CS 360: Fall 2014
Programming Assignment 1
Tournament & Interface
Due: Friday, October 17, 2014 @ midnight

Cheng Ye

September 27, 2014

1 Go Text Protocol

The Go Text Protocol, GTP, is a text based protocol originally designed for the the well-known game "Go." However, GTP works well with many board games, such as Othello, therefore, we have been able to define an interface that enables two Connect-Five programs to play against each other in an automated fashion. A brief sdescription of the Gnu Go protocls can be found in `http://en.wikipedia.org/wiki/GNU_Go`. More details about GTP,GTP documentation, and the source code can be obtained from: `http://www.lysator.liu.se/~gunnar/gtp/`.

2 The Connect Five Interface

The protocol is asymmetric and involves (1) the controller, i.e., GTP, that acts as a kind of arbiter or relay and (2) the game engines, which will be your Connect Five playing program. All communication is initiated by the controller in form of commands, to which your Conncet Five engine will respond. The communication channel is assumed to be free from errors (i.e., those are handled at a lower level). Examples are UNIX pipes or TCP/IP connections.

The current interface is developed to play the role of the arbiter or controller in a program vs program competition (Connect Five engine 1 $\rightarrow$ controller (arbiter) $\rightarrow$ Connect Five engine 2). The controller relays moves between the two engines and alternately asks the engines to generate moves. This involves two different GTP channels, the first between the controller and engine 1, and the second between the controller and engine 2. Figure 1 illustrates. There is no direct communication between the two engines. The controller dictates board size, who makes the move, total time taken by the Connect Five engines to generate moves, who wins the Game or whether the game ends in a draw, and if a Game Engine has exceeded its time limit for generating moves, and, therefore, loses.

2.1 Interface Features

GTP minimally provides interfaces that allow input/output in the form of text commands, which makes tournament play quite simple: basically it loads the two executable clients, and sends text commands to each using a standard input and output interface. The clients wait to receive commands, perform their execution, and return a result again in standard text format. Note that black always plays first. Figure 2 illustrates the commnads sent by the controller, and expected responses from the two programs.

Here are the commands and information that GTP sends the Connect Five engines, and queries and information from the engines to the controller:

- name, ♯version, ♯protocol_version
- genmove black ♯cputime ♯move_number, genmove white ♯cputime ♯move_number

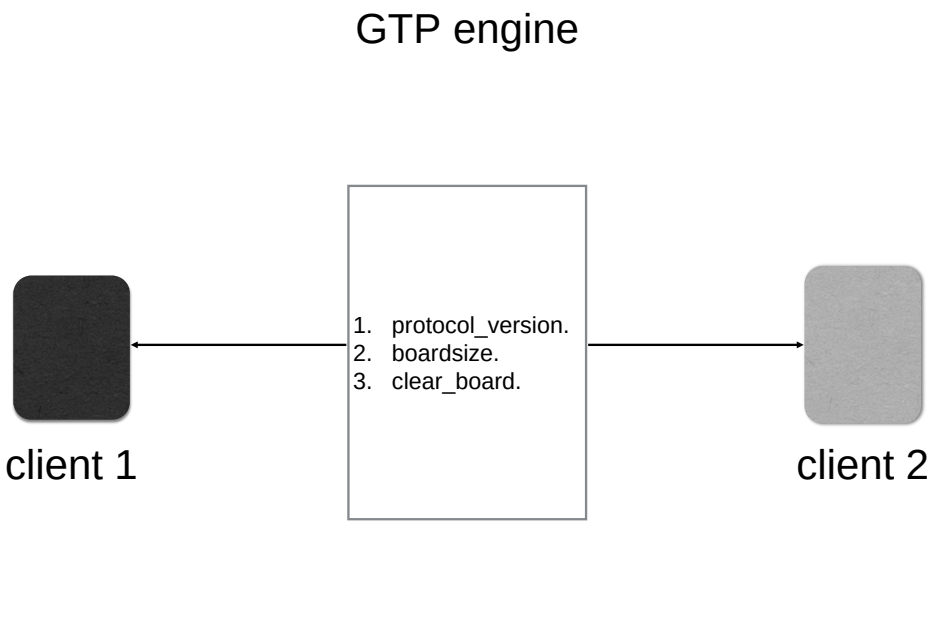

Figure 1: Initializing Game Controller

- play black ♯move, play white ♯move
- showboard
- ♯column_number
- quit

The ♯ sign above implies a number. However, the entire command is sent as an ASCII string. The controller expects that moves and other information sent to the controller will also be in ASCII format. When the game is over the controller returns the following information: winner, ♯final_score, ♯total_time_elapsed. The winner is either black or white, the final score is the number of moves played to win the game, and total_time_elapsed is the total CPU time used by the winner.

You can find the definition of the equivalent Go commands at the website:
`http://www.gnu.org/software/gnugo/gnugo_19.html`. For Connect Five the definitions are provided next. GTP provides engine vs. engine applications(`http://www.lysator.liu.se/~gunnar/gtp/`). "twogtp.py" should work well with different OS and engines written in different programming languages. You could use it to test the interfaces and your engine.

