In [1]:
```
import matplotlib.pyplot as plt
from math import ceil
import pandas as pd
from aux import *
from z3 import *
```

Nota. O ficheiro aux.py é um ficheiro python com as definições das classes Projeto e Schedule e funções auxiliares que permitem fazer o display dos horários. Decidiu-se colocar num ficheiro auxiliar para não desviar a atenção do código 3 de todas as fórmulas lógicas utilizadas. Na diretoria "Trabalho1/Projetos/" encontram-se os documentos csv que foram utilizados como input para a resolução deste problema.

# Problema 1: Horário de uma Startup

É necessário criar um horário semanal para uma startup, sequindo as seguintes condições:

1. Há $P$ projetos, enumerados de $(0, \ldots, P-1)$.
2. Cada projeto $p$ tem $C_p$ colaboradores enumerados de $(0, \ldots, C_p-1)$ dos quais um ($C_p = 0$) é líder.
3. Para cada projeto $p$ há um mínimo de $N_p$ reuniões semanais.
4. As reuniões têm de conter sempre o líder e pelo menos 50% dos colaboradores (incluindo o líder).
5. As reuniões ocorrem em uma das $S$ salas enumeradas de $(0, \ldots, S-1)$.
6. Os tempos são representados por slots (horas, dias).
7. Os dias $d$ são enumerados de $(0, \ldots, D-1)$ e as horas $h$ de $(0, H-1)$.

## Variáveis do problema

Para resolver este problema, precisamos de variáveis que nos permitam representar a disponibilidade dos colaboradores para determinar se datas valeis para a ocorrência de reuniões. Como tal, definiram-se os seguintes grupos de variáveis binárias que podem tomar os valores inteiros $(0, 1)$:

- $x_{p,c,d,h}$: variável que representa a disponibilidade do colaborador $c$ do projeto $p$, onde $c = 0$ é o líder, no dia $d$ e na hora $h$ desse dia.
- $r_{r,d,h,s}$: variável que representa a disponibilidade da sala $s$ no dia $d$ e hora $h$ para o projeto $p$.
- $t_{p,d,h,s}$: variável que representa a existência de uma reunião para o projeto $p$ no dia $d$, hora $h$ e sala $s$.

Onde $p \in [0, P-1]$, $c \in [0, C_p-1]$, $d \in [0, D-1]$, $h \in [0, H-1]$ e $s \in [0, S-1]$.

## Condição do valor das variáveis

Todas as variáveis acima descritas são inteiras unitárias ou nulas (exemplo: cada pessoa ou está disponível numa data ou não está):

$$0 \leq x_{p,c,d,h} \leq 1, \forall_{p,c,d,h}$$
$$0 \leq r_{r,d,h,s} \leq 1, \forall_{r,d,h,p}$$
$$0 \leq t_{p,d,h,s} \leq 1, \forall_{p,d,h,s}$$

## Condições de input

De forma a criar um horário coerente e compatível com as disponibilidades de todos os intervenientes, é necessário estabelecer restrições às variáveis definidas anteriormente.

As condições de input são relativas à disponibilidade de cada um dos colaboradores, e do número mínimo de reuniões que se pretende realizar nessa semana para cada projeto.

1. A disponibilidade dos colaboradores de cada projeto $x_{p,c,d,h}^{(c)}$ é dada como input do problema:

$$x_{p,c,d,h} = x_{p,c,d,h}^{(c)}, \forall_{p,c,d,h}$$

2. Deve haver um número mínimo de reuniões semanais por projeto $N_p$ dada como input do problema:

$$\sum_{d=0}^{D-1} \sum_{h=0}^{H-1} \sum_{s=0}^{S-1} t_{p,d,h,s} \geq N_p, \forall_p$$

## Condições inerentes ao problema

As condições inerentes aos problema do horário são relativas à verificação da coerência do mesmo (exemplo: uma reunião não pode ocorrer em duas salas ao mesmo tempo). Estas condições são:

