# Contents

### 0.0.0.1 Algorithm 1

Text

**Algorithm 1:** Extraction of data from S1 Activities dataset

```
Result: Intermediate dataframe with 'activity', 'date', 'startTime', 'endTime' as attributes
Input  : S1 Activities
Output: S1 Activities Intermediate
begin
    array1-21 = [ ]
    while $i < length(dataframe)$ do
        array1.append(dataframe[i][0])
        array2.append([x.strip() for x in array1[i].split(',')])
        array3.append(array2[i][0])
        array4.append(array2[i][1])
        array5.append(array2[i][2])
        array6.append(array2[i][3])
        i = i + 1
    end
    while $i < length(dataframe)$ do
        array7.append(dataframe[i][1])
        array8.append([x.strip() for x in array7[i].split(',')])
        array9.append(dataframe[i][2])
        array10.append([x.strip() for x in array9[i].split(',')])
        array11.append(dataframe[i][3])
        array12.append([x.strip() for x in array11[i].split(',')])
        array13.append(dataframe[i][4])
        array14.append([x.strip() for x in array13[i].split(',')])
        i = i + 1
    end
    while $i < length(dataframe)$ do
        for x in range(len(array8[i])) : array15.append(array4[i])
        i = i + 1
    end
    for sublist in array8: for item in sublist: array16.append(item)
    for sublist in array10: for item in sublist: array17.append(item)
    for sublist in array12: for item in sublist: array18.append(item)
    for sublist in array13: for item in sublist: array19.append(item)
    dfIntermediate = pandas.DataFrame(list(zip(array16, array17, array15, array18, array19)))
    start = (dfIntermediate.date + " " + dfIntermediate.startTime)
    end = (dfIntermediate.date + " " + dfIntermediate.endTime)
    while $i < length(start)$ do
        array20.append(datetime.strptime(start[i], mm/dd/yyyy HH:MM:SS))
        array21.append(datetime.strptime(end[i], mm/dd/yyyy HH:MM:SS))
        i = i + 1
    end
    dfFinal = pandas.DataFrame(list(zip(array16, array17, array20, array21)))
    return dfIntermediate
    return dfFinal
end
```

%algorithm usage

**Algorithm 2:** Step planning

```
Input: entities; actors; player; customRules;
Output: new world state
let entities be all entities, excluding actors and the player;
let actors be all actors, including the player;
foreach actor in actors do
    actor.rules.Invoke();

/* It's possible that one of the rules gave the player a constraint, so we'll check      */
if $player.constraint == null$ then
    GenerateConstraint(player);

foreach rule in customRules do
    rule.Invoke();
```

%procedure usage

**Procedure** GenerateConstraint(Player)

```
Input: player
Output: new constraint for the player
if $rules.count \geq 0$ then
    foreach item in rules do
        rules.invoke(player);
else
    select random $e$ from entities;
    player.constraint = $e$;
```

**Algorithm 3:** Algorithm with procedure

```
Algorithm algo()
    xxx
    xxx
    proc()
    return
Procedure proc()
    xxx
    return
```