns(self, episode: list, index: int, gamma: float):
    if index is None or index >= len(episode):
        return 0

    sum_returns = 0
    for i in range(len(episode) - index):
        state = self.env_representation.state_dict[episode[index + i]['s']]
        reward = self.env_representation.R[
            episode[index + i]['s'], episode[index + i]['a'].number, episode[index + i]
['a'].function(state).number]
        sum_returns += reward * (gamma ** i)

    return sum_returns

def improve_policy(self, iteration: int, gamma=None):
    if gamma is None:
        gamma = self.discount_factor
    n = {(state.number, action.name): 0 for state in self.states for action in self.actions}
    g = {(state.number, action.name): 0 for state in self.states for action in self.actions}
    q = {(state.number, action.name): 0 for state in self.states for action in self.actions}
```

```
        known_states = [state.number for state in self.states]

        for i in range(iteration):
            print(f"Iterazione: {i}/{iteration}")
            for state in self.states:
                for action in self.actions:
                    epi = self.generate_episode(100, return_actions=True, first_state=state,
                                                first_action=action)
                    episode, actions = self.__get_state_action_from_episodes(epi)
                    self.reset_state()

                    # check if all states are known
                    for state_no in episode:
                        if state_no not in known_states:
                            known_states.append(state_no)
                            for action in self.actions:
                                n[(state_no, action.name)] = 0
                                g[(state_no, action.name)] = 0
                                q[(state_no, action.name)] = 0

                    ## COMPUTATION OF Q-VALUES
                    for j in range(len(episode)):
                        n[(episode[j], actions[j].name)] += 1
                        g[(episode[j], actions[j].name)] += self.compute_returns(epi, j, gamma)
                        q[(episode[j], actions[j].name)] = g[(episode[j], actions[j].name)] / n[
                            (episode[j], actions[j].name)]
        ## POLICY IMPROVEMENT
        for state_no, ac in q.keys():
            if state_no not in [s.number for s in self.states]:
                self.states.append(self.env_representation.state_dict[state_no])

            max_action = None
            max_value = -np.inf
            for action in self.actions:
                if q[(state_no, action.name)] > max_value:
                    max_value = q[(state_no, action.name)]
                    max_action = action.name

            new_policy = self.policy.copy()
            for action in self.actions:
                new_policy[(state_no, action.name)] = 1 if action.name == max_action else 0

            self.change_policy(new_policy)
        return self.policy
```

Il parametro (**discount factor**/**gamma**) descrive quanto l'agente si preoccupa dei rewards futuri. Nel caso in cui γ sia **0**, l'agente tiene conto soltanto dei rewards immediati. Più è vicino ad **1** più l'agente è interessato ai rewards futuri. Vediamo il risultato con un **gamma** impostato a 0.5 Quindi abbiamo un equilibrio tra i rewards immediati e Futuri.

In [18]:
```
agent = Agent(simple_policy)
agent.improve_policy(10)
```

```
Iterazione: 0/10
Iterazione: 1/10
Iterazione: 2/10
Iterazione: 3/10
Iterazione: 4/10
Iterazione: 5/10
Iterazione: 6/10
Iterazione: 7/10
Iterazione: 8/10
Iterazione: 9/10
```

Out[18]:
```
{(19, 'left'): 0,
 (19, 'right'): 0,
 (19, 'up'): 1,
 (19, 'down'): 0,
 (19, 'take_whiskr'): 0,
 (19, 'go_right'): 0,
 (19, 'go_left'): 0,
 (19, 'cook'): 0,
 (18, 'left'): 0,
 (18, 'right'): 1,
 (18, 'up'): 0,
 (18, 'down'): 0,
 (18, 'take_whiskr'): 0,
 (18, 'go_right'): 0,
 (18, 'go_left'): 0,
 (18, 'cook'): 0,
 (17, 'left'): 0,
 (17, 'right'): 0,
 (17, 'up'): 1,
 (17, 'down'): 0,
 (17, 'take_whiskr'): 0,
 (17, 'go_right'): 0,
 (17, 'go_left'): 0,
 (17, 'cook'): 0,
 (27, 'left'): 1,
 (27, 'right'): 0,
 (27, 'up'): 0,
 (27, 'down'): 0,
 (27, 'take_whiskr'): 0,
 (27, 'go_right'): 0,
 (27, 'go_left'): 0,
 (27, 'cook'): 0,
 (26, 'left'): 0,
 (26, 'right'): 1,
 (26, 'up'): 0,
 (26, 'down'): 0,
 (26, 'take_whiskr'): 0,
 (26, 'go_right'): 0,
 (26, 'go_left'): 0,
 (26, 'cook'): 0,
 (25, 'left'): 0,
 (25, 'right'): 1,
 (25, 'up'): 0,
 (25, 'down'): 0,
 (25, 'take_whiskr'): 0,
 (25, 'go_right'): 0,
 (25, 'go_left'): 0,
 (25, 'cook'): 0,
 (24, 'left'): 0,
 (24, 'right'): 1,
 (24, 'up'): 0,
 (24, 'down'): 0,
 (24, 'take_whiskr'): 0,
 (24, 'go_right'): 0,
 (24, 'go_left'): 0,
 (24, 'cook'): 0,
 (11, 'left'): 0,
 (11, 'right'): 0,
 (11, 'up'): 0,
 (11, 'down'): 0,
 (11, 'take_whiskr'): 0,
 (11, 'go_right'): 1,
 (11, 'go_left'): 0,
 (11, 'cook'): 0,
 (10, 'left'): 0,
 (10, 'right'): 1,
 (10, 'up'): 0,
 (10, 'down'): 0,
 (10, 'take_whiskr'): 0,
 (10, 'go_right'): 0,
 (10, 'go_left'): 0,
 (10, 'cook'): 0,
 (9, 'left'): 0,
 (9, 'right'): 1,
 (9, 'up'): 0,
```

```
(9, 'down'): 0,
(9, 'take_whiskr'): 0,
(9, 'go_right'): 0,
(9, 'go_left'): 0,
(9, 'cook'): 0,
(8, 'left'): 0,
(8, 'right'): 0,
(8, 'up'): 1,
(8, 'down'): 0,
(8, 'take_whiskr'): 0,
(8, 'go_right'): 0,
(8, 'go_left'): 0,
(8, 'cook'): 0,
(3, 'left'): 0,
(3, 'right'): 0,
(3, 'up'): 1,
(3, 'down'): 0,
(3, 'take_whiskr'): 0,
(3, 'go_right'): 0,
(3, 'go_left'): 0,
(3, 'cook'): 0,
(2, 'left'): 0,
(2, 'right'): 1,
(2, 'up'): 0,
(2, 'down'): 0,
(2, 'take_whiskr'): 0,
(2, 'go_right'): 0,
(2, 'go_left'): 0,
(2, 'cook'): 0,
(1, 'left'): 0,
(1, 'right'): 1,
(1, 'up'): 0,
(1, 'down'): 0,
(1, 'take_whiskr'): 0,
(1, 'go_right'): 0,
(1, 'go_left'): 0,
(1, 'cook'): 0,
(0, 'left'): 0,
(0, 'right'): 1,
(0, 'up'): 0,
(0, 'down'): 0,
(0, 'take_whiskr'): 0,
(0, 'go_right'): 0,
(0, 'go_left'): 0,
(0, 'cook'): 0,
(23, 'left'): 0,
(23, 'right'): 0,
(23, 'up'): 0,
(23, 'down'): 1,
(23, 'take_whiskr'): 0,
(23, 'go_right'): 0,
(23, 'go_left'): 0,
(23, 'cook'): 0,
(22, 'left'): 0,
(22, 'right'): 0,
(22, 'up'): 0,
(22, 'down'): 0,
(22, 'take_whiskr'): 1,
(22, 'go_right'): 0,
(22, 'go_left'): 0,
(22, 'cook'): 0,
(16, 'left'): 0,
(16, 'right'): 0,
(16, 'up'): 0,
(16, 'down'): 0,
(16, 'take_whiskr'): 1,
(16, 'go_right'): 0,
(16, 'go_left'): 0,
(16, 'cook'): 0,
(15, 'left'): 0,
(15, 'right'): 0,
(15, 'up'): 1,
(15, 'down'): 0,
(15, 'take_whiskr'): 0,
(15, 'go_right'): 0,
```

```
(15, 'go_left'): 0,
(15, 'cook'): 0,
(14, 'left'): 0,
(14, 'right'): 0,
(14, 'up'): 1,
(14, 'down'): 0,
(14, 'take_whiskr'): 0,
(14, 'go_right'): 0,
(14, 'go_left'): 0,
(14, 'cook'): 0,
(54, 'left'): 0,
(54, 'right'): 0,
(54, 'up'): 0,
(54, 'down'): 1,
(54, 'take_whiskr'): 0,
(54, 'go_right'): 0,
(54, 'go_left'): 0,
(54, 'cook'): 0,
(55, 'left'): 0,
(55, 'right'): 0,
(55, 'up'): 0,
(55, 'down'): 1,
(55, 'take_whiskr'): 0,
(55, 'go_right'): 0,
(55, 'go_left'): 0,
(55, 'cook'): 0,
(48, 'left'): 0,
(48, 'right'): 0,
(48, 'up'): 0,
(48, 'down'): 1,
(48, 'take_whiskr'): 0,
(48, 'go_right'): 0,
(48, 'go_left'): 0,
(48, 'cook'): 0,
(40, 'left'): 0,
(40, 'right'): 0,
(40, 'up'): 1,
(40, 'down'): 0,
(40, 'take_whiskr'): 0,
(40, 'go_right'): 0,
(40, 'go_left'): 0,
(40, 'cook'): 0,
(41, 'left'): 0,
(41, 'right'): 1,
(41, 'up'): 0,
(41, 'down'): 0,
(41, 'take_whiskr'): 0,
(41, 'go_right'): 0,
(41, 'go_left'): 0,
(41, 'cook'): 0,
(7, 'left'): 0,
(7, 'right'): 0,
(7, 'up'): 1,
(7, 'down'): 0,
(7, 'take_whiskr'): 0,
(7, 'go_right'): 0,
(7, 'go_left'): 0,
(7, 'cook'): 0,
(6, 'left'): 0,
(6, 'right'): 1,
(6, 'up'): 0,
(6, 'down'): 0,
(6, 'take_whiskr'): 0,
(6, 'go_right'): 0,
(6, 'go_left'): 0,
(6, 'cook'): 0,
(5, 'left'): 0,
(5, 'right'): 0,
(5, 'up'): 1,
(5, 'down'): 0,
(5, 'take_whiskr'): 0,
(5, 'go_right'): 0,
(5, 'go_left'): 0,
(5, 'cook'): 0,
(4, 'left'): 0,
```