1. Cada sala, por dia e hora, está disponível para reunião de apenas um projeto:

$$\sum_{p=0}^{P-1} r_{r,d,h,s} = 1, \forall_{r,d,h}$$

1. Cada projeto só pode ter no máximo uma reunião por dia e hora:

$$\sum_{s=0}^{S-1} r_{p,d,h,s} \leq 1, \forall_{p,d,h}$$

1. Para haver reunião de um projeto numa certa sala, dia e hora, o líder e pelo menos 50% dos colaboradores devem estar disponíveis e a sala também:

$$r_{p,d,h,s} = 1 \rightarrow \left( r_{r,d,h,p} = 1 \wedge x_{p,0,d,h} = 1 \wedge \sum_{c=1}^{C_p-1} x_{p,c,d,h} \geq \left\lceil \frac{C_p - 1}{2} \right\rceil \right), \forall_{p,d,h,s}$$

## Condições opcionais

Estas condições opcionais visam tornar o horário mais conveniente para todos os intervenientes. As condições exeqüinhas são as seguintes:

1. Se possível, marcar no máximo uma reunião por dia:

$$\sum_{h=0}^{H-1} \sum_{s=0}^{S-1} t_{p,d,h,s} \leq 1, \forall_{p,d}$$

1. No caso de não se verificar a condição anterior, para reuniões consecutivas do mesmo projeto, estas devem ser sempre na mesma sala:

$$t_{p,d,h,s} * \sum_{s'=0, s' \neq s}^{S-1} t_{p,d,h+1,s'} \leq 1, \forall_{p,s \in [0,S-1], d \in [0,D-1]}$$

Caso nenhuma das condições opcionais seja satisfazível, estas são removidas, mantendo as anteriores.

