Cops & Robbers

The ICFP 2005 Programming Contest Task

The ICFP 2005 Programming Contest Committee

Version 7

1 Introduction

In the far-flung future of the year 2000, functional programming has taken over the world and so humans live in an almost unimaginable luxury. Since it's so easy, humans have used robots to automate everything, even law enforcement and bank robbery — the only job left to humans is to write their robots' control programs. In this year's task, you will play two roles in the world of 2000 by writing the control program that guides a Robber-Bot through a quiet urban neighborhood on a mission to rob every bank without getting caught, and the control program for a Cop-Bot dedicated to stopping it. Your Robber-Bot has the benefit of stealth; your Cop-Bot will work on a team with four other contestants' entries and have the power of numbers and coordination (not to mention an awesome late-model Police Robo-Cruiser).

The game takes place in a virtual Hyde Park, Chicago, in the year 2000. As the game begins, the Cop-Bots have just received word that a Robber-Bot was just spotted at a park on 54th and Ridgewood and are headed out in pursuit, either on foot for better awareness of their surroundings or in their Robo-Cruisers built for speed. They cannot see the Robber-Bot, but they may be able to infer its whereabouts: whenever it robs a bank they know about it, it occasionally leaves behind evidence that cops can use to pinpoint its past locations, and the Robber-Bot has a certain odor that Cop-Bots can sense when they are nearby.

Furthermore, though each Cop-Bot will be a different program from a different contestant with the ability to act independently, on every turn Cop-Bots can coordinate strategies with each other and share any knowledge they may have found. Their goal is to coordinate with each other to apprehend the robber with as little money stolen as possible.

Meanwhile the Robber-Bot — who knows the locations of all the Cop-Bots at all times — is trying to keep from getting caught while stealing as much money from Hyde Park's six banks as possible. It cannot outrun the cops, since it moves at the same pace as a cop on foot and can usually be overtaken by a Robo-Cruiser, but it can use uncertainty about its location to its advantage.

Your programming task is to write a program for a Robber-Bot and a program for a Cop-Bot that will work on teams with other contestants' Cop-Bots. You will have to make some hard decisions: as a Cop-Bot you need to work with others to catch the robber, but you need to get ahead of them to stand out. As the Robber-Bot you need to give away as little about your position as you can while at the same time robbing as many banks as you can, all the while keeping one step ahead of the Cop-Bots.

Your Cop-Bot will earn points for how quickly the Cop-Bot team as a group catches the Robber-Bot and individually for how much evidence it collects, how persuasive a leader it is, and whether it is the one that actually steps in and nabs the bad guy. Your Robber-Bot will earn points for how much money it can steal from banks without getting caught.

But that's not all: the most important part of the contest is that your programs will have to be written so they can easily adapt to changing requirements. After you've turned in your programs and have had a chance to rest and recuperate, we will announce a set of rule changes. Once we do, you'll have just 24 hours to modify your program to reflect them. We will run two tournaments, one for your original programs and one for the modified ones. As you might expect, your final ranking will depend more on the modified than the unmodified versions. The changes we make to the rules might be minor or major, but your code's flexibility will be the difference between a program that can barely follow the rules and a program that wins the contest.

The rest of this document explains the first phase of the competition in more detail. In the next section we explain the rules of the game and the scoring system we will use to determine the winners. In section 3 we explain the protocol that your programs will use to communicate with the game manager. In section 4, we explain the tournament structure.

The submission format is specified at the submission page, linked from the ICFP contest home page. Please see the submission instructions there.

`http://icfpc.plt-scheme.org/`

2 Rules

Cops & Robbers is a turn-based game where cops and robbers alternate moving from intersection to intersection on a map of Hyde Park. Each game has five Cop-Bots and one Robber-Bot, and each robot is controlled by a different contestant's program (the tournament consists of multiple games, see section 4 for details).

As the game progresses and the robbers and cops interact, they see snapshots of the state of the game, called worlds. Time in the game is marked by numbering those worlds. The robbers see the even numbered worlds and the cops see the odd-numbered worlds.