```
(4, 'right'): 0,
(4, 'up'): 1,
(4, 'down'): 0,
(4, 'take_whiskr'): 0,
(4, 'go_right'): 0,
(4, 'go_left'): 0,
(4, 'cook'): 0,
(46, 'left'): 0,
(46, 'right'): 0,
(46, 'up'): 1,
(46, 'down'): 0,
(46, 'take_whiskr'): 0,
(46, 'go_right'): 0,
(46, 'go_left'): 0,
(46, 'cook'): 0,
(47, 'left'): 0,
(47, 'right'): 0,
(47, 'up'): 1,
(47, 'down'): 0,
(47, 'take_whiskr'): 0,
(47, 'go_right'): 0,
(47, 'go_left'): 0,
(47, 'cook'): 0,
(43, 'left'): 0,
(43, 'right'): 0,
(43, 'up'): 0,
(43, 'down'): 0,
(43, 'take_whiskr'): 0,
(43, 'go_right'): 1,
(43, 'go_left'): 0,
(43, 'cook'): 0,
(42, 'left'): 0,
(42, 'right'): 1,
(42, 'up'): 0,
(42, 'down'): 0,
(42, 'take_whiskr'): 0,
(42, 'go_right'): 0,
(42, 'go_left'): 0,
(42, 'cook'): 0,
(13, 'left'): 0,
(13, 'right'): 0,
(13, 'up'): 1,
(13, 'down'): 0,
(13, 'take_whiskr'): 0,
(13, 'go_right'): 0,
(13, 'go_left'): 0,
(13, 'cook'): 0,
(12, 'left'): 0,
(12, 'right'): 0,
(12, 'up'): 1,
(12, 'down'): 0,
(12, 'take_whiskr'): 0,
(12, 'go_right'): 0,
(12, 'go_left'): 0,
(12, 'cook'): 0,
(39, 'left'): 0,
(39, 'right'): 0,
(39, 'up'): 1,
(39, 'down'): 0,
(39, 'take_whiskr'): 0,
(39, 'go_right'): 0,
(39, 'go_left'): 0,
(39, 'cook'): 0,
(38, 'left'): 0,
(38, 'right'): 1,
(38, 'up'): 0,
(38, 'down'): 0,
(38, 'take_whiskr'): 0,
(38, 'go_right'): 0,
(38, 'go_left'): 0,
(38, 'cook'): 0,
(37, 'left'): 0,
(37, 'right'): 0,
(37, 'up'): 1,
(37, 'down'): 0,
```

```
(37, 'take_whiskr'): 0,
(37, 'go_right'): 0,
(37, 'go_left'): 0,
(37, 'cook'): 0,
(36, 'left'): 0,
(36, 'right'): 0,
(36, 'up'): 1,
(36, 'down'): 0,
(36, 'take_whiskr'): 0,
(36, 'go_right'): 0,
(36, 'go_left'): 0,
(36, 'cook'): 0,
(51, 'left'): 0,
(51, 'right'): 0,
(51, 'up'): 1,
(51, 'down'): 0,
(51, 'take_whiskr'): 0,
(51, 'go_right'): 0,
(51, 'go_left'): 0,
(51, 'cook'): 0,
(50, 'left'): 0,
(50, 'right'): 1,
(50, 'up'): 0,
(50, 'down'): 0,
(50, 'take_whiskr'): 0,
(50, 'go_right'): 0,
(50, 'go_left'): 0,
(50, 'cook'): 0,
(49, 'left'): 0,
(49, 'right'): 0,
(49, 'up'): 1,
(49, 'down'): 0,
(49, 'take_whiskr'): 0,
(49, 'go_right'): 0,
(49, 'go_left'): 0,
(49, 'cook'): 0,
(35, 'left'): 0,
(35, 'right'): 0,
(35, 'up'): 1,
(35, 'down'): 0,
(35, 'take_whiskr'): 0,
(35, 'go_right'): 0,
(35, 'go_left'): 0,
(35, 'cook'): 0,
(34, 'left'): 0,
(34, 'right'): 1,
(34, 'up'): 0,
(34, 'down'): 0,
(34, 'take_whiskr'): 0,
(34, 'go_right'): 0,
(34, 'go_left'): 0,
(34, 'cook'): 0,
(33, 'left'): 0,
(33, 'right'): 1,
(33, 'up'): 0,
(33, 'down'): 0,
(33, 'take_whiskr'): 0,
(33, 'go_right'): 0,
(33, 'go_left'): 0,
(33, 'cook'): 0,
(32, 'left'): 0,
(32, 'right'): 1,
(32, 'up'): 0,
(32, 'down'): 0,
(32, 'take_whiskr'): 0,
(32, 'go_right'): 0,
(32, 'go_left'): 0,
(32, 'cook'): 0,
(21, 'left'): 0,
(21, 'right'): 0,
(21, 'up'): 1,
(21, 'down'): 0,
(21, 'take_whiskr'): 0,
(21, 'go_right'): 0,
(21, 'go_left'): 0,
```

```
(21, 'cook'): 0,
(20, 'left'): 0,
(20, 'right'): 0,
(20, 'up'): 1,
(20, 'down'): 0,
(20, 'take_whiskr'): 0,
(20, 'go_right'): 0,
(20, 'go_left'): 0,
(20, 'cook'): 0,
(45, 'left'): 0,
(45, 'right'): 0,
(45, 'up'): 1,
(45, 'down'): 0,
(45, 'take_whiskr'): 0,
(45, 'go_right'): 0,
(45, 'go_left'): 0,
(45, 'cook'): 0,
(44, 'left'): 0,
(44, 'right'): 0,
(44, 'up'): 1,
(44, 'down'): 0,
(44, 'take_whiskr'): 0,
(44, 'go_right'): 0,
(44, 'go_left'): 0,
(44, 'cook'): 0,
(59, 'left'): 1,
(59, 'right'): 0,
(59, 'up'): 0,
(59, 'down'): 0,
(59, 'take_whiskr'): 0,
(59, 'go_right'): 0,
(59, 'go_left'): 0,
(59, 'cook'): 0,
(58, 'left'): 0,
(58, 'right'): 1,
(58, 'up'): 0,
(58, 'down'): 0,
(58, 'take_whiskr'): 0,
(58, 'go_right'): 0,
(58, 'go_left'): 0,
(58, 'cook'): 0,
(57, 'left'): 0,
(57, 'right'): 1,
(57, 'up'): 0,
(57, 'down'): 0,
(57, 'take_whiskr'): 0,
(57, 'go_right'): 0,
(57, 'go_left'): 0,
(57, 'cook'): 0,
(56, 'left'): 0,
(56, 'right'): 1,
(56, 'up'): 0,
(56, 'down'): 0,
(56, 'take_whiskr'): 0,
(56, 'go_right'): 0,
(56, 'go_left'): 0,
(56, 'cook'): 0,
(29, 'left'): 0,
(29, 'right'): 0,
(29, 'up'): 0,
(29, 'down'): 1,
(29, 'take_whiskr'): 0,
(29, 'go_right'): 0,
(29, 'go_left'): 0,
(29, 'cook'): 0,
(28, 'left'): 0,
(28, 'right'): 0,
(28, 'up'): 0,
(28, 'down'): 1,
(28, 'take_whiskr'): 0,
(28, 'go_right'): 0,
(28, 'go_left'): 0,
(28, 'cook'): 0,
(53, 'left'): 0,
(53, 'right'): 0,
```

```
(53, 'up'): 1,
(53, 'down'): 0,
(53, 'take_whiskr'): 0,
(53, 'go_right'): 0,
(53, 'go_left'): 0,
(53, 'cook'): 0,
(52, 'left'): 0,
(52, 'right'): 0,
(52, 'up'): 1,
(52, 'down'): 0,
(52, 'take_whiskr'): 0,
(52, 'go_right'): 0,
(52, 'go_left'): 0,
(52, 'cook'): 0,
(30, 'left'): 0,
(30, 'right'): 1,
(30, 'up'): 0,
(30, 'down'): 0,
(30, 'take_whiskr'): 0,
(30, 'go_right'): 0,
(30, 'go_left'): 0,
(30, 'cook'): 0,
(61, 'left'): 0,
(61, 'right'): 0,
(61, 'up'): 0,
(61, 'down'): 1,
(61, 'take_whiskr'): 0,
(61, 'go_right'): 0,
(61, 'go_left'): 0,
(61, 'cook'): 0,
(60, 'left'): 0,
(60, 'right'): 0,
(60, 'up'): 0,
(60, 'down'): 1,
(60, 'take_whiskr'): 0,
(60, 'go_right'): 0,
(60, 'go_left'): 0,
(60, 'cook'): 0,
(31, 'left'): 1,
(31, 'right'): 0,
(31, 'up'): 0,
(31, 'down'): 0,
(31, 'take_whiskr'): 0,
(31, 'go_right'): 0,
(31, 'go_left'): 0,
(31, 'cook'): 0,
(62, 'left'): 0,
(62, 'right'): 0,
(62, 'up'): 0,
(62, 'down'): 0,
(62, 'take_whiskr'): 0,
(62, 'go_right'): 0,
(62, 'go_left'): 0,
(62, 'cook'): 1,
(63, 'left'): 1,
(63, 'right'): 0,
(63, 'up'): 0,
(63, 'down'): 0,
(63, 'take_whiskr'): 0,
(63, 'go_right'): 0,
(63, 'go_left'): 0,
(63, 'cook'): 0}
```

In [19]:
```
agent.easy_policy
```

Out[19]:
```
{19: 'up',
 18: 'right',
 17: 'up',
 27: 'left',
 26: 'right',
 25: 'right',
 24: 'right',
 11: 'go_right',
 10: 'right',
 9: 'right',
 8: 'up',
 3: 'up',
 2: 'right',
 1: 'right',
 0: 'right',
 23: 'down',
 22: 'take_whiskr',
 16: 'take_whiskr',
 15: 'up',
 14: 'up',
 54: 'down',
 55: 'down',
 48: 'down',
 40: 'up',
 41: 'right',
 7: 'up',
 6: 'right',
 5: 'up',
 4: 'up',
 46: 'up',
 47: 'up',
 43: 'go_right',
 42: 'right',
 13: 'up',
 12: 'up',
 39: 'up',
 38: 'right',
 37: 'up',
 36: 'up',
 51: 'up',
 50: 'right',
 49: 'up',
 35: 'up',
 34: 'right',
 33: 'right',
 32: 'right',
 21: 'up',
 20: 'up',
 45: 'up',
 44: 'up',
 59: 'left',
 58: 'right',
 57: 'right',
 56: 'right',
 29: 'down',
 28: 'down',
 53: 'up',
 52: 'up',
 30: 'right',
 61: 'down',
 60: 'down',
 31: 'left',
 62: 'cook',
 63: 'left'}
```

In [20]:
```
agent.print_policy(agent.easy_policy)
```

```
→  →  →  ←      ↓  ↓  C  ←
↓  ↑  →  ↑      ↑  ↑  ↓  ↓
↑  →  →  R      ↑  ↑  ↑  ↑
→  →  →  ↑      ↑  ↑  →  ↑
```

```
→  →  →  ←      ↓  ↓  →  ←
T  ↑  →  ↑      ↑  ↑  T  ↓
↑  →  →  R      ↑  ↑  ↑  ↑
→  →  →  ↑      ↑  ↑  →  ↑
```

Come possiamo vedere questa non è una Policy del tutto ottima, è buona ma non perfetta. Probabilmente, aumentando il numero di iterazioni massime dell'algoritmo, si riesce a convergere a una policy migliore di questa, mantenendo lo stesso gamma. Proviamo a vedere cosa succede se impostiamo solo il γ da 0.5 a 0.1

In [21]:
```
agent.discount_factor=0.1
agent.improve_policy(100)
```

