Wumpus World
AI Project

By
Abdullah Younis

For
CS-171
Introduction to Artificial Intelligence

Tournament Manual

Input

The input to the tournament can be a student submission archive, a EEE "AssignmentSubmission" archive, or any series of these. If EEE dropbox's have been abandoned since I wrote this project, please overhaul the input preparation script inside Resources/Scripts.

How to work it

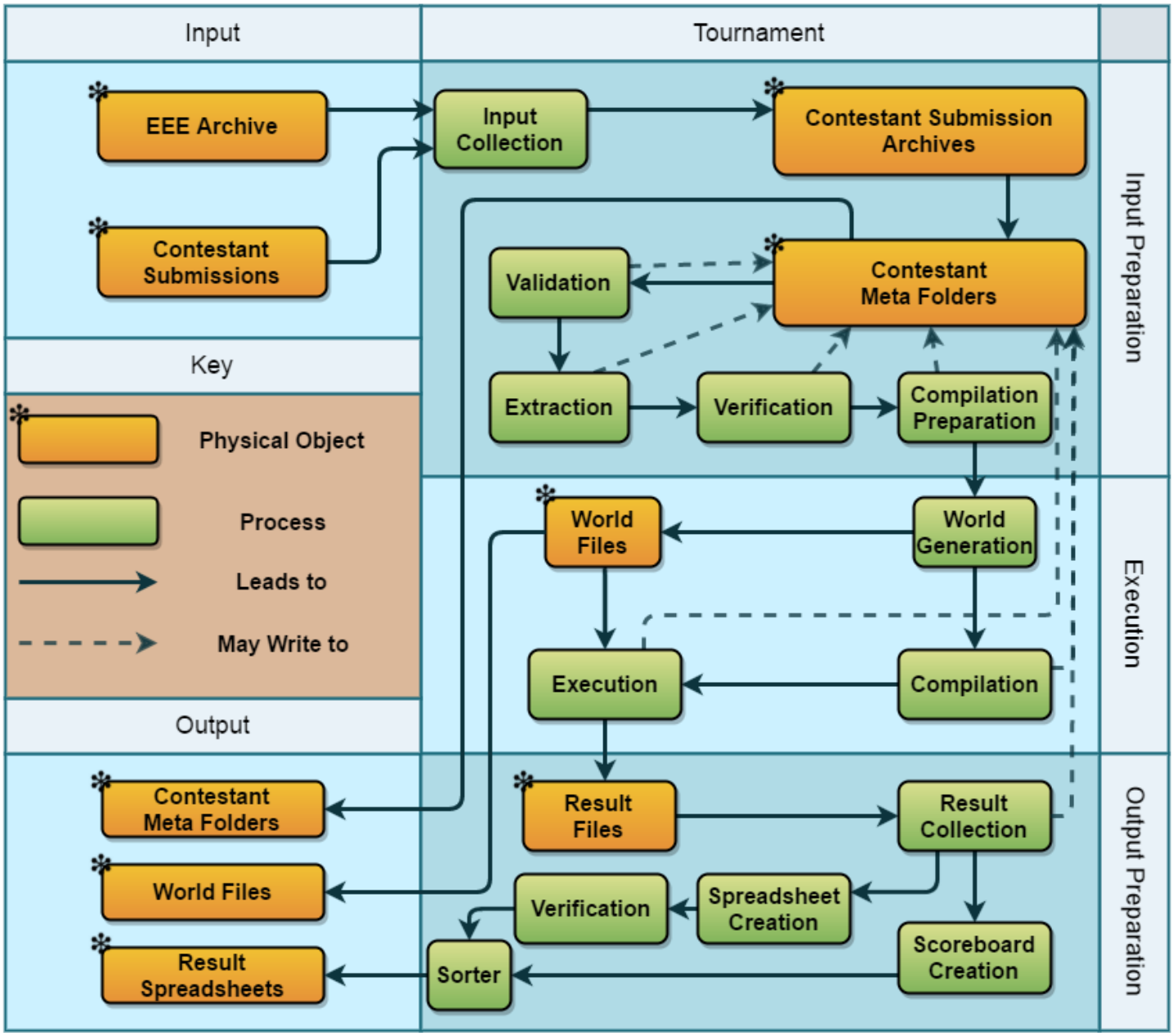
A standard run:

1. Upload the tournament folder to openlabs
2. Place all student submissions or EEE "AssignmentSubmission" archive into the input folder
3. Execute the command: **module load python/3.5.2**
4. Execute **make**
5. Check the Output folder

To use an already created set of worlds:

1. Upload the tournament folder to openlabs
2. Place all student submissions or EEE "AssignmentSubmission" archive into the input folder
3. Execute the command: **module load python/3.5.2**
4. Inside of Resources, create a folder called "Worlds" if it doesn't already exist
5. Places all of your world files directly in this folder
6. Navigate back to the root folder
7. Execute **make**
8. Check the Output folder

Tournament Design



The tournament program plays an important role in this project, and the above diagram represents the various steps employed throughout the whole process. Its primary job is to produce and organize scores for all contestants, which are submitted agents playing in the tournament. Nevertheless, the tournament plays a bigger role. It also verifies that all agents are following the rules of the tournament and of the project. The tournament program accomplishes all of this by breaking down the process into three parts: input preparation, execution, and output preparation.