2.2 Details

The details are summarized below.

name, ♯version, ♯protocol_version

Returns the name and version of your game engine. For protocol_version we use the command format for version 2, so the controller returns "2"

genmove black ♯cputime ♯move_number, genmove white ♯cputime ♯move_number

We use command format of version 2: strings sent to the two engines genmove is a request to the particular player, black or white to generate a move, and ♯cputime is the CPU time that player has used till their last move. ♯move_number is the number of the current move being made. To this the corresponding player will return the move, which is the column number in which they want the next piece to be played (see Figure 2).

In case the correspond player's program thinks it has lost, it can send the move as the string "quit."

If the engine thinks it has won with this move, it just returns the move as a column number.

play black ♯move, play white ♯move

THis is the controller telling a player's program what move the other player just made, again reported as a column number. The program acknowledges the move by returning the empty string (see Figure 2).

showboard

If either white or black sends this command when it is their turn, controller returns the current status of the board in column format as a string.
\n _ _ _\n _ B _\n _ W _ \n".

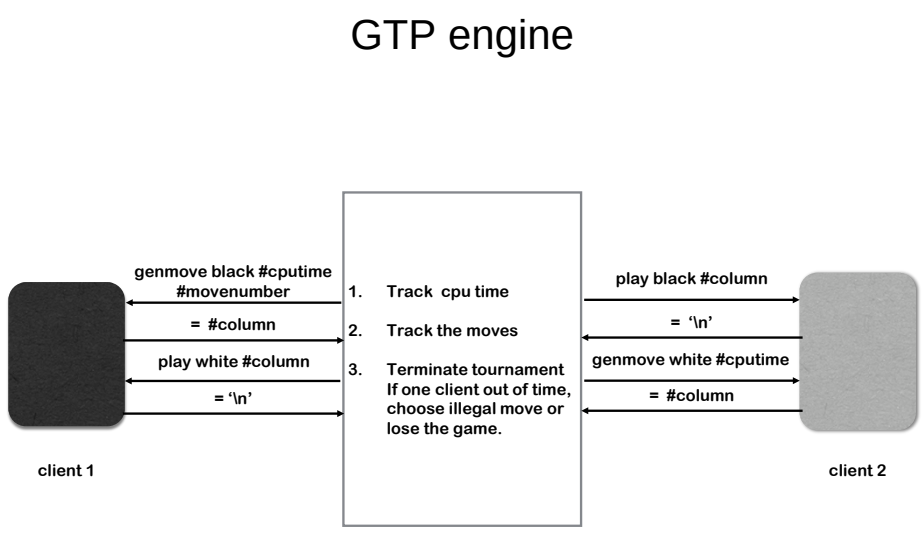

Figure 2: Communication between Controller and Game Engines

winner, final_score, total_time_elapsed

When the game ends, the controller returns a "1" if black wins, a "-1" if white wins, and "0" if the game ends in a tie.

It also returns the number of moves played to win − the smaller this number the better for the winner return "1" if you win, return "0" if lost or tie.

Important Hints

- All strings returned by the controller have format as "= string".
- You may configure global variables "debug" and "verbose" in "twogtp.py" to debug your code.
- For interfaces that return "nothing", you still need to write "= \n" back to stdout.
- "twogtp.py" uses module "popen2"to load subprocesses, which has been deprecated since python 2.6. However, "popen2" still works in python 2.7. You could replace "popen2" with "subprocess" if "popen2" doesn't work.`https://docs.python.org/2/library/subprocess.html`.
- You could develop your own GTP application, just make sure your engine is compatible with the interfaces.

3 Simple demo

Here is a simple engine demo written in python.It shows the output and commands generated by the controller, and replies, i.e., input from the game engines.

```
import sys
import time
cnt=1
name = sys.argv[1]
sys.stderr.write("get␣name␣" + name + '\n')
while True:
        # wait for twogtp.py
        result = ""
        line = sys.stdin.readline()
        while line != "\n":
            result = result + line
            line = sys.stdin.readline()
        # black resign
        if cnt == 9 and name == 'black' and result.find('genmove') >= 0:
                sys.stderr.write("I␣am␣" + name + "␣,␣I␣get␣" + result)
                print "=␣resign\n"
                sys.stdout.flush()
                sys.stderr.write("I␣am␣" + name + ",␣I␣resign\n")
        else:
                sys.stderr.write("I␣am␣" + name + "␣,␣I␣get␣" + result)
                if result.find("showboard") < 0:
                        # if receive quit
                        if result.find('quit') >= 0:
                                sys.stderr.write("I␣am␣" + name + "␣,␣I␣get␣" + result)
                                print "=␣\n"
                                sys.stderr.write("I␣am␣" + name + ",␣I␣exit\n" )
                                exit()
                        else:
                                if result.find('protocol_version') >= 0:
                                        print "=␣2\n"
                                        sys.stdout.flush()
                                else:
                                        # for any other command, just send back a number
                                        print "=␣" + str(cnt) + '\n'
                                        sys.stdout.flush()
                else:
                        print "=␣\n␣␣␣\n␣B␣\n␣W␣\n"
```

```
                    sys.stdout.flush()

        cnt += 1

        time.sleep(0.1)
```

Tournament:
python twogtp.py --white "python demo.py white" --black "python demo.py black" --games 1 --verbose 2
Here is a othello engine, which support GTP:
https://code.google.com/p/edax-reversi/source/browse/src/gtp.c