When the Robber-Bot is informed of the state of the world, it receives full information about the location of the cops, in addition to the amount of money currently in the banks. It inspects the world and decides where to move. If it moves onto a bank it automatically steals all the money that the bank contains. It also sometimes leaves behind evidence of its whereabouts as it leaves a location.

When the Cop-Bots are informed of the state of the world, they learn how far away the robber is (if it is nearby), every other Cop-Bot's location and mode of transportation, which bank the Robber-Bot is robbing (if any), and the money left in the banks. They can then share information with each other. Afterwards, they each propose a plan that indicates where all of the Cop-Bots should move, vote on all the plans, and then individually move. The voting is non-binding, but helps the Cop-Bots coordinate.

Cop-Bots can move either on foot or in a Robo-Cruiser: foot cops can smell the robber up to two squares away, and can move one intersection away without regard for whether a street is one-way or two way. Cop-Bots in Robo-Cruisers can travel one intersection along extra-long Robo-Cruiser-only roads or normal roads, but can only smell the robber if it is one square away, and must obey one-way streets.

The game is over when the Cop-Bots catch the Robber-Bot or when the Robber-Bot has evaded the Cop-Bots for 201 worlds (numbered 0 thru 200). At the end of the game, the Robber-Bot is rewarded for how much money it managed to get without being caught and the Cop-Bots are rewarded as a team for how quickly they caught the Robber-Bot and individually for gathering the most evidence, having the most-selected plans, and being the actual Cop-Bot to catch the robber.

The rest of this section explains these rules in more detail. Section 2.1 explains the game map, section 2.2 explains the rules that pertain to Cop-Bots and Robber-Bots, section 2.3 explains the turn structure, and section 2.4 explains the scoring rules.

2.1 The Game Map

A *game map* is a directed graph with labelled nodes and two distinguished kinds of edges. In the Cops & Robbers game, the game map is fixed and is a representation of the Hyde Park neighborhood of Chicago. Vertices are labelled and are called *intersections*; edges are called *streets*. All gameplay takes place on this map: each Cop-Bot and Robber-Bot is positioned on an intersection, all banks are located on intersections, robots can only travel along streets, and so forth.

Intersections Each intersection in the map has a unique name. An intersection may be occupied by more than one player. Players move between intersections along streets. There are two kinds of distinguished intersections: banks and police headquarters, whose roles are explained in subsequent sections.

Streets Streets go from one intersection to another, and are considered to be one way (two-way streets are represented by a pair of one-way streets going opposite directions).

There are two kinds of streets: those that any Cop-Bot may travel along and those that can only be taken by a Cop-Bot in a Robo-Cruiser. The directionality of a street is only important to car cops; Cop-Bots on foot can always travel in either direction on a street (if they can travel on the street at all). For more information about how robots travel along streets see section 2.2.

Banks There are six banks on the game map. Each bank occupies a unique intersection. Each bank has a certain amount of cash in its vault that is announced to all players every turn. When a Robber-Bot moves into an intersection occupied by a bank, he or she automatically robs the bank, steals the entire contents of the vault, and triggers an alarm that informs all cops of the robber's exact location.

Each bank starts the game with 1000 dollars in its vault. A robbed bank will receive additional funds after several turns of play according to the following rules: if the robber moves onto a bank between world n and world $n + 1$, then between worlds $n + 8$ and $n + 9$ the bank receives funds from the other banks. The fund transfer is instantaneous. The robber sees world $n + 8$ with uneven amounts in the bank and the cops see world $n + 9$ with the funds transferred. Each bank contributes $\frac{1}{6}$ of the difference between the contents of its vault and the contents of the robbed bank's value to the robbed bank, at the time the transfer occurs. This transfer occurs even if the bank had no money in it when it was robbed. If the robber is robbing a bank when the transfer occurs, the robber robs the bank *before* the transfer.

Police Headquarters Hyde Park has a local police headquarters located at one of the intersections in the game map. All Cop-Bots start the game at police headquarters. Whenever a cop is located at headquarters, it may choose to pick up a Police Robo-Cruiser from the parking lot or to exit its Robo-Cruiser and leave on foot. In tournament play, the headquarters will always be at the intersection labelled `55-and-woodlawn`.