```
Iterazione: 0/100
Iterazione: 1/100
Iterazione: 2/100
Iterazione: 3/100
Iterazione: 4/100
Iterazione: 5/100
Iterazione: 6/100
Iterazione: 7/100
Iterazione: 8/100
Iterazione: 9/100
Iterazione: 10/100
Iterazione: 11/100
Iterazione: 12/100
Iterazione: 13/100
Iterazione: 14/100
Iterazione: 15/100
Iterazione: 16/100
Iterazione: 17/100
Iterazione: 18/100
Iterazione: 19/100
Iterazione: 20/100
Iterazione: 21/100
Iterazione: 22/100
Iterazione: 23/100
Iterazione: 24/100
Iterazione: 25/100
Iterazione: 26/100
Iterazione: 27/100
Iterazione: 28/100
Iterazione: 29/100
Iterazione: 30/100
Iterazione: 31/100
Iterazione: 32/100
Iterazione: 33/100
Iterazione: 34/100
Iterazione: 35/100
Iterazione: 36/100
Iterazione: 37/100
Iterazione: 38/100
Iterazione: 39/100
Iterazione: 40/100
Iterazione: 41/100
Iterazione: 42/100
Iterazione: 43/100
Iterazione: 44/100
Iterazione: 45/100
Iterazione: 46/100
Iterazione: 47/100
Iterazione: 48/100
Iterazione: 49/100
Iterazione: 50/100
Iterazione: 51/100
Iterazione: 52/100
Iterazione: 53/100
Iterazione: 54/100
Iterazione: 55/100
Iterazione: 56/100
Iterazione: 57/100
Iterazione: 58/100
Iterazione: 59/100
Iterazione: 60/100
Iterazione: 61/100
Iterazione: 62/100
Iterazione: 63/100
Iterazione: 64/100
Iterazione: 65/100
Iterazione: 66/100
Iterazione: 67/100
Iterazione: 68/100
Iterazione: 69/100
Iterazione: 70/100
Iterazione: 71/100
Iterazione: 72/100
Iterazione: 73/100
Iterazione: 74/100
```

```
Iterazione: 75/100
Iterazione: 76/100
Iterazione: 77/100
Iterazione: 78/100
Iterazione: 79/100
Iterazione: 80/100
Iterazione: 81/100
Iterazione: 82/100
Iterazione: 83/100
Iterazione: 84/100
Iterazione: 85/100
Iterazione: 86/100
Iterazione: 87/100
Iterazione: 88/100
Iterazione: 89/100
Iterazione: 90/100
Iterazione: 91/100
Iterazione: 92/100
Iterazione: 93/100
Iterazione: 94/100
Iterazione: 95/100
Iterazione: 96/100
Iterazione: 97/100
Iterazione: 98/100
Iterazione: 99/100
```

Out[21]:
```
{(19, 'left'): 0,
 (19, 'right'): 0,
 (19, 'up'): 0,
 (19, 'down'): 1,
 (19, 'take_whiskr'): 0,
 (19, 'go_right'): 0,
 (19, 'go_left'): 0,
 (19, 'cook'): 0,
 (18, 'left'): 1,
 (18, 'right'): 0,
 (18, 'up'): 0,
 (18, 'down'): 0,
 (18, 'take_whiskr'): 0,
 (18, 'go_right'): 0,
 (18, 'go_left'): 0,
 (18, 'cook'): 0,
 (17, 'left'): 0,
 (17, 'right'): 1,
 (17, 'up'): 0,
 (17, 'down'): 0,
 (17, 'take_whiskr'): 0,
 (17, 'go_right'): 0,
 (17, 'go_left'): 0,
 (17, 'cook'): 0,
 (27, 'left'): 0,
 (27, 'right'): 0,
 (27, 'up'): 0,
 (27, 'down'): 1,
 (27, 'take_whiskr'): 0,
 (27, 'go_right'): 0,
 (27, 'go_left'): 0,
 (27, 'cook'): 0,
 (26, 'left'): 1,
 (26, 'right'): 0,
 (26, 'up'): 0,
 (26, 'down'): 0,
 (26, 'take_whiskr'): 0,
 (26, 'go_right'): 0,
 (26, 'go_left'): 0,
 (26, 'cook'): 0,
 (25, 'left'): 1,
 (25, 'right'): 0,
 (25, 'up'): 0,
 (25, 'down'): 0,
 (25, 'take_whiskr'): 0,
 (25, 'go_right'): 0,
 (25, 'go_left'): 0,
 (25, 'cook'): 0,
 (24, 'left'): 0,
 (24, 'right'): 1,
 (24, 'up'): 0,
 (24, 'down'): 0,
 (24, 'take_whiskr'): 0,
 (24, 'go_right'): 0,
 (24, 'go_left'): 0,
 (24, 'cook'): 0,
 (11, 'left'): 0,
 (11, 'right'): 0,
 (11, 'up'): 0,
 (11, 'down'): 0,
 (11, 'take_whiskr'): 0,
 (11, 'go_right'): 1,
 (11, 'go_left'): 0,
 (11, 'cook'): 0,
 (10, 'left'): 0,
 (10, 'right'): 1,
 (10, 'up'): 0,
 (10, 'down'): 0,
 (10, 'take_whiskr'): 0,
 (10, 'go_right'): 0,
 (10, 'go_left'): 0,
 (10, 'cook'): 0,
 (9, 'left'): 1,
 (9, 'right'): 0,
 (9, 'up'): 0,
```

```
(9, 'down'): 0,
(9, 'take_whiskr'): 0,
(9, 'go_right'): 0,
(9, 'go_left'): 0,
(9, 'cook'): 0,
(8, 'left'): 0,
(8, 'right'): 0,
(8, 'up'): 1,
(8, 'down'): 0,
(8, 'take_whiskr'): 0,
(8, 'go_right'): 0,
(8, 'go_left'): 0,
(8, 'cook'): 0,
(3, 'left'): 0,
(3, 'right'): 0,
(3, 'up'): 1,
(3, 'down'): 0,
(3, 'take_whiskr'): 0,
(3, 'go_right'): 0,
(3, 'go_left'): 0,
(3, 'cook'): 0,
(2, 'left'): 0,
(2, 'right'): 1,
(2, 'up'): 0,
(2, 'down'): 0,
(2, 'take_whiskr'): 0,
(2, 'go_right'): 0,
(2, 'go_left'): 0,
(2, 'cook'): 0,
(1, 'left'): 0,
(1, 'right'): 1,
(1, 'up'): 0,
(1, 'down'): 0,
(1, 'take_whiskr'): 0,
(1, 'go_right'): 0,
(1, 'go_left'): 0,
(1, 'cook'): 0,
(0, 'left'): 0,
(0, 'right'): 1,
(0, 'up'): 0,
(0, 'down'): 0,
(0, 'take_whiskr'): 0,
(0, 'go_right'): 0,
(0, 'go_left'): 0,
(0, 'cook'): 0,
(23, 'left'): 1,
(23, 'right'): 0,
(23, 'up'): 0,
(23, 'down'): 0,
(23, 'take_whiskr'): 0,
(23, 'go_right'): 0,
(23, 'go_left'): 0,
(23, 'cook'): 0,
(22, 'left'): 0,
(22, 'right'): 0,
(22, 'up'): 0,
(22, 'down'): 0,
(22, 'take_whiskr'): 1,
(22, 'go_right'): 0,
(22, 'go_left'): 0,
(22, 'cook'): 0,
(16, 'left'): 0,
(16, 'right'): 0,
(16, 'up'): 0,
(16, 'down'): 0,
(16, 'take_whiskr'): 1,
(16, 'go_right'): 0,
(16, 'go_left'): 0,
(16, 'cook'): 0,
(15, 'left'): 1,
(15, 'right'): 0,
(15, 'up'): 0,
(15, 'down'): 0,
(15, 'take_whiskr'): 0,
(15, 'go_right'): 0,
```

```
(15, 'go_left'): 0,
(15, 'cook'): 0,
(14, 'left'): 0,
(14, 'right'): 0,
(14, 'up'): 1,
(14, 'down'): 0,
(14, 'take_whiskr'): 0,
(14, 'go_right'): 0,
(14, 'go_left'): 0,
(14, 'cook'): 0,
(54, 'left'): 0,
(54, 'right'): 1,
(54, 'up'): 0,
(54, 'down'): 0,
(54, 'take_whiskr'): 0,
(54, 'go_right'): 0,
(54, 'go_left'): 0,
(54, 'cook'): 0,
(55, 'left'): 0,
(55, 'right'): 0,
(55, 'up'): 0,
(55, 'down'): 0,
(55, 'take_whiskr'): 0,
(55, 'go_right'): 0,
(55, 'go_left'): 1,
(55, 'cook'): 0,
(48, 'left'): 0,
(48, 'right'): 0,
(48, 'up'): 0,
(48, 'down'): 1,
(48, 'take_whiskr'): 0,
(48, 'go_right'): 0,
(48, 'go_left'): 0,
(48, 'cook'): 0,
(40, 'left'): 0,
(40, 'right'): 1,
(40, 'up'): 0,
(40, 'down'): 0,
(40, 'take_whiskr'): 0,
(40, 'go_right'): 0,
(40, 'go_left'): 0,
(40, 'cook'): 0,
(41, 'left'): 0,
(41, 'right'): 1,
(41, 'up'): 0,
(41, 'down'): 0,
(41, 'take_whiskr'): 0,
(41, 'go_right'): 0,
(41, 'go_left'): 0,
(41, 'cook'): 0,
(7, 'left'): 1,
(7, 'right'): 0,
(7, 'up'): 0,
(7, 'down'): 0,
(7, 'take_whiskr'): 0,
(7, 'go_right'): 0,
(7, 'go_left'): 0,
(7, 'cook'): 0,
(6, 'left'): 1,
(6, 'right'): 0,
(6, 'up'): 0,
(6, 'down'): 0,
(6, 'take_whiskr'): 0,
(6, 'go_right'): 0,
(6, 'go_left'): 0,
(6, 'cook'): 0,
(5, 'left'): 1,
(5, 'right'): 0,
(5, 'up'): 0,
(5, 'down'): 0,
(5, 'take_whiskr'): 0,
(5, 'go_right'): 0,
(5, 'go_left'): 0,
(5, 'cook'): 0,
(4, 'left'): 0,
```

