650.045 – Advanced Topics in A.I. 2 Winter Term, 2022/23

Mini-project: Reverse planning

Francesco Bombassei De Bona - 12138677

1 Introduction

As mini-project I choose the proposed task A. The goal of this task is to implement a STRIPS planning domain where there is at least one action that is uniformly reversible. I implemented a domain both in PDDL and in K.

2 Project structure

The project is structured as follows:

- **report.pdf**: this report
- **pddl**: contains the encoding of the domain in PDD and an example problem
- **k**: contains the encoding of the domain in K, an example problem and the background information for the problem
- **img**: contains the images used in this report

3 Domain

As domain I've chosen to imagine a moving company that has to move furniture between various locations. The company has to perform some actions in order to be able to move the furniture. After performing such actions, the company can move the furniture from one location to another. I identified the following actions:

- **assemble**: assemble a piece of furniture
- **disassemble**: disassemble a piece of furniture
- **pack**: pack a piece of furniture
- **unpack**: unpack a piece of furniture
- **load**: load a piece of furniture into a truck
- **unload**: unload a piece of furniture from a truck
- **move**: move a piece of furniture from one location to another

The company has to perform the following actions in order to be able to move a piece of furniture from one location to another:

1. disassemble the piece of furniture
2. pack the piece of furniture
3. load the piece of furniture into a truck
4. move the truck to the destination location
5. unload the piece of furniture from the truck
6. unpack the piece of furniture
7. assemble the piece of furniture

Accordingly to the previous actions and to the domain description, I've defined the following predicates:

- **assembled**: a piece of furniture is assembled
- **packed**: a piece of furniture is packed
- **loaded**: a piece of furniture is loaded into a truck
- **moved**: a piece of furniture is moved from one location to another

To keep the domain simple and focus more on the uniformly reversible actions, I've decided to assume that the company don't track the location of the trucks.

In Figure 1 there is a graphical representation of the domain states and effects of actions.

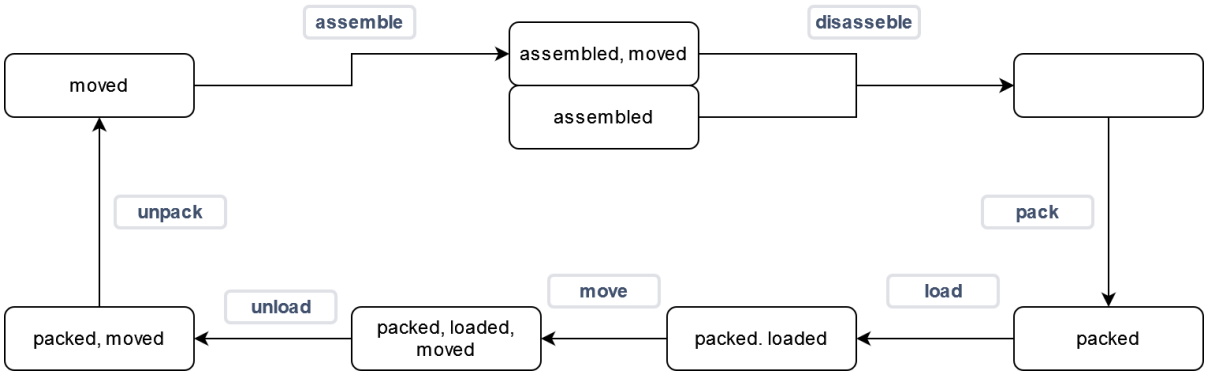

Figure 1: Domain states and effects of actions

4 PDDL

4.1 Domain description

The PDDL encoding of the domain is in the file `pddl/domain.pddl`. I defined two types of objects: `furniture` and `truck`. Each action has as parameters a piece of furniture and, additionally, the actions `load` and `unload` have as second parameter a truck. Every action has fixed preconditions over the four possible predicates, except for the action `disassemble` that is a valid action in any state were a piece of furniture was either moved or not. As for the effects, there are three couples of actions where each action of the couple is the reverse of the other, namely: `assemble` and `disassemble`, `pack` and `unpack` and `load` and `unload`.