Evidence An intersection may have one or more pieces of *evidence* on it. Evidence is represented in the game by a single number on an intersection. The number indicates which world the Robber-Bot occupied the intersection.

During game play, the intersection the Robber-Bot occupies in world 8, and any subsequent world whose number is divisible by 8, is marked with evidence. In other words, if the Robber-Bot is on intersection i in world n, $n\%8 = 0$ and $n \geq 8$, then n becomes a piece of evidence on intersection i.

When a Cop-Bot moves into an intersection that contains one or more pieces of evidence, the Cop-Bot is given all the evidence on that intersection and all evidence is removed from that intersection. If more than one

cop occupies an intersection in the same world and there is evidence on that intersection, each cop present is given a copy of the evidence.

All evidence that was placed on world n is removed on world $n + 24$ (if it had not already been collected by a Cop-Bot).

Smells If a Cop-Bot in a Robo-Cruiser is within one intersection of the Robber-Bot (according to a legal move with the cop's current mode of transportation; see section 2.2 for a description of the legal moves) it is informed that it can smell the Robber-Bot. Similarly, if a Cop-Bot on foot is within two intersections of the Robber-Bot (according to its legal moves), it is informed that it can smell the robber, and it is informed of the distance to the robber.

The smells do not take into account the fact that the Cop-Bot might change its mode of transportation. For example, if a Cop-Bot is at the police headquarters in a Robo-Cruiser, and the Robber-Bot is two intersections away, the Cop-Bot does not smell the Robber-Bot.

The smells do take into account car-only edges, however. For example, a Cop-Bot in a Robo-Cruiser at `55-and-south-hyde-park` can smell both `55-and-cottage-grove` and `55-and-cornell` (and the other intersections it could move to).

2.2 The Robots

There are two kinds of robots in the game: Cop-Bots and Robber-Bots. In each game there are exactly five Cop-Bots and exactly one Robber-Bot. Each robot is controlled by a different player.

2.2.1 Cops

Cop-Bots earn points by locating and capturing the Robber-Bot. The Robber-Bot is captured when one or more Cop-Bots occupies the same intersection as the Robber-Bot.

Cop-Bot starting position All Cop-Bots begin the game at the police headquarters. In tournament play this is always the intersection labelled `55-and-woodlawn`.

Cop visibility All cops can see the locations and transportation of all the other cops in the game.

Cop-Bots are informed of the Robber-Bot's location when the Robber-Bot is occupying an intersection that contains a bank.

Cop communication Cop-Bots can communicate with each other during designated communication rounds before they decide on moves. Communication consists of three rounds: the announcement round, the planning round, and the voting round. These rounds are explained in game terms in section 2.3.2 and the exact protocol format is given in section 3. Informally, in these rounds the Cop-Bots share knowledge, propose plans of action, and then vote to decide which plan to collectively follow.

While cops do select a plan collectively, that plan is *non-binding*. Cops may make any (legal) move they want regardless of the plan's instructions for them.

Cop movement Cops on foot may move one intersection per turn along any street that allows pedestrian access, and in either direction. Cops in cars may move one intersection per turn along any car street or pedestrian access street, but must observe the direction of one way streets. See section 2.1 for more details of the street structure.

2.2.2 Robbers

The Robber-Bot wins points by robbing banks while avoiding capture by Cop-Bots.

Robber starting position The robber begins the game at a particular, specially labelled intersection on the map. In tournament play that intersection will always be the intersection labelled `54-and-ridgewood`.

Robber visibility The robber has a police scanner and knows the exact location and mode of transportation of all cops on the map. The robber leaves evidence on certain intersections as described in section 2.1 and cannot cover its tracks.

Robber movement The robber moves on foot following the same movement rules as the Cop-Bots on foot.

2.3 Turn Structure

Game play is broken into a series of turns. Conceptually, turns occur between worlds (and are what generate new worlds).