```
(4, 'right'): 1,
(4, 'up'): 0,
(4, 'down'): 0,
(4, 'take_whiskr'): 0,
(4, 'go_right'): 0,
(4, 'go_left'): 0,
(4, 'cook'): 0,
(46, 'left'): 0,
(46, 'right'): 1,
(46, 'up'): 0,
(46, 'down'): 0,
(46, 'take_whiskr'): 0,
(46, 'go_right'): 0,
(46, 'go_left'): 0,
(46, 'cook'): 0,
(47, 'left'): 1,
(47, 'right'): 0,
(47, 'up'): 0,
(47, 'down'): 0,
(47, 'take_whiskr'): 0,
(47, 'go_right'): 0,
(47, 'go_left'): 0,
(47, 'cook'): 0,
(43, 'left'): 0,
(43, 'right'): 0,
(43, 'up'): 0,
(43, 'down'): 0,
(43, 'take_whiskr'): 0,
(43, 'go_right'): 1,
(43, 'go_left'): 0,
(43, 'cook'): 0,
(42, 'left'): 0,
(42, 'right'): 1,
(42, 'up'): 0,
(42, 'down'): 0,
(42, 'take_whiskr'): 0,
(42, 'go_right'): 0,
(42, 'go_left'): 0,
(42, 'cook'): 0,
(13, 'left'): 0,
(13, 'right'): 0,
(13, 'up'): 0,
(13, 'down'): 1,
(13, 'take_whiskr'): 0,
(13, 'go_right'): 0,
(13, 'go_left'): 0,
(13, 'cook'): 0,
(12, 'left'): 0,
(12, 'right'): 0,
(12, 'up'): 0,
(12, 'down'): 1,
(12, 'take_whiskr'): 0,
(12, 'go_right'): 0,
(12, 'go_left'): 0,
(12, 'cook'): 0,
(39, 'left'): 1,
(39, 'right'): 0,
(39, 'up'): 0,
(39, 'down'): 0,
(39, 'take_whiskr'): 0,
(39, 'go_right'): 0,
(39, 'go_left'): 0,
(39, 'cook'): 0,
(38, 'left'): 1,
(38, 'right'): 0,
(38, 'up'): 0,
(38, 'down'): 0,
(38, 'take_whiskr'): 0,
(38, 'go_right'): 0,
(38, 'go_left'): 0,
(38, 'cook'): 0,
(37, 'left'): 1,
(37, 'right'): 0,
(37, 'up'): 0,
(37, 'down'): 0,
```

```
(37, 'take_whiskr'): 0,
(37, 'go_right'): 0,
(37, 'go_left'): 0,
(37, 'cook'): 0,
(36, 'left'): 0,
(36, 'right'): 1,
(36, 'up'): 0,
(36, 'down'): 0,
(36, 'take_whiskr'): 0,
(36, 'go_right'): 0,
(36, 'go_left'): 0,
(36, 'cook'): 0,
(51, 'left'): 0,
(51, 'right'): 0,
(51, 'up'): 0,
(51, 'down'): 1,
(51, 'take_whiskr'): 0,
(51, 'go_right'): 0,
(51, 'go_left'): 0,
(51, 'cook'): 0,
(50, 'left'): 1,
(50, 'right'): 0,
(50, 'up'): 0,
(50, 'down'): 0,
(50, 'take_whiskr'): 0,
(50, 'go_right'): 0,
(50, 'go_left'): 0,
(50, 'cook'): 0,
(49, 'left'): 0,
(49, 'right'): 1,
(49, 'up'): 0,
(49, 'down'): 0,
(49, 'take_whiskr'): 0,
(49, 'go_right'): 0,
(49, 'go_left'): 0,
(49, 'cook'): 0,
(35, 'left'): 0,
(35, 'right'): 0,
(35, 'up'): 1,
(35, 'down'): 0,
(35, 'take_whiskr'): 0,
(35, 'go_right'): 0,
(35, 'go_left'): 0,
(35, 'cook'): 0,
(34, 'left'): 0,
(34, 'right'): 1,
(34, 'up'): 0,
(34, 'down'): 0,
(34, 'take_whiskr'): 0,
(34, 'go_right'): 0,
(34, 'go_left'): 0,
(34, 'cook'): 0,
(33, 'left'): 0,
(33, 'right'): 1,
(33, 'up'): 0,
(33, 'down'): 0,
(33, 'take_whiskr'): 0,
(33, 'go_right'): 0,
(33, 'go_left'): 0,
(33, 'cook'): 0,
(32, 'left'): 0,
(32, 'right'): 1,
(32, 'up'): 0,
(32, 'down'): 0,
(32, 'take_whiskr'): 0,
(32, 'go_right'): 0,
(32, 'go_left'): 0,
(32, 'cook'): 0,
(21, 'left'): 1,
(21, 'right'): 0,
(21, 'up'): 0,
(21, 'down'): 0,
(21, 'take_whiskr'): 0,
(21, 'go_right'): 0,
(21, 'go_left'): 0,
```

```
(21, 'cook'): 0,
(20, 'left'): 0,
(20, 'right'): 1,
(20, 'up'): 0,
(20, 'down'): 0,
(20, 'take_whiskr'): 0,
(20, 'go_right'): 0,
(20, 'go_left'): 0,
(20, 'cook'): 0,
(45, 'left'): 0,
(45, 'right'): 0,
(45, 'up'): 0,
(45, 'down'): 1,
(45, 'take_whiskr'): 0,
(45, 'go_right'): 0,
(45, 'go_left'): 0,
(45, 'cook'): 0,
(44, 'left'): 0,
(44, 'right'): 0,
(44, 'up'): 0,
(44, 'down'): 1,
(44, 'take_whiskr'): 0,
(44, 'go_right'): 0,
(44, 'go_left'): 0,
(44, 'cook'): 0,
(59, 'left'): 0,
(59, 'right'): 0,
(59, 'up'): 0,
(59, 'down'): 1,
(59, 'take_whiskr'): 0,
(59, 'go_right'): 0,
(59, 'go_left'): 0,
(59, 'cook'): 0,
(58, 'left'): 1,
(58, 'right'): 0,
(58, 'up'): 0,
(58, 'down'): 0,
(58, 'take_whiskr'): 0,
(58, 'go_right'): 0,
(58, 'go_left'): 0,
(58, 'cook'): 0,
(57, 'left'): 1,
(57, 'right'): 0,
(57, 'up'): 0,
(57, 'down'): 0,
(57, 'take_whiskr'): 0,
(57, 'go_right'): 0,
(57, 'go_left'): 0,
(57, 'cook'): 0,
(56, 'left'): 0,
(56, 'right'): 1,
(56, 'up'): 0,
(56, 'down'): 0,
(56, 'take_whiskr'): 0,
(56, 'go_right'): 0,
(56, 'go_left'): 0,
(56, 'cook'): 0,
(29, 'left'): 1,
(29, 'right'): 0,
(29, 'up'): 0,
(29, 'down'): 0,
(29, 'take_whiskr'): 0,
(29, 'go_right'): 0,
(29, 'go_left'): 0,
(29, 'cook'): 0,
(28, 'left'): 0,
(28, 'right'): 1,
(28, 'up'): 0,
(28, 'down'): 0,
(28, 'take_whiskr'): 0,
(28, 'go_right'): 0,
(28, 'go_left'): 0,
(28, 'cook'): 0,
(53, 'left'): 1,
(53, 'right'): 0,
```

```
(53, 'up'): 0,
(53, 'down'): 0,
(53, 'take_whiskr'): 0,
(53, 'go_right'): 0,
(53, 'go_left'): 0,
(53, 'cook'): 0,
(52, 'left'): 0,
(52, 'right'): 1,
(52, 'up'): 0,
(52, 'down'): 0,
(52, 'take_whiskr'): 0,
(52, 'go_right'): 0,
(52, 'go_left'): 0,
(52, 'cook'): 0,
(30, 'left'): 1,
(30, 'right'): 0,
(30, 'up'): 0,
(30, 'down'): 0,
(30, 'take_whiskr'): 0,
(30, 'go_right'): 0,
(30, 'go_left'): 0,
(30, 'cook'): 0,
(61, 'left'): 0,
(61, 'right'): 1,
(61, 'up'): 0,
(61, 'down'): 0,
(61, 'take_whiskr'): 0,
(61, 'go_right'): 0,
(61, 'go_left'): 0,
(61, 'cook'): 0,
(60, 'left'): 0,
(60, 'right'): 1,
(60, 'up'): 0,
(60, 'down'): 0,
(60, 'take_whiskr'): 0,
(60, 'go_right'): 0,
(60, 'go_left'): 0,
(60, 'cook'): 0,
(31, 'left'): 1,
(31, 'right'): 0,
(31, 'up'): 0,
(31, 'down'): 0,
(31, 'take_whiskr'): 0,
(31, 'go_right'): 0,
(31, 'go_left'): 0,
(31, 'cook'): 0,
(62, 'left'): 0,
(62, 'right'): 0,
(62, 'up'): 0,
(62, 'down'): 0,
(62, 'take_whiskr'): 0,
(62, 'go_right'): 0,
(62, 'go_left'): 0,
(62, 'cook'): 1,
(63, 'left'): 1,
(63, 'right'): 0,
(63, 'up'): 0,
(63, 'down'): 0,
(63, 'take_whiskr'): 0,
(63, 'go_right'): 0,
(63, 'go_left'): 0,
(63, 'cook'): 0}
```

In [22]:
```
agent.easy_policy
```

Out[22]:
```
{19: 'down',
 18: 'left',
 17: 'right',
 27: 'down',
 26: 'left',
 25: 'left',
 24: 'right',
 11: 'go_right',
 10: 'right',
 9: 'left',
 8: 'up',
 3: 'up',
 2: 'right',
 1: 'right',
 0: 'right',
 23: 'left',
 22: 'take_whiskr',
 16: 'take_whiskr',
 15: 'left',
 14: 'up',
 54: 'right',
 55: 'go_left',
 48: 'down',
 40: 'right',
 41: 'right',
 7: 'left',
 6: 'left',
 5: 'left',
 4: 'right',
 46: 'right',
 47: 'left',
 43: 'go_right',
 42: 'right',
 13: 'down',
 12: 'down',
 39: 'left',
 38: 'left',
 37: 'left',
 36: 'right',
 51: 'down',
 50: 'left',
 49: 'right',
 35: 'up',
 34: 'right',
 33: 'right',
 32: 'right',
 21: 'left',
 20: 'right',
 45: 'down',
 44: 'down',
 59: 'down',
 58: 'left',
 57: 'left',
 56: 'right',
 29: 'left',
 28: 'right',
 53: 'left',
 52: 'right',
 30: 'left',
 61: 'right',
 60: 'right',
 31: 'left',
 62: 'cook',
 63: 'left'}
```

In [23]:
```
agent.print_policy(agent.easy_policy)
```

```
→  ←  ←  ↓      →  →  C  ←
↓  →  ←  ↓      →  ←  →  L
→  →  →  R      ↓  ↓  →  ←
→  →  →  ↑      →  ←  ←  ←
```

```
→  ←  ←  ↓      →  ←  ←  ←
T  →  ←  ↓      →  ←  T  ←
↑  ←  →  R      ↓  ↓  ↑  ←
→  →  →  ↑      →  ←  ←  ←
```

Come possiamo vedere questa non è una Policy del tutto ottima, è buona ma non perfetta. Rispetto alla precedente policy questa risulta essere meno ottima. Proviamo a vedere cosa succede se impostiamo solo il γ da 0.1 a 0.9. Quindi con l'agente che è interessato più ai rewards futuri.

In [24]:
```
agent.discount_factor=0.9
agent.improve_policy(100)
```