4.2 Problem description

The PDDL encoding of the problem is in the file `pddl/problem.pddl`. The problem is defined as follows:

- **Initial state**: the initial state is defined as follows:
  - `assembled` is true for the piece of furniture `couch` and `kitchen`
- **Goal state**: the goal state is defined as follows:
  - `assembled` is true for the piece of furniture `couch` and `kitchen`
  - `moved` is true for the piece of furniture `couch` and `kitchen`

The problem is solvable by the planner and the solution is the represented in Figure 2.

4.3 Uniformly reversible actions

Starting from the two previous files, I produced the translation of the domain and the problem in the `ASP` format. The shell command used to produce the translation is the following:

```
plasp translate domain.pddl problem.pddl | cat - <(echo "\#show chosen/1.") > domain.lp
```

As for the previous command, to the original translation I added the directive `#show chosen/1.` to show the chosen actions. Using the command `clingo domain.lp --const horizon=6 sequential-horizon.uurev.lp -n 0` I could find all the uniformly reversible actions. As expected, there are six actions that are uniformly reversible where only one plan reverse the initial state to the state previous to the chosen action. The uniformly reversible actions are the following:

- `assemble`

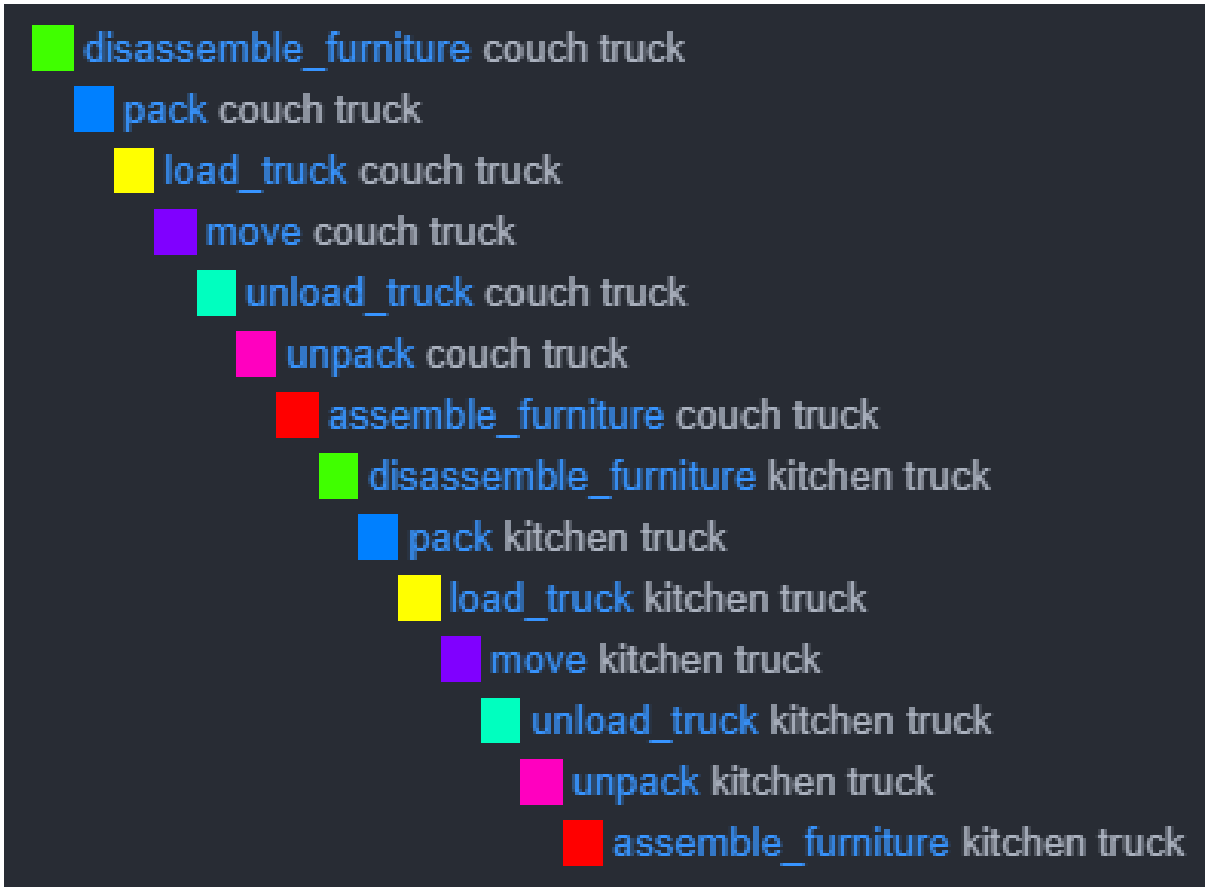

Figure 2: Solution to the problem

- `pack`
- `unpack`
- `load`
- `unload`
- `move`

In the Figure 3 I show the six uniformly reversible actions and the plan that reverse the initial state to the state previous to the chosen actions. For simplicity, I've changed the background information of the problem to have only one piece of furniture and one truck. Changing the background information of the problem doesn't affect the uniformly reversible actions, that increase proportionally to the number of pieces of furniture and trucks. The action `disassemble` is not uniformly reversible because there are two different initial states for the action, as shown in Figure 1, and only one has a reverse plan that reverse the initial state to the state previous to the chosen action (the one where it holds that {`assembled, moved`} is true for the piece of furniture).

```
❯ clingo domain.lp --const horizon=6 sequential-horizon.uurev.lp -n 0
clingo version 5.6.2 (d469780)
Reading from domain.lp ...
Solving...
Answer: 1