In the following discussion it will be useful to refer to the current world. We will use the variable n for that purpose. Also, we will refer to robots "announcing" information. These announcements are made to the game manager, and are not shared with any other robot. Points at which robots are informed of other robots' announcements are explicitly stated.

2.3.1 Robber's turns

The Robber-Bot's turn happens between an even numbered world and an odd numbered world. That means that a Robber-Bot is always given an even numbered world to view. Once the robber submits its move, a number of changes happen to the world to produce the next world.

Evidence placement and collection If this world number is one where the robber would leave evidence, as described in section 2.1, the evidence is placed before generating world $n + 1$. The evidence, tho, is labelled n, since the robber was on the intersection where the evidence was placed in world n. Similarly, any evidence that documents the robber location on the board in world $n - 24$ is removed from the board.

Larceny stage If the Robber-Bot is standing on a bank, it empties the bank's vault and collects the full value of the vault's contents. In other words, the amount the robber is considered to have robbed is increased by the current amount of money in the bank, the bank is considered to have been robbed on turn n and its current money is set to zero.

Bank transfer stage In this stage money is transferred between banks. Any bank that was robbed on world $n - 8$ (that is, the robber was standing on the bank in world $n - 8$) now has money transferred to it using the process described in section 2.1. If a bank transfer would happen at the same time as the robber robs a bank, the robbery happens first and the transfer happens second.

2.3.2 Cop turns

The Cop-Bot's turn happens between an odd numbered world and an even numbered world. That means that a Cop-Bot is always given an odd numbered world to view and discuss with its fellow Cop-Bots. Once the discussion is complete, Cop-Bots submit their moves and the world is changed to reflect the moves (becoming an even-numbered world).

World updates stage At the start of this stage, each Cop-Bot is told about the world, including any evidence it has collected, if it can smell the Robber-Bot (and at what distance), the positions of the other Cop-Bots, and the amount of money in each bank. In addition, if the Robber-Bot is standing on a bank, the world shows the Robber-Bot's position.

Communication stage In this stage Cop-Bots can share information. Cop-Bots announce information about their observations and can also make hypotheses about the robber's location. The cop communication stage is over after all cops have announced their observations and hypotheses.

Planning stage At the start of the planning stage, each Cop-Bot receives any announcements sent to them during the communication stage. Each Cop-Bot then formulates a plan and announces it. The planning stage is over after all Cop-Bots have submitted a plan.

Voting stage At the start of the voting stage each cop receives a copy of all plans announced in the previous stage. Each Cop-Bot announces a ranking of the plans it received, from best to worst.

Once the cops announce their votes, the votes are tallied. The algorithm used to select the winning plan is as follows:

1. Initialize `ballots` to the list of all ballots where each ballot is ordered by decreasing preference (most desirable candidate first). Go to step 2.
2. Initialize `running candidates` to the set of all candidates. Go to step 3.
3. If `running candidates` is empty, or all of the ballots are empty, voting is over without a winner. Otherwise, go to step 4.
4. If `running candidates` is a singleton set, return its only element as the winner of the election. Otherwise, go to step 5.
5. For each candidate `x` in `running candidates`, determine the number `C(x)` of times it appears in first place on a ballot in `ballots`. Go to step 6.
6. Let `top` be the subset of `running candidates` such that `x` is in `top` if and only if `C(x)` is maximal. Go to step 7.

7. If `top` is equal to `running candidates`, then discard the top votes on each ballot, i.e., replace each ballot `b` in `ballots` with the tail of `b`. Go to step 8.

8. Set `running candidates` to `top`, i.e., keep candidates in the running only if they are top-ranked. Go to step 9.

9. For each ballot `b` in `ballots`, delete from `b` all candidates that are not in `running candidates`. If a ballot is now empty, then discard it. Then go to step 3.

Cop Movement After the votes are tallied, the Cop-Bots are informed of the winning plan. Note that the winning plan is *not binding*. Cop-Bots are free to disobey it.

Each cop then announces where to move and whether it is travelling by foot or in a Robo-Cruiser. It is only legal to switch modes of transportation if the Cop-Bot is on the police headquarters in the current world.