```
Iterazione: 0/100
Iterazione: 1/100
Iterazione: 2/100
Iterazione: 3/100
Iterazione: 4/100
Iterazione: 5/100
Iterazione: 6/100
Iterazione: 7/100
Iterazione: 8/100
Iterazione: 9/100
Iterazione: 10/100
Iterazione: 11/100
Iterazione: 12/100
Iterazione: 13/100
Iterazione: 14/100
Iterazione: 15/100
Iterazione: 16/100
Iterazione: 17/100
Iterazione: 18/100
Iterazione: 19/100
Iterazione: 20/100
Iterazione: 21/100
Iterazione: 22/100
Iterazione: 23/100
Iterazione: 24/100
Iterazione: 25/100
Iterazione: 26/100
Iterazione: 27/100
Iterazione: 28/100
Iterazione: 29/100
Iterazione: 30/100
Iterazione: 31/100
Iterazione: 32/100
Iterazione: 33/100
Iterazione: 34/100
Iterazione: 35/100
Iterazione: 36/100
Iterazione: 37/100
Iterazione: 38/100
Iterazione: 39/100
Iterazione: 40/100
Iterazione: 41/100
Iterazione: 42/100
Iterazione: 43/100
Iterazione: 44/100
Iterazione: 45/100
Iterazione: 46/100
Iterazione: 47/100
Iterazione: 48/100
Iterazione: 49/100
Iterazione: 50/100
Iterazione: 51/100
Iterazione: 52/100
Iterazione: 53/100
Iterazione: 54/100
Iterazione: 55/100
Iterazione: 56/100
Iterazione: 57/100
Iterazione: 58/100
Iterazione: 59/100
Iterazione: 60/100
Iterazione: 61/100
Iterazione: 62/100
Iterazione: 63/100
Iterazione: 64/100
Iterazione: 65/100
Iterazione: 66/100
Iterazione: 67/100
Iterazione: 68/100
Iterazione: 69/100
Iterazione: 70/100
Iterazione: 71/100
Iterazione: 72/100
Iterazione: 73/100
Iterazione: 74/100
```

```
Iterazione: 75/100
Iterazione: 76/100
Iterazione: 77/100
Iterazione: 78/100
Iterazione: 79/100
Iterazione: 80/100
Iterazione: 81/100
Iterazione: 82/100
Iterazione: 83/100
Iterazione: 84/100
Iterazione: 85/100
Iterazione: 86/100
Iterazione: 87/100
Iterazione: 88/100
Iterazione: 89/100
Iterazione: 90/100
Iterazione: 91/100
Iterazione: 92/100
Iterazione: 93/100
Iterazione: 94/100
Iterazione: 95/100
Iterazione: 96/100
Iterazione: 97/100
Iterazione: 98/100
Iterazione: 99/100
```

Out[24]:
```
{(19, 'left'): 0,
 (19, 'right'): 0,
 (19, 'up'): 0,
 (19, 'down'): 1,
 (19, 'take_whiskr'): 0,
 (19, 'go_right'): 0,
 (19, 'go_left'): 0,
 (19, 'cook'): 0,
 (18, 'left'): 0,
 (18, 'right'): 1,
 (18, 'up'): 0,
 (18, 'down'): 0,
 (18, 'take_whiskr'): 0,
 (18, 'go_right'): 0,
 (18, 'go_left'): 0,
 (18, 'cook'): 0,
 (17, 'left'): 0,
 (17, 'right'): 0,
 (17, 'up'): 1,
 (17, 'down'): 0,
 (17, 'take_whiskr'): 0,
 (17, 'go_right'): 0,
 (17, 'go_left'): 0,
 (17, 'cook'): 0,
 (27, 'left'): 0,
 (27, 'right'): 0,
 (27, 'up'): 0,
 (27, 'down'): 1,
 (27, 'take_whiskr'): 0,
 (27, 'go_right'): 0,
 (27, 'go_left'): 0,
 (27, 'cook'): 0,
 (26, 'left'): 0,
 (26, 'right'): 1,
 (26, 'up'): 0,
 (26, 'down'): 0,
 (26, 'take_whiskr'): 0,
 (26, 'go_right'): 0,
 (26, 'go_left'): 0,
 (26, 'cook'): 0,
 (25, 'left'): 0,
 (25, 'right'): 0,
 (25, 'up'): 0,
 (25, 'down'): 1,
 (25, 'take_whiskr'): 0,
 (25, 'go_right'): 0,
 (25, 'go_left'): 0,
 (25, 'cook'): 0,
 (24, 'left'): 0,
 (24, 'right'): 1,
 (24, 'up'): 0,
 (24, 'down'): 0,
 (24, 'take_whiskr'): 0,
 (24, 'go_right'): 0,
 (24, 'go_left'): 0,
 (24, 'cook'): 0,
 (11, 'left'): 0,
 (11, 'right'): 0,
 (11, 'up'): 0,
 (11, 'down'): 0,
 (11, 'take_whiskr'): 0,
 (11, 'go_right'): 1,
 (11, 'go_left'): 0,
 (11, 'cook'): 0,
 (10, 'left'): 0,
 (10, 'right'): 1,
 (10, 'up'): 0,
 (10, 'down'): 0,
 (10, 'take_whiskr'): 0,
 (10, 'go_right'): 0,
 (10, 'go_left'): 0,
 (10, 'cook'): 0,
 (9, 'left'): 1,
 (9, 'right'): 0,
 (9, 'up'): 0,
```

```
(9, 'down'): 0,
(9, 'take_whiskr'): 0,
(9, 'go_right'): 0,
(9, 'go_left'): 0,
(9, 'cook'): 0,
(8, 'left'): 0,
(8, 'right'): 0,
(8, 'up'): 1,
(8, 'down'): 0,
(8, 'take_whiskr'): 0,
(8, 'go_right'): 0,
(8, 'go_left'): 0,
(8, 'cook'): 0,
(3, 'left'): 0,
(3, 'right'): 0,
(3, 'up'): 1,
(3, 'down'): 0,
(3, 'take_whiskr'): 0,
(3, 'go_right'): 0,
(3, 'go_left'): 0,
(3, 'cook'): 0,
(2, 'left'): 0,
(2, 'right'): 1,
(2, 'up'): 0,
(2, 'down'): 0,
(2, 'take_whiskr'): 0,
(2, 'go_right'): 0,
(2, 'go_left'): 0,
(2, 'cook'): 0,
(1, 'left'): 0,
(1, 'right'): 1,
(1, 'up'): 0,
(1, 'down'): 0,
(1, 'take_whiskr'): 0,
(1, 'go_right'): 0,
(1, 'go_left'): 0,
(1, 'cook'): 0,
(0, 'left'): 0,
(0, 'right'): 1,
(0, 'up'): 0,
(0, 'down'): 0,
(0, 'take_whiskr'): 0,
(0, 'go_right'): 0,
(0, 'go_left'): 0,
(0, 'cook'): 0,
(23, 'left'): 1,
(23, 'right'): 0,
(23, 'up'): 0,
(23, 'down'): 0,
(23, 'take_whiskr'): 0,
(23, 'go_right'): 0,
(23, 'go_left'): 0,
(23, 'cook'): 0,
(22, 'left'): 0,
(22, 'right'): 0,
(22, 'up'): 0,
(22, 'down'): 0,
(22, 'take_whiskr'): 1,
(22, 'go_right'): 0,
(22, 'go_left'): 0,
(22, 'cook'): 0,
(16, 'left'): 0,
(16, 'right'): 0,
(16, 'up'): 0,
(16, 'down'): 0,
(16, 'take_whiskr'): 1,
(16, 'go_right'): 0,
(16, 'go_left'): 0,
(16, 'cook'): 0,
(15, 'left'): 1,
(15, 'right'): 0,
(15, 'up'): 0,
(15, 'down'): 0,
(15, 'take_whiskr'): 0,
(15, 'go_right'): 0,
```

```
(15, 'go_left'): 0,
(15, 'cook'): 0,
(14, 'left'): 0,
(14, 'right'): 0,
(14, 'up'): 1,
(14, 'down'): 0,
(14, 'take_whiskr'): 0,
(14, 'go_right'): 0,
(14, 'go_left'): 0,
(14, 'cook'): 0,
(54, 'left'): 0,
(54, 'right'): 1,
(54, 'up'): 0,
(54, 'down'): 0,
(54, 'take_whiskr'): 0,
(54, 'go_right'): 0,
(54, 'go_left'): 0,
(54, 'cook'): 0,
(55, 'left'): 0,
(55, 'right'): 0,
(55, 'up'): 0,
(55, 'down'): 0,
(55, 'take_whiskr'): 0,
(55, 'go_right'): 0,
(55, 'go_left'): 1,
(55, 'cook'): 0,
(48, 'left'): 0,
(48, 'right'): 0,
(48, 'up'): 0,
(48, 'down'): 1,
(48, 'take_whiskr'): 0,
(48, 'go_right'): 0,
(48, 'go_left'): 0,
(48, 'cook'): 0,
(40, 'left'): 0,
(40, 'right'): 1,
(40, 'up'): 0,
(40, 'down'): 0,
(40, 'take_whiskr'): 0,
(40, 'go_right'): 0,
(40, 'go_left'): 0,
(40, 'cook'): 0,
(41, 'left'): 0,
(41, 'right'): 1,
(41, 'up'): 0,
(41, 'down'): 0,
(41, 'take_whiskr'): 0,
(41, 'go_right'): 0,
(41, 'go_left'): 0,
(41, 'cook'): 0,
(7, 'left'): 0,
(7, 'right'): 0,
(7, 'up'): 1,
(7, 'down'): 0,
(7, 'take_whiskr'): 0,
(7, 'go_right'): 0,
(7, 'go_left'): 0,
(7, 'cook'): 0,
(6, 'left'): 0,
(6, 'right'): 1,
(6, 'up'): 0,
(6, 'down'): 0,
(6, 'take_whiskr'): 0,
(6, 'go_right'): 0,
(6, 'go_left'): 0,
(6, 'cook'): 0,
(5, 'left'): 0,
(5, 'right'): 1,
(5, 'up'): 0,
(5, 'down'): 0,
(5, 'take_whiskr'): 0,
(5, 'go_right'): 0,
(5, 'go_left'): 0,
(5, 'cook'): 0,
(4, 'left'): 0,
```

```
(4, 'right'): 0,
(4, 'up'): 1,
(4, 'down'): 0,
(4, 'take_whiskr'): 0,
(4, 'go_right'): 0,
(4, 'go_left'): 0,
(4, 'cook'): 0,
(46, 'left'): 0,
(46, 'right'): 0,
(46, 'up'): 1,
(46, 'down'): 0,
(46, 'take_whiskr'): 0,
(46, 'go_right'): 0,
(46, 'go_left'): 0,
(46, 'cook'): 0,
(47, 'left'): 0,
(47, 'right'): 0,
(47, 'up'): 1,
(47, 'down'): 0,
(47, 'take_whiskr'): 0,
(47, 'go_right'): 0,
(47, 'go_left'): 0,
(47, 'cook'): 0,
(43, 'left'): 0,
(43, 'right'): 0,
(43, 'up'): 0,
(43, 'down'): 0,
(43, 'take_whiskr'): 0,
(43, 'go_right'): 1,
(43, 'go_left'): 0,
(43, 'cook'): 0,
(42, 'left'): 0,
(42, 'right'): 1,
(42, 'up'): 0,
(42, 'down'): 0,
(42, 'take_whiskr'): 0,
(42, 'go_right'): 0,
(42, 'go_left'): 0,
(42, 'cook'): 0,
(13, 'left'): 0,
(13, 'right'): 0,
(13, 'up'): 1,
(13, 'down'): 0,
(13, 'take_whiskr'): 0,
(13, 'go_right'): 0,
(13, 'go_left'): 0,
(13, 'cook'): 0,
(12, 'left'): 0,
(12, 'right'): 0,
(12, 'up'): 1,
(12, 'down'): 0,
(12, 'take_whiskr'): 0,
(12, 'go_right'): 0,
(12, 'go_left'): 0,
(12, 'cook'): 0,
(39, 'left'): 0,
(39, 'right'): 0,
(39, 'up'): 1,
(39, 'down'): 0,
(39, 'take_whiskr'): 0,
(39, 'go_right'): 0,
(39, 'go_left'): 0,
(39, 'cook'): 0,
(38, 'left'): 0,
(38, 'right'): 1,
(38, 'up'): 0,
(38, 'down'): 0,
(38, 'take_whiskr'): 0,
(38, 'go_right'): 0,
(38, 'go_left'): 0,
(38, 'cook'): 0,
(37, 'left'): 0,
(37, 'right'): 1,
(37, 'up'): 0,
(37, 'down'): 0,
```