chosen(("unload_truck",constant("couch"),constant("truck"))) plan(("unpack",constant("couch"),constant("truck")),1) plan(("assemble_furniture",constant("couch"),constant("truck")),2) plan(("disassemble_furniture",constant("couch"),constant("truck")),3) plan(("pack",constant("couch"),constant("truck")),4) plan(("load_truck",constant("couch"),constant("truck")),5) plan(("move",constant("couch"),constant("truck")),6)
Answer: 2
chosen(("assemble_furniture",constant("couch"),constant("truck"))) plan(("disassemble_furniture",constant("couch"),constant("truck")),1) plan(("pack",constant("couch"),constant("truck")),2) plan(("load_truck",constant("couch"),constant("truck")),3) plan(("move",constant("couch"),constant("truck")),4) plan(("unload_truck",constant("couch"),constant("truck")),5) plan(("unpack",constant("couch"),constant("truck")),6)
Answer: 3
chosen(("unpack",constant("couch"),constant("truck"))) plan(("assemble_furniture",constant("couch"),constant("truck")),1) plan(("disassemble_furniture",constant("couch"),constant("truck")),2) plan(("pack",constant("couch"),constant("truck")),3) plan(("load_truck",constant("couch"),constant("truck")),4) plan(("move",constant("couch"),constant("truck")),5) plan(("unload_truck",constant("couch"),constant("truck")),6)
Answer: 4
chosen(("move",constant("couch"),constant("truck"))) plan(("unload_truck",constant("couch"),constant("truck")),1) plan(("unpack",constant("couch"),constant("truck")),2) plan(("assemble_furniture",constant("couch"),constant("truck")),3) plan(("disassemble_furniture",constant("couch"),constant("truck")),4) plan(("pack",constant("couch"),constant("truck")),5) plan(("load_truck",constant("couch"),constant("truck")),6)
Answer: 5
chosen(("load_truck",constant("couch"),constant("truck"))) plan(("move",constant("couch"),constant("truck")),1) plan(("unload_truck",constant("couch"),constant("truck")),2) plan(("unpack",constant("couch"),constant("truck")),3) plan(("assemble_furniture",constant("couch"),constant("truck")),4) plan(("disassemble_furniture",constant("couch"),constant("truck")),5) plan(("pack",constant("couch"),constant("truck")),6)
Answer: 6
chosen(("pack",constant("couch"),constant("truck"))) plan(("load_truck",constant("couch"),constant("truck")),1) plan(("move",constant("couch"),constant("truck")),2) plan(("unload_truck",constant("couch"),constant("truck")),3) plan(("unpack",constant("couch"),constant("truck")),4) plan(("assemble_furniture",constant("couch"),constant("truck")),5) plan(("disassemble_furniture",constant("couch"),constant("truck")),6)
SATISFIABLE

Models       : 6
Calls        : 1
Time         : 0.015s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.009s
```

