CPS822: Assignment 6: 100 points.

Upload an ELECTRONIC copy of your assignment into a folder on D2L before the deadline. SC axioms in LaTeX are preferred, or include images of a legible hand-written copy. Remember to include all files.

> Note that **no** communication whatsoever is allowed between groups: similar solutions submitted by different groups will be heavily penalized. According to the Course Management Form, the groups who share code or parts of their solutions will get **0** for this assignment and an extra penalty up to negative 4% of the final grade. Note you **cannot** use any external resources (including those you find on the Web) in your solutions. Failure to do this will have negative effect on your mark. By submitting this assignment you acknowledge that you read and understood the Policy on Collaboration in homework assignments and Contract Cheating policy stated in the Course Management Form (CMF).

This assignment is designed to let you get some hands-on experience with different AI planning techniques and learn conceptual differences between them. Also, you will learn to translate planning problems from one representation to another. In particular, in this assignment, you have to develop your own planning domain as a situation calculus basic action theory, which includes precondition axioms and successor state axioms. You will use a familiar CPS721-style planner and a new A* planner. Both are implemented in PROLOG. (Warning: recall that in PROLOG you need capital letters for variables and lower-case letters for constants representing names of the objects.) In addition, from this assignment, you will get experience of experimental evaluation of different planning techniques and experience of collecting data from an efficient program for best-first planning based on A^*-search.

(1) First, develop your own planning domain with at least 5-10 fluents and 5-10 actions. Save your file as **domain.pl** All the actions should have different meaning, and the fluents should represent significantly different properties. This can be related to solving any CS problems, such as sorting, process coordination, database transactions, or it can be about any realistic problem taken from the real life. Just to give you a few examples, there are planning domains about changing a flat tire, replacing a broken compressor attached with screws to the back panel of a refrigerator, cooking dishes from several ingredients using different techniques, shaking and dispensing cocktails in a pub to customers, serving sandwiches to children with allergies, scuba diving in caves on the coast of Australia, dressing for a party, and many others. Some of these domains are described in popular textbooks, or used as benchmarks in International Planning Competitions (IPC) to evaluate performance of different planners. Your domain should be **new**, and it must not be similar to any of the previously developed planning domains. It is expected that domains developed by different students will be significantly different from each other. Moreover, planning in your domain should require a significant degree of *combinatorial search*, e.g., reaching a goal from an initial state should involve evaluating several tentative sequences of actions. If you developed situation calculus precondition and successor state axioms for a bonus question in the 5th assignment, you can adapt them to this assignment. The TA will check all submitted domains carefully. (You can consult with the TA about your proposed planning domain, if you are not sure.) In addition, you have to decide how many objects (implemented as constants) will be present in your domain: they can be arguments of your actions and fluents. Your planning domain should be preferably in ADL (i.e., SSAs should have context conditions), but it is ok if it is in STRIPS (i.e., actions have unconditional effects). Once you have developed your planning domain, create at least 10 planning instances (problems) of varying complexity with a different number of objects. You are welcome to consider a few large domains (say, with up to 25-50 objects) to test the limits of the given planners, but also include a few simple problems for which you and the TA can easily see what a valid plan may look like. Your new domain should not be too simple for the purposes of this assignment. You can use types, or you can develop a domain without types. Try to avoid universally

quantified effects in actions, and consider only actions with simple effects. Make sure that your Prolog implementation is correct. Test your program by solving a few simple planning problems using the initial and goal states of your choice. Write brief comments in your file *domain.pl* to explain your rules. You have to print your Prolog code and hand it in the lab.

(2) Second, you will be evaluating the automated planners given to you in the files *dfsPlannerWithListOfNodes.pl* and *aStarRevised.pl*. The first planner is very similar to the CPS721 planner. It does iterative depth-first search, and it does not have any heuristics. The second planner uses a domain independent heuristic *planningGraphHeuristicRevised.pl* that is also given to you. You can query the second planner as *?- plan(100,graphplan,Sit).* and it will return a computed plan as a substitution for the variable Sit. Both planners require a special encoding of the domain. You are given as examples the encoding of the Blocks World, Elevator (Miconic), and Gripper domains. It is important that your planning domain should follow the same encoding. The details of this encoding are explained in the file *situation2StateUpdate.pl* given to you, and in the provided sample file *gripperRevised.pl*

- Run both planners with your own planning problems *test1.pl ... test10.pl* using your own planning file *domain.pl* Each time save output to a file. Start with the easier instances, but make sure you try also a more challenging planning problems. Make sure that at least one planner runs successfully, produces a valid plan, and prints how long it took to compute a plan for each problem. Each output file should include total time in seconds. You have to submit on D2L a ZIP file **myData.zip** that includes all your data and output files. There is no need to print them, but they must be submitted.