Input Preparation

The input preparation portion of the tournament is responsible for collecting input, constructing necessary files for each contestant, and verifying that only valid contestants are allowed to play in the tournament. More specifically, input preparation consists of input collection, meta folder creation, validation, extraction, verification, and final preparations for compilation.

Input Collection

The tournament's first step involves collecting all submissions from the input folder; this step also extracts submissions from any EEE downloaded archive. An EEE downloaded archive starts with "AssignmentSubmission".

Meta Folder Creation

In this stage, all contestants receive a working directory inside a meta directory. These folders start with only the raw submission and a meta file, which serves as an important log file throughout the tournament. Each player's meta directory will be populated with more data at most every other stage of the tournament.

Validation

At this point, the submission archives undergo a validation check, which consists of filtering out invalid submissions and retrieving information from valid submission names. A contestant's submission can be invalid if it is not in "zip" format or if it doesn't contain the correct amount of underscores. Contestants with invalid submissions will be disqualified, and a report will be detailed in their meta file. Valid contestants will have their UCI Net ID, last name, ID number, and team name extracted from their submission name.

Extraction

The submission archives are extracted. Any failures here will lead to a contestant’s disqualification; the failure will be recorded in the contestant’s meta file.

Verification

Contestant's submission now undergo their final pre-execution check in the verification stage. Contestants are verified that they are following the rules of the project. Any submission must include a "pdf" report as well as valid source code of the MyAI class. If no "pdf" files are found, an error is recorded to contestant's meta file; however, if no source code is found the contestant is disqualified. In the case of a disqualification, as always, a report is made to the contestant's meta file.

Compilation Preparation

Contestant’s submission language has been determined already and MyAI classes have been extracted. The final portion of input preparation involves copying clean-slate versions of the remaining source code and placing it inside each contestant’s working folder.

Execution

The Execution portion of the tournament focuses on producing scores for each contestant. To accomplish this, it must first compile each contestant and generate a set of worlds. There is a large possibility for error in this section, so the tournament handles this portion with care.

World Generation

The first part of the execution step is to generate a set of worlds. The world generator is invoked multiple times to produce the desired number of each type of world. This set of worlds is then saved and used by each contestant's program. If the tournament program is started with a pre-existing set of worlds, this step is skipped.

Compilation

Each contestant's source code is then compiled. This means having an executable produced for the C++ agents, a jar file for the java agents, and python bytecode files for the python agents. Any error during this stage results in a disqualification and a report detailed in the contestant's meta file, which will include the compilation errors.

Execution

During the final step of execution, each submission will be executed as its own process against the set of worlds generated earlier. There will be a timeout feature implemented, which will disqualify any agent lingering longer than a predefined number of minutes and log an appropriate message. If an agent crashes on at least one world, this contestant will also be disqualified. Scores for each contestant will be produced and continued on to the output preparation stage.

Output Preparation

The Output Preparation stage is in charge of organizing results, packaging tournament files, and verifying contestants did not cheat. This is accomplished by collecting results, writing a teacher's spreadsheet, writing a contestant scoreboard, verifying results, sorting results, and packaging output.

Result Collection

In this step, all result files are read and scores are written to the contestant's meta file. Each meta file should be saturated with information now.

Scoreboard Creation

A scoreboard is constructed for the contestants detailing their team name, average score, standard deviation, language, and any errors. In addition to this, the scoreboard will record the reason for disqualification, if there was one. This scoreboard is created by reading each contestant's meta file and skipping over confidential information.

Spreadsheet Creation

A spreadsheet is constructed for the tournament instructors detailing all information, which includes contestant's UCI Net ID, last name, student ID, team name, submission name, average score, standard deviation, language, and any errors. This spreadsheet also details disqualification reasons for those that were disqualified. A hanging column is left blank for verification stage. This spreadsheet is created by reading each contestant's meta file.

Verification

This step checks the spreadsheet for suspicious information and flags any questionable results in the hanging column made by the previous step. A result may come into question if its score matches the score of another result or if its standard deviation is unusually low. A low standard deviation might mean that agent followed the same instructions for every world.

Sorter

The final step of the tournament sorts both the scoreboard and spreadsheet by descending score.

Output Packaging

After all is said and done, all desired output files are collected and organized into the output folder.

Tournament Policy Recommendations

Timeout Value

An agent shouldn’t take longer than thirty minutes to complete an absurd amount of worlds. With that said, if grading is based primarily on performance measure, every agent should have a fair opportunity at completing their task. The default timeout value is two hours; this can be changed inside the tournament’s makefile script. Two hours works great because every agent will be run simultaneously. This sets an upper-bound of around two and a half hours for the whole tournament process.

World Set

A set of ten thousand worlds should be plenty to get a good sense of how well an agent can think. The tournament uses twenty-five hundred 4x4 worlds and five hundred of each world size up to 7x7. This gives a total of ten thousand. Any change here would require an update in the world generation portion of the tournament as well as the makefile script of the world generator.