Unified Planning: A Python Library
Making Planning Technology Accessible

programmational modelling, transformation and solving of planning tasks in one framework

multi-paradigm: classical, temporal, numeric, multi-agent, hierarchical ...

multi-engine: e.g. Fast Downward, ENHSP, TAMER, skdecide, Tarski transformations, ...

from unified_planning_atoms import *

# Suppose we have a graph of locations and we want to plan how to move from INIT to DEST...
# This is application-specific data
location_map = {'INIT', 'DEST' : generate_netsys_topology(), 'start_node', 'goal_node' locations = (init(l) : Object(l[0]), location) for l in location_map.keys())

# Planning problem can be created entirely from code!
# Python code and libraries can be used to build problems and plans are data structures
Location = UserType('Location')
problem = Problem('robot')
robot.at = Fluent('robot.at', BoolType(l, position=Location)) connected = Fluent('connected', BoolType(l, 1_from=Location, 1_to=Location))
problem.add_fluent(robot.at, default_initial_value=False)
problem.add_fluent(connected, default_initial_value=False)

move = InstantaneousAction('move', 1_from=Location, 1_to=Location)
move.add_precondition(And(robot.at(1_from), connected(1_from, 1_to)))
move.add_effect(robot.at(1_to), False)
move.add_effect(connected(1_to, True))
problem.add_action(move)

problem.add_objects(location_map.values())
problem.set_initial_value(robot.at(locations[INIT]), True)
for f, t in location_map.items():
    problem.set_initial_value(connected(locations[f], locations[t]), True)
problem.add_goal(robot.at(locations[DEST]))

# We can now solve the problem with any planner installed supporting this problem kind!
# The library can detect the used syntactical features and can filter suitable planners

# 'OneshotPlanner' is just one of the 'Operation Modes' supported by the library!

with OneShotPlanner(problem, kind=problem.kind) as planner:
    result = planner.solve(problem)
    if result.status == PlanGenerationResultStatus.SOLVED_SATISFYING:
        print('One solution found: plan: %s' % (result.plan))
    else:
        print('No plan found!')