The moves are then applied to the world to produce a new world (for the robber).

2.4 Scoring

Each robot is awarded points from some *base* for overall performance; Cop-Bots also may get some additional *bonus* points added to that base for individual performance. Scoring is divided into two cases: the case in which the Cop-Bots capture the Robber-Bot, and the case in which the Robber-Bot gets away.

If the Robber-Bot is captured, the Robber-Bot receives 0 points and each Cop-Bot gets one fifth of the total amount of money left in banks as a base. If the robber escapes, the robber receives 1 point for each dollar stolen from the banks and the cops all receive 0 points as a base. In either case, cops can receive additional individual bonus points:

- The cop that finds the most evidence get a bonus of 60 points.
- The cop whose plans are selected most often gets a bonus of 60 points.
- The cop who captures the robber, if any, gets a bonus of 60 points.

The same Cop-Bot can win more than one bonus, in which case the bonus points are cumulative. For all bonuses, if there is a tie (e.g., if two Cop-Bots capture the Robber-Bot simultaneously) all cops involved in the tie split the points evenly. For the evidence bonus, if multiple Cop-Bots discover a piece of evidence on the same intersection simultaneously, they are both considered to have collected that evidence.

3 The Communications Protocol

It is your task to create a Bot-Brain that will control the actions of the Cop-Bots and Robber-Bots. However, in order to maximize available computing power and to keep robot production costs down, you will not be able to control them directly. All instructions must be sent through a Manager, a machine that relays messages between your Bot-Brain to actual Cop-Bots or Robber-Bots in the field.

The manager uses a client-server architecture. Your Bot-Brain will communicate only with the Manager. Communication between Bot-Brains and the Manager occurs over standard input and standard output. Specifically, when the Manager sends a message to a Bot-Brain, the message is written into the Bot-Brain's standard input (STDIN). Similarly, when a Bot-Brain wants to send a message to the Manager, it should print to standard output (STDOUT). Each character is sent as a single ASCII octet.

3.1 The State Machine

The rounds described in section 2.3 correspond to states of the Bot-Brains communication protocols in figure 1 and figure 2. In these diagrams, $C \rightarrow S$ means that the message is sent from the player to the game manager. $S \rightarrow C$ means that the game manager sends this message to the player. The labels on the edges are the names of the messages (defined in section 3.2.2), and the names in the boxes are the names of the states.
Robber states:

- initial: The start state. The game has just started and the game manager is waiting for all players to register.
- waiting_for_world: The player has registered and is waiting for the game manager to send an overview of the world.
- waiting_for_turn: The player is waiting to see the next world.
- moving: After receiving the world update, the robber sends a move in to the server. After this state, we move to the waiting_for_turn state, and this process repeats until the game is over.
- game_over: The game has finished.

Cop states:

- The initial, waiting_for_world, waiting_for_turn, and game_over states are just like the robber's states.
- inform: Once the manager sends the world update, cops share information about the world with other cops.
- waiting_for_inform: Once a cop has submitted messages to the other cops, it waits until it receives the messages from the others.
- planning: Once the cops have shared information with each other, they formulate suggested plans and transmit them to the server.
- waiting_for_plans: After your Cop-Brain has submitted its plan, it waits to hear others' plans.
- voting: Once all of the cops submit a plan the plans are sent to all of the cops, and the cops evaluate them. The cops then cast a ballot indicating their preference for the plans.
- waiting_for_results: Your Cop-Brain has voted, and is waiting for the results from the server.
- moving: The game manager sends the winning plan to all of the cops, the cops respond with their moves, and the cycle repeats (until the game is over).

3.2 Message Grammar

The message protocol is line-based. Each line consists of a series of tokens, separated by a tab or a space, and each message is a series of lines. Each separator must be exactly one tab character or exactly one space character and lines may not begin with spaces or tabs. Newlines may be either just a single newline character (ASCII 10), or a return (ASCII 13) followed by a newline.

