CS 481 — AI — Project 1 — Blocks World RBS Planner

Project 1 — Blocks World RBS Planner
Introduction

You will recall the lecture discussions of the Blocks World, State-Space Search, and Rule-Based Systems (RBSs). In this project, we will build a Blocks World RBS planner that will take a Blocks World rule-set, a starting state and a goal state and construct a sequence of gripper operations that will convert the starting state into the goal state. It will also employ a reasonable heuristic function to avoid "expanding" too many nodes in the search.

We will take the rule-set from Rules 4-7 Luger, section 8.4. We will assume we've upgraded to the "Smart Gripper", which can move to the correctly named block (eg, via sensory input). Thus, we have the 4 Blocks World Rules in our Rule Base System (RBS). Each rule represents a set of potential operations on an RBS state in a State-Space Search for a goal state.

One sample Start State (SS) is from Luger Fig 2.3, page 67. Another sample SS is from Luger Fig 8.18, age 315. As a heuristic function, we will use an (unordered) Hamming distance of WMEM assertions vs grounded rule pre-condition expressions (exprs). One sample goal state (GS) will be to get the blocks into an alphabetic tower with A on top, B next down, etc. Another sample GS is to get the blocks in a tower in reverse alphabetic order, with A on the bottom, B next up, etc.

What we particularly want, however, is a plan – that is, a list of the states on a path from SS to GS, or more particularly, a list of the grounded rules (which are gripper "move" operations) connecting one state to the next. Moreover, during processing, a log will be printed for each state "expanded" indicating the mom state (by name is sufficient), the grounded-rules (to be discussed in lecture) generated along with their unifiers, the child state generated from each grounded-rule, and its Hamming distance to the GS, and the updated RBS search state of the Open and Closed lists.

A sample lispified set of rules will be provided and discussed along with a few infrastructure functions to help out.

Teams

We recommend working in a team. Your team may contain from one to four members. (We would prefer large teams.) Pick a short-ish (<= 12 letter and/or number) name for your team (e.g., "ABX", or "Groggy", or "we-like-cats", etc.). [For the next project, team membership, and names, can be changed.]

Heuristic Function, F(N) = G(N) + H(N)

We will use an unordered Hamming distance between two lists of exprs. The distance value increments by one for each expr in the first list that is not in the second, and also by one for distance between to lists of exprs. The distance value increments by one for each expr in the second list that is not in the first. For example: this Hamming distance between the groups

```
  ((r b c) (r c d) (s b) (p e))
```

and

```
  ((r c b) (r c d) (s b) (p a) (p e))
```

is 5 because these 5 exprs are not common to both groups:

```
  (r b c) (p e) (r c b) (r c d) (p a).
```

Goal Completion

Once your searh reaches the goal state, it should stop.

Project Development Reporting

Standup Status Report, twice weekly. The Standup Status Report is due **Monday's** and **Friday's** by noon-ish. One report per team, **CC'ing the other team members**. It should contain, team name and the member names. This documents should be delivered **as a PDF file**; and the **filename** should be in the following format: include your course and section number, project number, your team name, the document type (Standup), and the date as YYMMDD:. E.g., "`335-02-p1-Groggy-Standup-210231.pdf`". ("02" or "03" is your section.)

Standup Status Report contents should be, for each team member, a list of the 3 **Standup question short answers**: Q1: what have you **completed** (name sub-tasks) by the time of this Standup report; Q2: what do you **plan to complete** (name sub-tasks) by the time of the next Standup report; and Q3: what obstacles if any (1-line description) are **currently blocking you** (for which you've reasonably tried to find the answers by yourself, including asking your team about them – known as "due diligence"). Note, that you can email the professor, or ask questions during office hours to get answers.

Readme File

When your project is complete, you should provide a README.txt text file. Be clear in your instruction on how to build and use the project by providing instructions a novice programmer would understand. If there are any external dependencies for building, the README must also list them and how to find and incorporate them. Usage should include an example invocation. A README would cover the following:

- Class number
- Project number and name
- Team name and members
- Intro (including the algorithm used)
- Contents: Files in the .zip submission
- External Requirements (None?)
- Setup and Installation (if any)
- Sample invocation
- Features (both included and missing)
- Bugs (if any)

Academic Rules

Correctly and properly attribute all third party material and references, lest points be taken off.

Submission

All Necessary Files: Your submission must, at a minimum, include a plain ASCII text file called

Submission

All Necessary Files: Your submission must, at a minimum, include a plain ASCII text file called
`README.txt`, all project documentation files (except those already delivered), all necessary source files to allow the submission to be built and run independently by the instructor. [For this project, no unusual files are expected.] Note, the instructor not use use your IDE or O.S.

Headers: All source code files must include a comment header identifying the author, author's contact info (please, no phone numbers), and a brief description of the file.

No Binaries: Do not include any IDE-specific files, object files, binary **executables**, or other superfluous files.

Project Folder: Place your submission files in a **folder named** like your Standup report files: `335-02-p1-Groggy`.

Project Zip File: Then zip up this folder. Name the .zip file the **same as the folder name**, like `335-02-p1-Groggy.zip`. Turn in by 11pm on the due date (as specified in the bulletin-board post) by **submitted via emailed zip file(s) (preferred)**, or via accessible cloud (eg, Github, Gdrive, Dropbox) with emailing the accessible cloud link/URL. See the Syllabus for the correct email address. The email subject title should include **the folder name**, like `335-02-p1-Groggy`. Note that some email-programs block .ZIP files (but maybe allow you to change .ZIP to .ZAP) and block sending zipped files with .JS files inside (but maybe allow you to change .JS to .JS.TXT).

Email Body: Please include your team members' names (but not your IDs) at the end of the email.

Project Problems: If there is a problem with your project, don't put it in the email body – put it in the README.txt file, under an "ISSUES" section.

Grading

- 80% for compiling and executing with no errors or warnings
- 10% for clean and well-documented code (Rule #5(Clean))
- 5% for a clean and reasonable documentation files
- 5% for successfully following Submission rules