ATAI 2025 Project 2

Author: Frederik Wagner
Matricular Nr.: 11906799
E-Mail: frederik.wagner@student.tugraz.at

---

1 Task 1 - Rebuild elevator.lp

In the elevator.lp we need to define the externals. we do this on the beginning of the file. We define the external facts with each parameters and make sure the initial fact definitions are commented out.

```
#external at(E,F,T) : elevator(E), floor(F), time(T).
#external todo_call(F,D,T) : floor(F), dir(D), time(T).
#external todo_deliver(E,F,T) : elevator(E), floor(F), time(T).
#external priority(E,D,T) : elevator(E), dir(D), time(T).
```

2 Task 2 - Multi-shot control.py

Next step is to convert the control.py into a multi-shot solver. The first steps are done out of the loop since we only want them to be done in the beginning and not in every step. We start by creating the clingo control object:

```
ctl = clingo.Control(arguments = ["--opt-strategy", "usc", "--warn", "none"])
for arg in sys.argv[1:]:
    ctl.load(arg)
ctl.load("next.lp")
ctl.ground([("instance", []), ("events", [])])
```

Next we read in the initial states and events:

```
state = []
event = []

for atom in ctl.symbolic_atoms.by_signature("init", 2):
    args = atom.symbol.arguments
    args.append(Number(0))
    state.append(Function("at", args))
for atom in ctl.symbolic_atoms.by_signature("call", 2):
    args = atom.symbol.arguments
    args.append(Number(0))
    state.append(Function("todo_call", args))
    args.append(Number(0))
    event.append(Function("call", args))
for atom in ctl.symbolic_atoms.by_signature("deliver", 2):
    args = atom.symbol.arguments
    args.append(Number(0))
    state.append(Function("todo_deliver", args))
for atom in ctl.symbolic_atoms.by_signature("priority", 2):
    args = atom.symbol.arguments
    args.append(Number(0))
    state.append(Function("priority", args))
for atom in ctl.symbolic_atoms.by_signature("call", 3):
    args = atom.symbol.arguments
    args.append(Number(0))
    event.append(Function("call", args))
for atom in ctl.symbolic_atoms.by_signature("call_deliver", 4):
    args = atom.symbol.arguments
    args.append(Number(0))
    event.append(Function("call_deliver", args))

print("INITIAL STATE:", [str(a) for a in state])
print("INITIAL EVENTS:", [str(a) for a in event])
```

Now that everything is setup we can start with the solving.

```
active_externals = []
while todo:
    step = step+1
    todo = False
    answer = 1

    for ext in active_externals:
        ctl.assign_external(ext, False)

    active_externals = []

    for atom in state:
        sym = clingo.Function(atom.name, atom.arguments)
        ctl.assign_external(sym, True)
        active_externals.append(sym)

    events_name = "event_" + str(step)
    if event: ctl.add(events_name, [], ".".join([str(a) for a in event]) + ".")
    ctl.ground([("next", [Number(step)]), (events_name, [])])

    ctl.solve(on_model = on_model)
```

Each loop we get one step further. During each step we load all new states. First we set the old externals we saved in the active_state_externals to inactive. Next we set the externals we need for this step to active. then we load the events for this step. After grounding the events we can solve the model for this step.

3 Task 3 - Solving performance