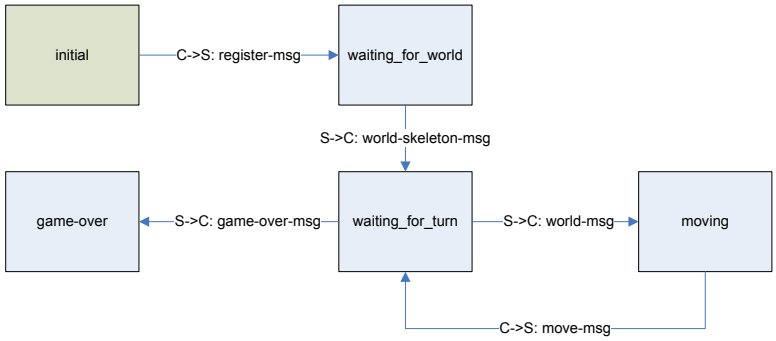

Figure 1: Robber-Bot state machine

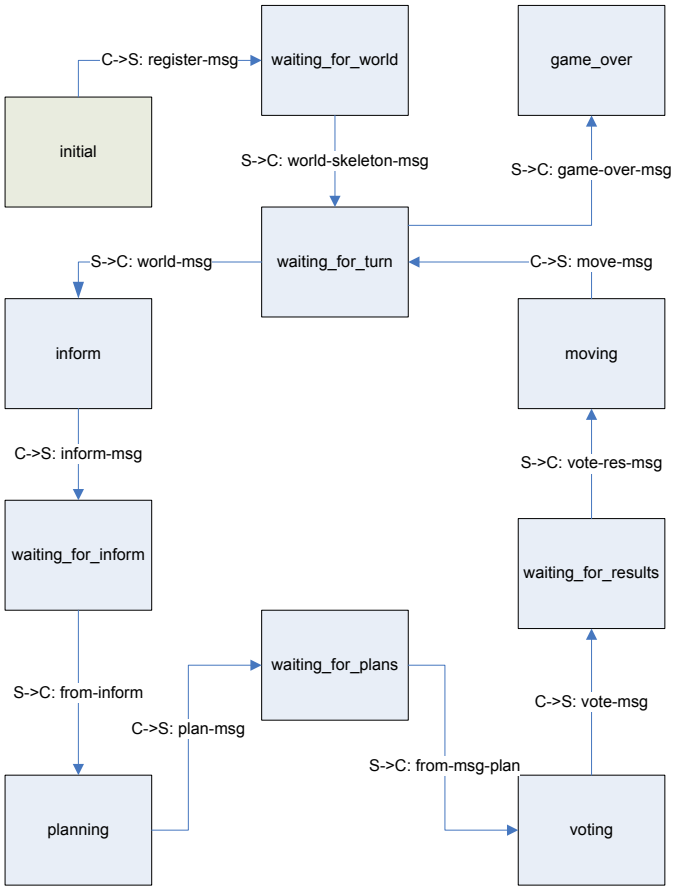

Figure 2: Cop-Bot state machine

3.2.1 Tokens

This subsection describes the tokens that build up lines. Each of the categories below is specified with a regular expression, using square brackets for character ranges, + to indicate at least one repetition, | for alternatives, and otherwise characters are literals. The characters `\r` and `\n` are used for return (ASCII 13) and newline (ASCII 10), respectively. In addition to the constraints below, no token may be more than 100 characters.

name	:=	`[-a-zA-Z0-9_#()]+`	names include hyphen, letters, numbers underscore, hash, and parens
number	:=	`[0-9]+`	sequence of digits
negnumber	:=	`-[0-9]+`	a negative sign followed by a sequence of digits
bot	:=	*name*	
loc	:=	*name*	
bank-value	:=	*number*	must be between 0 and 1000 inclusive
loot	:=	*number*	must be between 0 and 6000 inclusive
world	:=	*number*	must be between 0 and 200 inclusive
distance	:=	*number*	must be between 0 and the number of nodes inclusive
certainty	:=	*number* | *negnumber*	must be between -100 and 100 inclusive
coordinate	:=	*number*	must be betwen 0 and 1023 inclusive
eol	:=	`\r\n` | `\n`	
ptype	:=	`cop-foot` | `cop-car` | `robber`	
edge-type	:=	`car` | `foot`	
node-tag	:=	`hq` | `bank` | `robber-start` | `ordinary`	

3.2.2 Messages

Each of the grammars below describes the format of the messages from the edges in the state machines (figures 1 and 2).