```
(37, 'take_whiskr'): 0,
(37, 'go_right'): 0,
(37, 'go_left'): 0,
(37, 'cook'): 0,
(36, 'left'): 0,
(36, 'right'): 0,
(36, 'up'): 1,
(36, 'down'): 0,
(36, 'take_whiskr'): 0,
(36, 'go_right'): 0,
(36, 'go_left'): 0,
(36, 'cook'): 0,
(51, 'left'): 0,
(51, 'right'): 0,
(51, 'up'): 0,
(51, 'down'): 1,
(51, 'take_whiskr'): 0,
(51, 'go_right'): 0,
(51, 'go_left'): 0,
(51, 'cook'): 0,
(50, 'left'): 0,
(50, 'right'): 1,
(50, 'up'): 0,
(50, 'down'): 0,
(50, 'take_whiskr'): 0,
(50, 'go_right'): 0,
(50, 'go_left'): 0,
(50, 'cook'): 0,
(49, 'left'): 0,
(49, 'right'): 0,
(49, 'up'): 1,
(49, 'down'): 0,
(49, 'take_whiskr'): 0,
(49, 'go_right'): 0,
(49, 'go_left'): 0,
(49, 'cook'): 0,
(35, 'left'): 0,
(35, 'right'): 0,
(35, 'up'): 1,
(35, 'down'): 0,
(35, 'take_whiskr'): 0,
(35, 'go_right'): 0,
(35, 'go_left'): 0,
(35, 'cook'): 0,
(34, 'left'): 0,
(34, 'right'): 1,
(34, 'up'): 0,
(34, 'down'): 0,
(34, 'take_whiskr'): 0,
(34, 'go_right'): 0,
(34, 'go_left'): 0,
(34, 'cook'): 0,
(33, 'left'): 0,
(33, 'right'): 1,
(33, 'up'): 0,
(33, 'down'): 0,
(33, 'take_whiskr'): 0,
(33, 'go_right'): 0,
(33, 'go_left'): 0,
(33, 'cook'): 0,
(32, 'left'): 0,
(32, 'right'): 1,
(32, 'up'): 0,
(32, 'down'): 0,
(32, 'take_whiskr'): 0,
(32, 'go_right'): 0,
(32, 'go_left'): 0,
(32, 'cook'): 0,
(21, 'left'): 1,
(21, 'right'): 0,
(21, 'up'): 0,
(21, 'down'): 0,
(21, 'take_whiskr'): 0,
(21, 'go_right'): 0,
(21, 'go_left'): 0,
```

```
(21, 'cook'): 0,
(20, 'left'): 0,
(20, 'right'): 1,
(20, 'up'): 0,
(20, 'down'): 0,
(20, 'take_whiskr'): 0,
(20, 'go_right'): 0,
(20, 'go_left'): 0,
(20, 'cook'): 0,
(45, 'left'): 0,
(45, 'right'): 0,
(45, 'up'): 1,
(45, 'down'): 0,
(45, 'take_whiskr'): 0,
(45, 'go_right'): 0,
(45, 'go_left'): 0,
(45, 'cook'): 0,
(44, 'left'): 0,
(44, 'right'): 0,
(44, 'up'): 1,
(44, 'down'): 0,
(44, 'take_whiskr'): 0,
(44, 'go_right'): 0,
(44, 'go_left'): 0,
(44, 'cook'): 0,
(59, 'left'): 0,
(59, 'right'): 0,
(59, 'up'): 0,
(59, 'down'): 1,
(59, 'take_whiskr'): 0,
(59, 'go_right'): 0,
(59, 'go_left'): 0,
(59, 'cook'): 0,
(58, 'left'): 0,
(58, 'right'): 1,
(58, 'up'): 0,
(58, 'down'): 0,
(58, 'take_whiskr'): 0,
(58, 'go_right'): 0,
(58, 'go_left'): 0,
(58, 'cook'): 0,
(57, 'left'): 0,
(57, 'right'): 0,
(57, 'up'): 0,
(57, 'down'): 1,
(57, 'take_whiskr'): 0,
(57, 'go_right'): 0,
(57, 'go_left'): 0,
(57, 'cook'): 0,
(56, 'left'): 0,
(56, 'right'): 1,
(56, 'up'): 0,
(56, 'down'): 0,
(56, 'take_whiskr'): 0,
(56, 'go_right'): 0,
(56, 'go_left'): 0,
(56, 'cook'): 0,
(29, 'left'): 1,
(29, 'right'): 0,
(29, 'up'): 0,
(29, 'down'): 0,
(29, 'take_whiskr'): 0,
(29, 'go_right'): 0,
(29, 'go_left'): 0,
(29, 'cook'): 0,
(28, 'left'): 0,
(28, 'right'): 1,
(28, 'up'): 0,
(28, 'down'): 0,
(28, 'take_whiskr'): 0,
(28, 'go_right'): 0,
(28, 'go_left'): 0,
(28, 'cook'): 0,
(53, 'left'): 0,
(53, 'right'): 0,
```

```
(53, 'up'): 1,
(53, 'down'): 0,
(53, 'take_whiskr'): 0,
(53, 'go_right'): 0,
(53, 'go_left'): 0,
(53, 'cook'): 0,
(52, 'left'): 0,
(52, 'right'): 0,
(52, 'up'): 1,
(52, 'down'): 0,
(52, 'take_whiskr'): 0,
(52, 'go_right'): 0,
(52, 'go_left'): 0,
(52, 'cook'): 0,
(30, 'left'): 1,
(30, 'right'): 0,
(30, 'up'): 0,
(30, 'down'): 0,
(30, 'take_whiskr'): 0,
(30, 'go_right'): 0,
(30, 'go_left'): 0,
(30, 'cook'): 0,
(61, 'left'): 0,
(61, 'right'): 1,
(61, 'up'): 0,
(61, 'down'): 0,
(61, 'take_whiskr'): 0,
(61, 'go_right'): 0,
(61, 'go_left'): 0,
(61, 'cook'): 0,
(60, 'left'): 0,
(60, 'right'): 1,
(60, 'up'): 0,
(60, 'down'): 0,
(60, 'take_whiskr'): 0,
(60, 'go_right'): 0,
(60, 'go_left'): 0,
(60, 'cook'): 0,
(31, 'left'): 1,
(31, 'right'): 0,
(31, 'up'): 0,
(31, 'down'): 0,
(31, 'take_whiskr'): 0,
(31, 'go_right'): 0,
(31, 'go_left'): 0,
(31, 'cook'): 0,
(62, 'left'): 0,
(62, 'right'): 0,
(62, 'up'): 0,
(62, 'down'): 0,
(62, 'take_whiskr'): 0,
(62, 'go_right'): 0,
(62, 'go_left'): 0,
(62, 'cook'): 1,
(63, 'left'): 1,
(63, 'right'): 0,
(63, 'up'): 0,
(63, 'down'): 0,
(63, 'take_whiskr'): 0,
(63, 'go_right'): 0,
(63, 'go_left'): 0,
(63, 'cook'): 0}
```

In [25]:
```
agent.easy_policy
```

Out[25]:
```
{19: 'down',
 18: 'right',
 17: 'up',
 27: 'down',
 26: 'right',
 25: 'down',
 24: 'right',
 11: 'go_right',
 10: 'right',
 9: 'left',
 8: 'up',
 3: 'up',
 2: 'right',
 1: 'right',
 0: 'right',
 23: 'left',
 22: 'take_whiskr',
 16: 'take_whiskr',
 15: 'left',
 14: 'up',
 54: 'right',
 55: 'go_left',
 48: 'down',
 40: 'right',
 41: 'right',
 7: 'up',
 6: 'right',
 5: 'right',
 4: 'up',
 46: 'up',
 47: 'up',
 43: 'go_right',
 42: 'right',
 13: 'up',
 12: 'up',
 39: 'up',
 38: 'right',
 37: 'right',
 36: 'up',
 51: 'down',
 50: 'right',
 49: 'up',
 35: 'up',
 34: 'right',
 33: 'right',
 32: 'right',
 21: 'left',
 20: 'right',
 45: 'up',
 44: 'up',
 59: 'down',
 58: 'right',
 57: 'down',
 56: 'right',
 29: 'left',
 28: 'right',
 53: 'up',
 52: 'up',
 30: 'left',
 61: 'right',
 60: 'right',
 31: 'left',
 62: 'cook',
 63: 'left'}
```

In [26]:
```
agent.print_policy(agent.easy_policy)
```

```
→  ↓  →  ↓      →  →  C  ←
↓  ↑  →  ↓      ↑  ↑  →  L
→  →  →  R      ↑  ↑  ↑  ↑
→  →  →  ↑      ↑  →  →  ↑
```

```
→  ↓  →  ↓      →  ←  ←  ←
T  ↑  →  ↓      →  ←  T  ←
↑  ←  →  R      ↑  ↑  ↑  ←
→  →  →  ↑      ↑  →  →  ↑
```

Come possiamo vedere questa non è una Policy del tutto ottima, è buona ma non perfetta. Rispetto alla precedente policy questa risulta essere più ottima. Se la paragoniamo a quella con il gamma 0.5 risulta essere leggermente più performante. Probabilmente, aumentando il numero di iterazioni massime dell'algoritmo, si riesce a convergere a una policy migliore di questa.

Proviamo a implementare l'algoritmo **Monte Carlo Online Control / On Policy Improvement** disponibile a pagina 44 delle slide del modulo **Model-free**.