In [2]:
```
def get_meetings(self):
    # Variáveis auxiliares
    days = ["mon", "tue", "wed", "thu", "fri"]
    rooms = self.rooms
    vogais = {'á': 'a', 'é': 'e', 'í': 'i', 'ó': 'o', 'ú': 'u', 'à': 'a', 'ó': 'o'}

    # Remover acentos nos nomes (problema do z3 com pontuação em string s)
    x_in = {}
    for p in self.x_in:
        x_in[p] = {}
        for nome in self.x_in[p]:
            novo = nome
            for v in vogais:
                novo = novo.replace(v, vogais[v])
            x_in[p][novo] = self.x_in[p][nome]

    solver = Solver()
    # Criar as variáveis de interesse x, y e r descritos anteriormente
    x = {p: {c: {d: {h: Int(f'x{p},{c},{d},{h}') for h in range(5, 17)} for d in days} for c in x_in[p]} for p in x_in}
    y = {s: {d: {h: {p: Int(f'y{s},{d},{h},{p}') for p in x_in} for h in range(5, 17)} for d in days} for s in rooms}
    r = {p: {d: {h: Int(f'r{p},{s},{d},{h}') for s in rooms} for h in range(5, 17)} for d in days} for p in x_in}

    # Condição do valor das variáveis e condição de input 1
    for p in x:
        for c in x[p]:
            for d in x[p][c]:
                for h in x[p][c][d]:
                    solver.add(x[p][c][d][h] >= 0, x[p][c][d][h] <= 1)
                    solver.add(x[p][c][d][h] >= x_in[p][c][d][h])

    for s in y:
        for d in y[s]:
            for h in y[s][d]:
                for p in y[s][d][h]:
                    solver.add(y[s][d][h][p] <= 1, y[s][d][h][p] >= 0)
                    solver.add(r[p][d][h][s] <= 1, r[p][d][h][s] >= 0)

    # Condição inerente 1
    for s in y:
        for d in y[s]:
            for h in y[s][d]:
                solver.add(Sum([y[s][d][h][p] for p in y[s][d][h]]) == 1
)

    # Condição inerente 2
    for p in r:
        for d in r[p]:
            for h in r[p][d]:
                solver.add(Sum([r[p][d][h][s] for s in r[p][d][h]]) <= 1
)

    # Condição inerente 3
    for p in r:
        C = len(x_in[p])
        leader_name = next(iter(x[p]))
        for d in r[p]:
            for h in r[p][d]:
                for s in r[p][d][h]:
                    solver.add(Implies(r[p][d][h][s] == 1, x[p][leader_n
ame][d][h] >= 1))
                    solver.add(Implies(r[p][d][h][s] == 1, y[s][d][h][p]
>= 1))
                    solver.add(Implies(r[p][d][h][s] == 1, Sum([x[p][c][
d][h] for c in x[p]]) >= ceil((C-1)/2)))

    # Condição de input 2
    for p in r:
        N = self.num_meetings[p]
        soma = 0
        for d in r[p]:
            for h in r[p][d]:
                for s in r[p][d][h]:
                    soma += r[p][d][h][s]
        solver.add(soma >= N)

    # Condição opcional de uma reunião máxima por dia
    solver.push()
    for p in r:
        for d in r[p]:
            soma = 0
            for h in r[p][d]:
                for s in r[p][d][h]:
                    soma += r[p][d][h][s]
            solver.add(soma <= 1)

    if solver.check() == sat: # Verificar satisfatibilidade
        print("\nThis schedule is solvable.\n")
        m = solver.model()
    else: # Condições não verificadas
        solver.pop() # Remover condição de reuniões em dias separados

        # Condição opcional de reuniões seguidas terem a mesma sala
        solver.push()
        for p in r:
            for d in r[p]:
                for h in r[p][d]:
                    if h < 16:
                        for s in r[p][d][h]:
                            solver.add(r[p][d][h][s] + Sum([r[p][d][h+1]
[s_] for s_ in r[p][d][h+1] if s_ > s]) <= 1)

        if solver.check() == sat:
            print("\nThis schedule is solvable with more than one meeti
ng a day.\n")
            m = solver.model()
        else: # Condição opcional de reuniões consecutivas
            solver.pop() # Remover condição de reuniões consecutivas
            if solver.check() == sat: # Horário satisfazível sem as cond
ições opcionais
                print("\nThis schedule is solvable, but with more than o
ne meeting a day and consecutive meetings might have different rooms\n")
                m = solver.model()
            else: # Horário insatisfazível
                print("\nThis schedule is not solvable!\n")
                m = None

    if m != None: # Se o horário for satisfazível
        # Converter os dias das reuniões numa lista de tuplos
        result = [tuple(str(elem).split(':')) for elem in m if ":" in s
tr(elem) and m[elem] == 1]
        result = sorted([(r1,split('.')[1], r3, r4, r2) for (r1, r2, r3,
r4) in result])

        # Converter essa lista de tuplos num dataframe do pandas usando
 funções auxiliares definidas no ficheiro aux.py
        result = meeting_to_df(result)

        # Mostrar o resultado do horário e exportá-lo como ficheiro csv
        display(result)
        result.to_csv("schedule.csv")
    else: # Se o horário não for satisfazível
        result = None

    return result

Schedule.get_meetings = get_meetings
```

## Teste Exemplo

Este algoritmo lê os dados de input como ficheiros csv. Os ficheiros csv de cada projeto devem estar contidos numa diretoria, sendo o nome de cada projeto atribuído de acordo com o nome de cada ficheiro.

Podemos visualizar a estrutura esperada para os ficheiros csv, fazendo o display de um deles, relativo ao projeto "Lógica":

In [3]:
```
exemplo = read_schedule("Projetos/Logica.csv")
display(exemplo)
```

| | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| **8h-9h** | Pedro* Tomás Gil | Pedro* Tomás Zé | Pedro* Tomás Marco Zé | Pedro* Zé Miguel Gil | Pedro* Tomás Zé Marco |
| **9h-10h** | Tomás Zé João | Marco Miguel Zé | Zé Tomás Gil Marco | Tomás Zé João | Marco Miguel Zé |
| **10h-11h** | João Tomás Marco | Pedro* Zé Marco | João Pedro* Zé | João Tomás Marco | Pedro* Zé Marco |
| **11h-12h** | Marco Gil Miguel Pedro* | Tomás Gil Zé | NaN | Marco Gil Miguel Pedro* | Tomás Gil Zé |
| **12h-13h** | NaN | NaN | NaN | NaN | NaN |
| **13h-14h** | NaN | NaN | NaN | NaN | NaN |
| **14h-15h** | Tomás Pedro* Zé | João Pedro* Miguel Gil | Marco Tomás Pedro* | Tomás Pedro* Zé | João Pedro* Miguel Gil |
| **15h-16h** | Zé Marco | Tomás Miguel Marco João | Gil Zé Marco | Zé Marco | Tomás Miguel Marco João |
| **16h-17h** | Miguel Marco | NaN | NaN | Miguel Marco | NaN |

