## Planning and Optimization

M. Helmert, T. Keller
S. Eriksson, F. Pommerening, S. Sievers

University of Basel
Fall Semester 2019

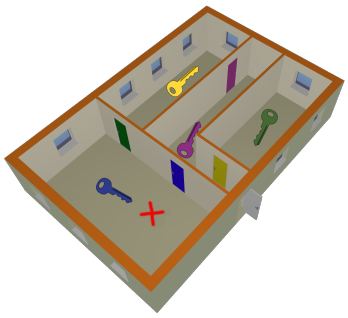
# Exercise Sheet A

**Due: October 06, 2019**

*The exercise sheets can be submitted in groups of three students. Please submit one single copy of the exercises per group (only one member of the group does the submission), and provide all student names on the submission. The files required for this exercise are in the directory* `exercise-a` *of the course repository (*`https://bitbucket.org/aibasel/planopt-hs19`*). All paths are relative to this directory. Update your clone of the repository with* `hg pull -u` *to see the files.*

**Exercise A.1** (3+3+2+2 marks)

Consider the following planning domain: An agent is moving on a map and is trying to reach a specific target. However, there are locked doors between some locations that can only be unlocked by first picking up their corresponding key from some other location.
Example instance with four rooms:



(a) Model the domain as a family of FDR tasks parameterized by a graph $G = \langle N, E \rangle$ representing the map, start and target location of the agent $s, t \in N$, a set $L \subseteq E$ of locked doors, a set *Keys* of keys and two functions *unlocks* : *Keys* $\to L$ denoting which door a key unlocks and *initial-location* : *Keys* $\to N$ denoting where each key is located in the beginning. Formally, you are defining an infinite number of planning tasks, one for every possible choice of $G, s, t, L$, *Keys*, *unlocks* and *initial-location*. Make sensible choices for your FDR variables and their domains.

(b) Transform your model into an equivalent family of STRIPS planning tasks. List the mutex groups that would induce the FDR variables you used in part (a).

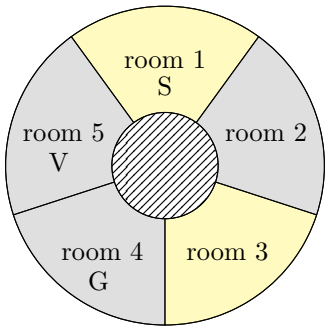
(c) Consider the example instance above in the FDR and STRIPS model from (a) and (b). Compare the state spaces of the two formulations. How many states do they have? How many of those states are reachable? Are the state spaces the same? Are they equivalent? What remains the same, what changes?

(d) Discuss advantages and disadvantages of each formulation. Look at different theoretical and practical angles in your discussion.

**Exercise A.2** (3+2+3+2 marks)
Consider the following planning task:

- You are trapped in the cellar of a building with a switch board full of light switches. In the rooms above you there is a vampire (V). Luckily, there also is a vampire slayer (S) in those rooms. To keep things simple, we consider only room layouts that are circular corridors where each room has a clockwise and an anti-clockwise neighbor.
- The vampire avoids the light: whenever the light in the vampire's room is switched on, it moves to a neighboring room. If one of the neighboring rooms is dark, it will move there, preferring the anti-clockwise one if both are dark. If both neighboring rooms are bright, it will move clockwise.
- The slayer tries to stay in the light. If the light in her room is switched off, she moves to a neighboring room. She moves clockwise if that room is bright and anti-clockwise otherwise.
- If the two of them meet in a room they will fight. The vampire wins the fight in a dark room unless there is garlic (G) in that room. In bright rooms or in rooms with garlic, the slayer wins.
- All you can do is use the switch board to toggle lights and watch the fight, when it happens. Your objective is to toggle the lights so that the slayer can win the fight.

Example instance with five rooms:




(a) There is a partial model of this domain in the directory `vampire`. Complete it by adding the effects of `toggle-light` and `watch-fight`. Do not add new actions or predicates.

*The directory also contains instances which you can use for debugging. Their optimal plan costs are 6, 4, 7, 5, 4, 12, 11, 10, 13, and 8.*

*Note that for technical reasons, we cannot use VAL but will use INVAL from now on. To install it (one-time setup), please run the script* `install-inval.sh` *in the directory* `INVAL` *directory of* `exercise-a`*. Aftwards, you can use INVAL by running* `inval` *from anywhere on your (virtual) machine.*

(b) PDDL uses first-order predicate logic to model planning tasks. However, the models discussed in the lecture are all based on propositional logic. Most planners convert PDDL into one of the propositional models in a step called *grounding*. The directory `preprocess` contains a Python tool to do this step. The call

```
./preprocess/ground.py vampire/domain.pddl vampire/p01.pddl
```

will create a new domain file `vampire/domain_grounded_for_p01.pddl` and a new task file `vampire/p01_grounded.pddl`. Repeat this for all task files and describe the effect of the grounding procedure.

(c) In addition to `ground.py` there is an incomplete Python program called `transform.py` in the directory `preprocess` which should transform grounded domains into conflict-free effect normal form. Complete the missing parts and use it to transform your grounded domains from exercise (b) into conflict-free effect normal form. The call

```
./preprocess/transform.py vampire/domain_grounded_for_p01.pddl
```

will create the file `vampire/domain_grounded_for_p01_normalized.pddl`.

(d) Use Fast Downward to generate plans for all tasks using the domains you created in exercises (a)–(c). Then use INVAL to validate each plan against each domain formulation. In which combinations are the plans valid? Discuss the reason for that.

**Exercise A.3** (10 marks)

*This exercise is a literature research question. We might repeat this kind of exercise from time to time. The goal of such exercises is to find information in research papers. We will provide you with starting points for your search. We don't expect you to fully read all papers. Instead, try to extract the relevant information to answer the question. This time, you will find all required information in the given papers, but in the future, you might need to follow references.*

Provide an overview of the complexity of different classes of planning tasks. In particular, your overview should cover classes that are:

- polynomial
- NP-complete
- PSPACE-complete
- semi-decidable or undecidable

The overview should be in the form of a written text, not only bullet points and tables. One page should be sufficient to answer this question in detail.
You'll find the necessary information in the following papers:

- Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2), 165–204.
- Helmert, M. (2002). Decidability and Undecidability Results for Planning with Numerical State Variables. In Proceedings of AIPS, 44–53.
- Erol, K., Nau, D. S., & Subrahmanian, V. S. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artificial intelligence*, 76(1–2), 75–88.