```
def __argmax(self, q: dict, s: int):
    chiavi_filtrate = [chiave for chiave in q if chiave[0] == s]
    chiave_max = max(chiavi_filtrate, key=lambda k: q[k])
    return chiave_max[1]

def __epsilon_greedy_policy(self, q: dict, epsilon: float):
    pi = {}
    for s in self.env_representation.S:
        pi[s.number] = np.random.choice(
            [self.__argmax(q, s.number), random.choice(self.actions).name],
            p=[1 - epsilon, epsilon])
    return pi

def __generate_episode_from_pi(self, pi: dict):
    episode = []
    for s, a in pi.items():
        episode.append({'s': s, 'a': self.env_representation.action_dict[a]})
    return episode

def monte_carlo_online_control_on_policy_improvement(self, iterations: int, gamma=None):
    if gamma is None:
        gamma = self.discount_factor
    q = {(state.number, action.name): 0 for state in self.states for action in self.actions}
    n = {(state.number, action.name): 0 for state in self.states for action in self.actions}
    k = 1
    epsilon = 1 / k
    pi = self.__epsilon_greedy_policy(q, epsilon)

    while k <= iterations:
        state_visited = []
        episode = self.__generate_episode_from_pi(pi)
        q_k = deepcopy(q)
        g_k = 0
        for t in reversed(range(0, len(episode))):
            s_t = episode[t]['s']
            a_t = episode[t]['a']
            r_t = self.env_representation.R[
                s_t, a_t.number, a_t.function(self.env_representation.state_dict[s_t]).number]
            g_k += (gamma ** t) * r_t
            if (s_t, a_t.name) not in state_visited:
                n[s_t, a_t.name] += 1
                q[s_t, a_t.name] = q_k[s_t, a_t.name] + (1 / n[s_t, a_t.name]) * (g_k -
q_k[s_t, a_t.name])
                state_visited.append((s_t, a_t.name))
        k += 1
        print(f"Iterazione: {k}/{iterations}")
        epsilon = 1 / k
        pi = self.__epsilon_greedy_policy(q, epsilon)

    # Policy Improvement
    new_policy = self.policy.copy()
```

```
        for state_no in self.states:
            for action in self.actions:
                new_policy[(state_no.number, action.name)] = 1 if action.name ==
    pi[state_no.number] else 0

        self.change_policy(new_policy)
        return self.policy
```

In [37]:
```
agent.monte_carlo_online_control_on_policy_improvement(100, 0.1)
```

```
Iterazione: 2/100
Iterazione: 3/100
Iterazione: 4/100
Iterazione: 5/100
Iterazione: 6/100
Iterazione: 7/100
Iterazione: 8/100
Iterazione: 9/100
Iterazione: 10/100
Iterazione: 11/100
Iterazione: 12/100
Iterazione: 13/100
Iterazione: 14/100
Iterazione: 15/100
Iterazione: 16/100
Iterazione: 17/100
Iterazione: 18/100
Iterazione: 19/100
Iterazione: 20/100
Iterazione: 21/100
Iterazione: 22/100
Iterazione: 23/100
Iterazione: 24/100
Iterazione: 25/100
Iterazione: 26/100
Iterazione: 27/100
Iterazione: 28/100
Iterazione: 29/100
Iterazione: 30/100
Iterazione: 31/100
Iterazione: 32/100
Iterazione: 33/100
Iterazione: 34/100
Iterazione: 35/100
Iterazione: 36/100
Iterazione: 37/100
Iterazione: 38/100
Iterazione: 39/100
Iterazione: 40/100
Iterazione: 41/100
Iterazione: 42/100
Iterazione: 43/100
Iterazione: 44/100
Iterazione: 45/100
Iterazione: 46/100
Iterazione: 47/100
Iterazione: 48/100
Iterazione: 49/100
Iterazione: 50/100
Iterazione: 51/100
Iterazione: 52/100
Iterazione: 53/100
Iterazione: 54/100
Iterazione: 55/100
Iterazione: 56/100
Iterazione: 57/100
Iterazione: 58/100
Iterazione: 59/100
Iterazione: 60/100
Iterazione: 61/100
Iterazione: 62/100
Iterazione: 63/100
Iterazione: 64/100
Iterazione: 65/100
Iterazione: 66/100
Iterazione: 67/100
Iterazione: 68/100
Iterazione: 69/100
Iterazione: 70/100
Iterazione: 71/100
Iterazione: 72/100
Iterazione: 73/100
Iterazione: 74/100
Iterazione: 75/100
Iterazione: 76/100
```

```
Iterazione: 77/100
Iterazione: 78/100
Iterazione: 79/100
Iterazione: 80/100
Iterazione: 81/100
Iterazione: 82/100
Iterazione: 83/100
Iterazione: 84/100
Iterazione: 85/100
Iterazione: 86/100
Iterazione: 87/100
Iterazione: 88/100
Iterazione: 89/100
Iterazione: 90/100
Iterazione: 91/100
Iterazione: 92/100
Iterazione: 93/100
Iterazione: 94/100
Iterazione: 95/100
Iterazione: 96/100
Iterazione: 97/100
Iterazione: 98/100
Iterazione: 99/100
Iterazione: 100/100
Iterazione: 101/100
```

Out[37]:
```
{(19, 'left'): 1,
 (19, 'right'): 0,
 (19, 'up'): 0,
 (19, 'down'): 0,
 (19, 'take_whiskr'): 0,
 (19, 'go_right'): 0,
 (19, 'go_left'): 0,
 (19, 'cook'): 0,
 (18, 'left'): 0,
 (18, 'right'): 0,
 (18, 'up'): 1,
 (18, 'down'): 0,
 (18, 'take_whiskr'): 0,
 (18, 'go_right'): 0,
 (18, 'go_left'): 0,
 (18, 'cook'): 0,
 (17, 'left'): 0,
 (17, 'right'): 1,
 (17, 'up'): 0,
 (17, 'down'): 0,
 (17, 'take_whiskr'): 0,
 (17, 'go_right'): 0,
 (17, 'go_left'): 0,
 (17, 'cook'): 0,
 (27, 'left'): 1,
 (27, 'right'): 0,
 (27, 'up'): 0,
 (27, 'down'): 0,
 (27, 'take_whiskr'): 0,
 (27, 'go_right'): 0,
 (27, 'go_left'): 0,
 (27, 'cook'): 0,
 (26, 'left'): 1,
 (26, 'right'): 0,
 (26, 'up'): 0,
 (26, 'down'): 0,
 (26, 'take_whiskr'): 0,
 (26, 'go_right'): 0,
 (26, 'go_left'): 0,
 (26, 'cook'): 0,
 (25, 'left'): 1,
 (25, 'right'): 0,
 (25, 'up'): 0,
 (25, 'down'): 0,
 (25, 'take_whiskr'): 0,
 (25, 'go_right'): 0,
 (25, 'go_left'): 0,
 (25, 'cook'): 0,
 (24, 'left'): 0,
 (24, 'right'): 1,
 (24, 'up'): 0,
 (24, 'down'): 0,
 (24, 'take_whiskr'): 0,
 (24, 'go_right'): 0,
 (24, 'go_left'): 0,
 (24, 'cook'): 0,
 (11, 'left'): 0,
 (11, 'right'): 0,
 (11, 'up'): 0,
 (11, 'down'): 0,
 (11, 'take_whiskr'): 0,
 (11, 'go_right'): 1,
 (11, 'go_left'): 0,
 (11, 'cook'): 0,
 (10, 'left'): 1,
 (10, 'right'): 0,
 (10, 'up'): 0,
 (10, 'down'): 0,
 (10, 'take_whiskr'): 0,
 (10, 'go_right'): 0,
 (10, 'go_left'): 0,
 (10, 'cook'): 0,
 (9, 'left'): 1,
 (9, 'right'): 0,
 (9, 'up'): 0,
```

```
(9, 'down'): 0,
(9, 'take_whiskr'): 0,
(9, 'go_right'): 0,
(9, 'go_left'): 0,
(9, 'cook'): 0,
(8, 'left'): 0,
(8, 'right'): 1,
(8, 'up'): 0,
(8, 'down'): 0,
(8, 'take_whiskr'): 0,
(8, 'go_right'): 0,
(8, 'go_left'): 0,
(8, 'cook'): 0,
(3, 'left'): 1,
(3, 'right'): 0,
(3, 'up'): 0,
(3, 'down'): 0,
(3, 'take_whiskr'): 0,
(3, 'go_right'): 0,
(3, 'go_left'): 0,
(3, 'cook'): 0,
(2, 'left'): 0,
(2, 'right'): 1,
(2, 'up'): 0,
(2, 'down'): 0,
(2, 'take_whiskr'): 0,
(2, 'go_right'): 0,
(2, 'go_left'): 0,
(2, 'cook'): 0,
(1, 'left'): 0,
(1, 'right'): 1,
(1, 'up'): 0,
(1, 'down'): 0,
(1, 'take_whiskr'): 0,
(1, 'go_right'): 0,
(1, 'go_left'): 0,
(1, 'cook'): 0,
(0, 'left'): 0,
(0, 'right'): 1,
(0, 'up'): 0,
(0, 'down'): 0,
(0, 'take_whiskr'): 0,
(0, 'go_right'): 0,
(0, 'go_left'): 0,
(0, 'cook'): 0,
(23, 'left'): 0,
(23, 'right'): 0,
(23, 'up'): 0,
(23, 'down'): 1,
(23, 'take_whiskr'): 0,
(23, 'go_right'): 0,
(23, 'go_left'): 0,
(23, 'cook'): 0,
(22, 'left'): 0,
(22, 'right'): 0,
(22, 'up'): 0,
(22, 'down'): 0,
(22, 'take_whiskr'): 1,
(22, 'go_right'): 0,
(22, 'go_left'): 0,
(22, 'cook'): 0,
(16, 'left'): 0,
(16, 'right'): 0,
(16, 'up'): 0,
(16, 'down'): 0,
(16, 'take_whiskr'): 1,
(16, 'go_right'): 0,
(16, 'go_left'): 0,
(16, 'cook'): 0,
(15, 'left'): 0,
(15, 'right'): 0,
(15, 'up'): 0,
(15, 'down'): 1,
(15, 'take_whiskr'): 0,
(15, 'go_right'): 0,
```

```
(15, 'go_left'): 0,
(15, 'cook'): 0,
(14, 'left'): 0,
(14, 'right'): 0,
(14, 'up'): 1,
(14, 'down'): 0,
(14, 'take_whiskr'): 0,
(14, 'go_right'): 0,
(14, 'go_left'): 0,
(14, 'cook'): 0,
(54, 'left'): 0,
(54, 'right'): 0,
(54, 'up'): 0,
(54, 'down'): 1,
(54, 'take_whiskr'): 0,
(54, 'go_right'): 0,
(54, 'go_left'): 0,
(54, 'cook'): 0,
(55, 'left'): 0,
(55, 'right'): 0,
(55, 'up'): 0,
(55, 'down'): 0,
(55, 'take_whiskr'): 0,
(55, 'go_right'): 0,
(55, 'go_left'): 1,
(55, 'cook'): 0,
(48, 'left'): 0,
(48, 'right'): 0,
(48, 'up'): 0,
(48, 'down'): 1,
(48, 'take_whiskr'): 0,
(48, 'go_right'): 0,
(48, 'go_left'): 0,
(48, 'cook'): 0,
(40, 'left'): 0,
(40, 'right'): 0,
(40, 'up'): 1,
(40, 'down'): 0,
(40, 'take_whiskr'): 0,
(40, 'go_right'): 0,
(40, 'go_left'): 0,
(40, 'cook'): 0,
(41, 'left'): 0,
(41, 'right'): 1,
(41, 'up'): 0,
(41, 'down'): 0,
(41, 'take_whiskr'): 0,
(41, 'go_right'): 0,
(41, 'go_left'): 0,
(41, 'cook'): 0,
(7, 'left'): 1,
(7, 'right'): 0,
(7, 'up'): 0,
(7, 'down'): 0,
(7, 'take_whiskr'): 0,
(7, 'go_right'): 0,
(7, 'go_left'): 0,
(7, 'cook'): 0,
(6, 'left'): 1,
(6, 'right'): 0,
(6, 'up'): 0,
(6, 'down'): 0,
(6, 'take_whiskr'): 0,
(6, 'go_right'): 0,
(6, 'go_left'): 0,
(6, 'cook'): 0,
(5, 'left'): 0,
(5, 'right'): 0,
(5, 'up'): 1,
(5, 'down'): 0,
(5, 'take_whiskr'): 0,
(5, 'go_right'): 0,
(5, 'go_left'): 0,
(5, 'cook'): 0,
(4, 'left'): 0,
```