Figure 3: Uniformly reversible actions

5 K

5.1 Domain and problem description

The same domain and problem described in Section 4 can be encoded in the `K` language. The `K` encoding of the domain is in the file `k/k_domain.plan`. The `K` encoding of the problem is in the file `k/k_problem.plan`. The `K` encoding of the domain is analogous to the one presented in the PDDL section. It was necessary to add the `inertial` property to all the fluents and to specify the `noConcurrency` option. Also a file containing the background information of the problem was necessary, just to specify the initial objects in the problem.

Running the DLV^k planner on the domain and the problem resulted in the same solution as the one found by the PDDL planner. This can be seen in Figure 4.

```
❯ dlv k_domain.plan k_problem.plan k_background.dl -FP
DLV [build BEN/Dec 16 2012   gcc 4.6.1]

STATE 0: assembled(couch)
ACTIONS: disassemble_furniture(couch,truck)
STATE 1: -assembled(couch), -moved(couch)
ACTIONS: pack(couch,truck)
STATE 2: packed(couch)
ACTIONS: load_truck(couch,truck)
STATE 3: packed(couch), loaded(couch,truck)
ACTIONS: move(couch,truck)
STATE 4: packed(couch), loaded(couch,truck), moved(couch)
ACTIONS: unload_truck(couch,truck)
STATE 5: packed(couch), moved(couch), -loaded(couch,truck)
ACTIONS: unpack(couch,truck)
STATE 6: moved(couch), -packed(couch)
ACTIONS: assemble_furniture(couch,truck)
STATE 7: moved(couch), assembled(couch)
PLAN: disassemble_furniture(couch,truck); pack(couch,truck); load_truck(couch,truck); move(couch,truck);
unload_truck(couch,truck); unpack(couch,truck); assemble_furniture(couch,truck)

Check whether that plan is secure (y/n)? y
The plan is secure.
```

Figure 4: Solution to the problem

5.2 Reverse plan

I tried to use `revplan` software to find the reverse plan of the solution found by the DLV^k planner but after running the command `java -cp plan-library-binary.jar planlibrary.ReverseDomain -x k_domain.plan k_background.dl cond.xml /usr/bin/dlv 15` for more than 24 hours, the software didn't find terminate at all (Figure 5). Even if the

```
❯ java -cp plan-library-binary.jar planlibrary.ReverseDomain -x k_domain.plan k_background.dl cond.xml /u
sr/bin/dlv 15
Parsing domain: k_domain
Finished parsing domain ... starting to compute reverse plans using DLV
./scripts/runDLV.sh /usr/bin/dlv 15 opt "/mnt/c/Users/Francesco/Desktop/ai2/k/k_background.dl" "/mnt/c/Us
ers/Francesco/Desktop/ai2/k/k_domain.simrev.plan" "/mnt/c/Users/Francesco/Desktop/ai2/k/k_domain.simrev.o
t"
```

Figure 5: Reverse plan

software didn't terminate, from the uniformly reversible actions retrieved in Section 4.3 and what shown in Figure 1, I can state that the reverse plan for each combination of furniture and goal state corresponds to the plan to reverse each single uniformly reversible action.

Only the cases of the goal states {`assembled`} and {`assembled, moved`} must be discussed. Regarding the second case, the reverse plan corresponds to the concatenation of the actions {`disasseble, pack, load, move, unload, unpack, assemble`}. The first case instead has no reverse plan because there is no actions sequence that can remove the single predicate `moved` from the state {`assembled, moved`}.