Note-se que em todos os ficheiros de input, os nomes de cada colaborador devem estar inseridos nas horas em que se encontram disponíveis, separados por espaços e o nome do líder é assinalado com um *.

Lendo todos os ficheiros da diretoria correspondentes a cada projeto, é então gerado (se possível) um horário que cumpra as restrições descritas anteriormente:

No caso abaixo é possível marcar apenas uma reunião por dia para cada projeto:

In [4]:
```
# Definir as salas disponíveis
rooms = ["S1", "S2", "S3", "S4", "S5"]

# Transformar os dados dos ficheiros csv num objeto da classe Schedule,
 definida no ficheiro aux.py
path = "Projetos"
startup = get_schedule(path, rooms)

# Chamar o método da classe Schedule que determina e apresenta os horári
os
r = startup.get_meetings()
```

```
This schedule is solvable.
```

| | mon | tue | wed | thu | fri |
|---|---|---|---|---|---|
| **8h-9h** | Logica - S2 || Logica - S3 || Logica - S1 || Logica - S1 || Logica - S3 || TMNT - S1 || The_Beatles - S2 || The_Office - S1 | Logica - S3 || TMNT - S1 || The_Beatles - S2 || The_Office - S1 | Logica - S2 || TMNT - S1 || The_Beatles - S2 | Logica - S3 || TMNT - S4 || The_Beatles - S2 | |
| **9h-10h** | The_Beatles - S1 | TMNT - S1 | The_Beatles - S1 | | |
| **10h-11h** | | | The_Office - S1 | | |
| **11h-12h** | | | | | |
| **12h-13h** | | | | | |
| **13h-14h** | | | | The_Office - S1 | |
| **14h-15h** | | | | | |
| **15h-16h** | | | | | |
| **16h-17h** | | | | | |

Neste outro caso tal não é possível, pois há mais reuniões que dias úteis na semana. No entanto, o horário continua satisfazível sem esta condição opcional.

Note-se que é possível, no entanto, alocar reuniões consecutivas na mesma sala.

In [5]:
```
# Definir as salas disponíveis
rooms = ["S1", "S2", "S3", "S4", "S5"]

# Transformar os dados dos ficheiros csv num objeto da classe Schedule,
 definida no ficheiro aux.py
path = "Projetos"
startup = get_schedule(path, rooms)

# Chamar o método da classe Schedule que determina e apresenta os horári
```

```
os
r = startup.get_meetings()
```

```
This schedule is solvable with more than one meeting a day.
```

| | mon | tue | wed | thu | fri |
|---|---|---|---|---|---|
| **8h-9h** | [ Logica - S2 ][ TMNT - S3 ][ The_Office - S1 ] | [ Logica - S3 ][ The_Beatles - S2 ][ The_Offic... | | | |
| **9h-10h** | [ The_Beatles - S1 ] | [ TMNT - S2 ][ The_Office - S3 ] | | | |
| **10h-11h** | [ The_Beatles - S1 ] | [ Logica - S3 ] | | | |
| **11h-12h** | [ Logica - S2 ][ TMNT - S3 ][ The_Office - S1 ] | | | | |
| **12h-13h** | [ TMNT - S1 ] | | | | |
| **13h-14h** | | | | | |
| **14h-15h** | [ Logica - S1 ][ TMNT - S3 ][ The_Beatles - S2 ] | [ Logica - S1 ] | | | |
| **15h-16h** | [ TMNT - S3 ][ The_Beatles - S2 ][ The_Office ... | | | | |
| **16h-17h** | [ The_Beatles - S2 ][ The_Office - S1 ] | | | | |