```
(4, 'right'): 1,
(4, 'up'): 0,
(4, 'down'): 0,
(4, 'take_whiskr'): 0,
(4, 'go_right'): 0,
(4, 'go_left'): 0,
(4, 'cook'): 0,
(46, 'left'): 0,
(46, 'right'): 1,
(46, 'up'): 0,
(46, 'down'): 0,
(46, 'take_whiskr'): 0,
(46, 'go_right'): 0,
(46, 'go_left'): 0,
(46, 'cook'): 0,
(47, 'left'): 1,
(47, 'right'): 0,
(47, 'up'): 0,
(47, 'down'): 0,
(47, 'take_whiskr'): 0,
(47, 'go_right'): 0,
(47, 'go_left'): 0,
(47, 'cook'): 0,
(43, 'left'): 0,
(43, 'right'): 0,
(43, 'up'): 0,
(43, 'down'): 0,
(43, 'take_whiskr'): 0,
(43, 'go_right'): 1,
(43, 'go_left'): 0,
(43, 'cook'): 0,
(42, 'left'): 1,
(42, 'right'): 0,
(42, 'up'): 0,
(42, 'down'): 0,
(42, 'take_whiskr'): 0,
(42, 'go_right'): 0,
(42, 'go_left'): 0,
(42, 'cook'): 0,
(13, 'left'): 0,
(13, 'right'): 0,
(13, 'up'): 0,
(13, 'down'): 1,
(13, 'take_whiskr'): 0,
(13, 'go_right'): 0,
(13, 'go_left'): 0,
(13, 'cook'): 0,
(12, 'left'): 0,
(12, 'right'): 1,
(12, 'up'): 0,
(12, 'down'): 0,
(12, 'take_whiskr'): 0,
(12, 'go_right'): 0,
(12, 'go_left'): 0,
(12, 'cook'): 0,
(39, 'left'): 1,
(39, 'right'): 0,
(39, 'up'): 0,
(39, 'down'): 0,
(39, 'take_whiskr'): 0,
(39, 'go_right'): 0,
(39, 'go_left'): 0,
(39, 'cook'): 0,
(38, 'left'): 0,
(38, 'right'): 1,
(38, 'up'): 0,
(38, 'down'): 0,
(38, 'take_whiskr'): 0,
(38, 'go_right'): 0,
(38, 'go_left'): 0,
(38, 'cook'): 0,
(37, 'left'): 1,
(37, 'right'): 0,
(37, 'up'): 0,
(37, 'down'): 0,
```

```
(37, 'take_whiskr'): 0,
(37, 'go_right'): 0,
(37, 'go_left'): 0,
(37, 'cook'): 0,
(36, 'left'): 0,
(36, 'right'): 0,
(36, 'up'): 1,
(36, 'down'): 0,
(36, 'take_whiskr'): 0,
(36, 'go_right'): 0,
(36, 'go_left'): 0,
(36, 'cook'): 0,
(51, 'left'): 0,
(51, 'right'): 0,
(51, 'up'): 1,
(51, 'down'): 0,
(51, 'take_whiskr'): 0,
(51, 'go_right'): 0,
(51, 'go_left'): 0,
(51, 'cook'): 0,
(50, 'left'): 1,
(50, 'right'): 0,
(50, 'up'): 0,
(50, 'down'): 0,
(50, 'take_whiskr'): 0,
(50, 'go_right'): 0,
(50, 'go_left'): 0,
(50, 'cook'): 0,
(49, 'left'): 0,
(49, 'right'): 0,
(49, 'up'): 1,
(49, 'down'): 0,
(49, 'take_whiskr'): 0,
(49, 'go_right'): 0,
(49, 'go_left'): 0,
(49, 'cook'): 0,
(35, 'left'): 0,
(35, 'right'): 0,
(35, 'up'): 1,
(35, 'down'): 0,
(35, 'take_whiskr'): 0,
(35, 'go_right'): 0,
(35, 'go_left'): 0,
(35, 'cook'): 0,
(34, 'left'): 0,
(34, 'right'): 1,
(34, 'up'): 0,
(34, 'down'): 0,
(34, 'take_whiskr'): 0,
(34, 'go_right'): 0,
(34, 'go_left'): 0,
(34, 'cook'): 0,
(33, 'left'): 1,
(33, 'right'): 0,
(33, 'up'): 0,
(33, 'down'): 0,
(33, 'take_whiskr'): 0,
(33, 'go_right'): 0,
(33, 'go_left'): 0,
(33, 'cook'): 0,
(32, 'left'): 0,
(32, 'right'): 1,
(32, 'up'): 0,
(32, 'down'): 0,
(32, 'take_whiskr'): 0,
(32, 'go_right'): 0,
(32, 'go_left'): 0,
(32, 'cook'): 0,
(21, 'left'): 0,
(21, 'right'): 0,
(21, 'up'): 0,
(21, 'down'): 1,
(21, 'take_whiskr'): 0,
(21, 'go_right'): 0,
(21, 'go_left'): 0,
```

```
(21, 'cook'): 0,
(20, 'left'): 0,
(20, 'right'): 1,
(20, 'up'): 0,
(20, 'down'): 0,
(20, 'take_whiskr'): 0,
(20, 'go_right'): 0,
(20, 'go_left'): 0,
(20, 'cook'): 0,
(45, 'left'): 1,
(45, 'right'): 0,
(45, 'up'): 0,
(45, 'down'): 0,
(45, 'take_whiskr'): 0,
(45, 'go_right'): 0,
(45, 'go_left'): 0,
(45, 'cook'): 0,
(44, 'left'): 0,
(44, 'right'): 0,
(44, 'up'): 0,
(44, 'down'): 1,
(44, 'take_whiskr'): 0,
(44, 'go_right'): 0,
(44, 'go_left'): 0,
(44, 'cook'): 0,
(59, 'left'): 0,
(59, 'right'): 0,
(59, 'up'): 0,
(59, 'down'): 1,
(59, 'take_whiskr'): 0,
(59, 'go_right'): 0,
(59, 'go_left'): 0,
(59, 'cook'): 0,
(58, 'left'): 1,
(58, 'right'): 0,
(58, 'up'): 0,
(58, 'down'): 0,
(58, 'take_whiskr'): 0,
(58, 'go_right'): 0,
(58, 'go_left'): 0,
(58, 'cook'): 0,
(57, 'left'): 0,
(57, 'right'): 1,
(57, 'up'): 0,
(57, 'down'): 0,
(57, 'take_whiskr'): 0,
(57, 'go_right'): 0,
(57, 'go_left'): 0,
(57, 'cook'): 0,
(56, 'left'): 0,
(56, 'right'): 1,
(56, 'up'): 0,
(56, 'down'): 0,
(56, 'take_whiskr'): 0,
(56, 'go_right'): 0,
(56, 'go_left'): 0,
(56, 'cook'): 0,
(29, 'left'): 0,
(29, 'right'): 1,
(29, 'up'): 0,
(29, 'down'): 0,
(29, 'take_whiskr'): 0,
(29, 'go_right'): 0,
(29, 'go_left'): 0,
(29, 'cook'): 0,
(28, 'left'): 0,
(28, 'right'): 1,
(28, 'up'): 0,
(28, 'down'): 0,
(28, 'take_whiskr'): 0,
(28, 'go_right'): 0,
(28, 'go_left'): 0,
(28, 'cook'): 0,
(53, 'left'): 0,
(53, 'right'): 0,
```

```
 (53, 'up'): 0,
 (53, 'down'): 1,
 (53, 'take_whiskr'): 0,
 (53, 'go_right'): 0,
 (53, 'go_left'): 0,
 (53, 'cook'): 0,
 (52, 'left'): 0,
 (52, 'right'): 0,
 (52, 'up'): 1,
 (52, 'down'): 0,
 (52, 'take_whiskr'): 0,
 (52, 'go_right'): 0,
 (52, 'go_left'): 0,
 (52, 'cook'): 0,
 (30, 'left'): 0,
 (30, 'right'): 1,
 (30, 'up'): 0,
 (30, 'down'): 0,
 (30, 'take_whiskr'): 0,
 (30, 'go_right'): 0,
 (30, 'go_left'): 0,
 (30, 'cook'): 0,
 (61, 'left'): 0,
 (61, 'right'): 0,
 (61, 'up'): 0,
 (61, 'down'): 1,
 (61, 'take_whiskr'): 0,
 (61, 'go_right'): 0,
 (61, 'go_left'): 0,
 (61, 'cook'): 0,
 (60, 'left'): 0,
 (60, 'right'): 1,
 (60, 'up'): 0,
 (60, 'down'): 0,
 (60, 'take_whiskr'): 0,
 (60, 'go_right'): 0,
 (60, 'go_left'): 0,
 (60, 'cook'): 0,
 (31, 'left'): 1,
 (31, 'right'): 0,
 (31, 'up'): 0,
 (31, 'down'): 0,
 (31, 'take_whiskr'): 0,
 (31, 'go_right'): 0,
 (31, 'go_left'): 0,
 (31, 'cook'): 0,
 (62, 'left'): 0,
 (62, 'right'): 0,
 (62, 'up'): 0,
 (62, 'down'): 0,
 (62, 'take_whiskr'): 0,
 (62, 'go_right'): 0,
 (62, 'go_left'): 0,
 (62, 'cook'): 1,
 (63, 'left'): 1,
 (63, 'right'): 0,
 (63, 'up'): 0,
 (63, 'down'): 0,
 (63, 'take_whiskr'): 0,
 (63, 'go_right'): 0,
 (63, 'go_left'): 0,
 (63, 'cook'): 0}
```

In [38]:
```
agent.print_policy(agent.easy_policy)
```

```
→  →  ←  ↓      →  ↓  C  ←
↓  ↑  ←  ↑      ↑  ↓  ↓  L
↑  →  ←  R      ↓  ←  →  ←
→  ←  →  ↑      ↑  ←  →  ←


→  ←  ←  ←      →  →  →  ←
T  →  ↑  ←      →  ↓  T  ↓
→  ←  ←  R      →  ↓  ↑  ↓
→  →  →  ←      →  ↑  ←  ←
```

Come possiamo vedere l'utilizzo dell'algoritmo **Monte Carlo Online Control / On Policy Improvement** riesce a produce una policy ottima simile all'algoritmo **improve_policy** con lo stesso γ.