- Produce a table (e.g., a spread-sheet) summarizing how much time it took for each planner to solve each of your planning problems. Make conclusion which planner was the best in your evaluation, i.e., which planner (and search options, if any) took the least time across all of your planning problems. Explain your results. If you could not identify the "winner", explain why. Save your table, explanation, and conclusion in the file **Report1.pdf** Zip both files *myData.zip* and *Report1.pdf* together into a single **yourLoginName.zip** file where yourLoginName must be a university user name of the student who uploads this file on D2L. Only one ZIP file from the group.

The well-known FastDownward (FD) planner software is installed in the */opt/fast-downward* folder on moon Linux machines. In particular, this folder includes numerous benchmarks (from the recent international planning competitions) in */opt/fast-downward/misc/tests/benchmarks* To browse instances of the planning problems available in each domain, list the corresponding benchmark folder. The low-numbered problems have fewer objects and are usually easier, while the high-numbered problems use more objects and their solutions may require longer plans. To familiarize yourself with some of the well-known planning domains, you might wish to read their descriptions provided at

`https://helios.hud.ac.uk/scommv/IPC-14/domains_sequential.html`

The most recent open-source release of FD is at `https://github.com/aibasel/downward` However, you do not need it to complete this homework. This is just for your information.

(3) Third, collect data by solving your planning problems using both the CPS721 planner and the new A* planner with declarative heuristics. Before you do this, include as many declarative heuristics as you can into your *domain.pl* Prolog code and save the resulting file as **domainWithDeclarativeHeuristics.pl** Recall from CPS721 that the declarative heuristics serve to cut the useless parts of a search tree; they can be implemented by the rules with the predicate $useless(A, List)$ in the head. If they are clever, they help to speed-up computation significantly. Run each instance of the planning problems that you created in the

previous step. Ideally, you should try to run same instances with both A* and with the *dfsPlannerWithListOfNodes.pl* planner. Before you can run them, remove the commented out lines related to predicate $useless(A, S)$. Include in your second report **Report2.pdf** the collected data about computation time. Do analysis and comparison of all data from your assessment experiments. Summarize your observations and make conclusions in your report. In particular, say which of the 4 (with/without declarative heuristic) planners was the most efficient in your evaluation. Include your file **Report2.pdf** into your ZIP.

(4) Fourth, translate your planning domain *manually* from PROLOG into the BAT situation calculus representation in first order logic. In particular, you will have to write precondition axioms, effect axioms in the normal form, and obtain the successor state axioms (SSAs) as you did in the 5th assignment. Be careful, when you write the frame axioms (i.e., the explanation closure axioms), since they have to mention all the negative effects. If you write your solution by hand, please write legibly. Make an electronic copy of your solution **SCaxioms** together with other parts of your assignment. Include a legible copy of your situation calculus axioms **SCaxioms** when you upload your homework on D2L.

How to submit this assignment.

Hand in printouts of the following files before beginning of the lab: **domain.pl Report1.pdf SCaxioms domainWithDeclarativeHeuristics.pl Report2.pdf** On the first page of your printout **write the name of your electronic ZIP file** to make sure the TA can locate it easily among other ZIP files. **Staple all pages!** If you wrote your Prolog code on a Windows machine, make sure you save your Prolog files as plain text that one can easily read on Linux machines. Before you submit your Prolog code electronically make sure that your files do not contain any extra binary symbols: it should be possible to load your Prolog program into a recent release of ECLiPSe Prolog, compile your program and ask testing queries. TA will mark your assignment using ECLiPSe Prolog. If you run any other version of Prolog on your home computer, it is your responsibility to make sure that your program will run on ECLiPSe Prolog (release 6 or any more recent release), as required. For example, you can run a command-line version of *ECLiPSe* Prolog on moon remotely from your home computer to test your program. To submit files electronically create a **zip** archive:

```
zip yourLoginName.zip domain.pl Report1.pdf SCaxioms myData.zip
      domainWithDeclarativeHeuristics.pl Report2.pdf
```

where `yourLoginName` is a university login name of the person who submits this assignment from a group. Recall that myData.zip archive file must include all your planning instances `test1.pl ... test10.pl` and the files with output produced from test files.

Remember to mention at the beginning of each report file *student numbers* and *names* of all people who participated in discussions (see the course management form). You may be penalized for not doing so.

Second, upload your file `yourLoginName.zip` to D2L before the deadline.