Text in `typewriter` font is literal data. References to other parts of the spec are typeset in *italics*. Lines that are surrounded in parenthesis with a trailing asterisk (*) may appear any number of times between 0 and 1000 (with the exception of the last two messages in this section; see below).

Register: ($C \rightarrow S$) Bots use this message to register with the server.

register-msg := `reg:` *bot ptype eol*

World Skeleton: ($S \rightarrow C$) This message indicates initial information that the player receives.

world-skeleton-msg := `wsk\` *eol*
`name:` *bot eol*
`robber:` *bot eol*
`cop:` *bot eol*
`cop:` *bot eol*
`cop:` *bot eol*
`cop:` *bot eol*
`cop:` *bot eol*
`nod\` *eol*
(`nod:` *loc node-tag coordinate coordinate eol*)*
`nod/` *eol*
`edg\` *eol*
(`edg:` *loc loc edge-type eol*)*
`edg/` *eol*
`wsk/` *eol*

The player's name is repeated back to the player, in case the server changed it. The server only changes the name when it is the same as some other player's name. Following the player's name, the server reports the names of all of the players in the game.

In a node line, the tag `bank` indicates that the given node is a bank, `hq` indicates that the given node is a headquarters, `robber-start` indicates that the given node is the node on which the robber starts the game, and `ordinary` indicates that the node is not one of those three types. There is always exactly one headquarters and exactly one robber start position on any world-skeleton message.

The coordinates associated with the nodes indicate relative positions of the nodes (in the usual computer-science coordinate system: (0,0) is at the top left, and bigger coordinates move to the right and down, respectively) to show where the nodes match up (approximately) to a real map of Hyde Park. The coordinates do not play a role in game play, but may help you visualize the game.

The `edg:` lines show the source edge first and the target edge second.

World ($S \to C$) This message contains all the data that is variable in the world (values at banks, whether you found evidence or not, location/types of players). The `rbd:` line contains the amount of loot the robber currently has. The `bv` section describes the amount of money in the banks. The `ev` section describes the evidence on the current node; it indicates the location and turn where the robber was (there are never any `ev:` lines for the robber). The `smell:` line indicates if cops can sense the robber nearby. If the cops cannot sense the robber, the distance will be zero (the distance is always zero for the robber). Finally, the `pl` section describes the visible players, their types, and their locations.

world-msg := `wor\` *eol*
`wor:` *world eol*
`rbd:` *loot eol*
`bv\` *eol*
(`bv:` *loc bank-value eol*)*
`bv/` *eol*
`ev\` *eol*
(`ev:` *loc world eol*)*
`ev/` *eol*
`smell:` *distance eol*
`pl\` *eol*
(`pl:` *bot loc ptype eol*)*
`pl/` *eol*
`wor/` *eol*

Move ($C \rightarrow S$) The players communicate their moves with a `mov:` line. The *ptype* must match the player's current *ptype* unless the player is a cop and is located at the cop headquarters in the current world. In that case, a cop may switch its type from `cop-car` to `cop-foot`.

move-msg := `mov:` *loc ptype eol*

Game Over ($S \rightarrow C$) This means the game has finished.

game-over-msg := `game-over` *eol*

Inform ($C \rightarrow S$) Cop-Brains share information about player locations with this message. Each line indicates where and when a player was (or is or will be), with a certainty. Positive certainties indicate how sure the reporter is that the information is true. Negative certainties should be interpreted as how certain the reporter is that the information is false.

inform-msg := `inf\` *eol*
(`inf:` *bot loc ptype world certainty eol*)*
`inf/` *eol*

Suggest Plan ($C \rightarrow S$) The plan message communicates suggested moves for each Cop-Bot. A plan may contain any number of moves for any robot.

plan-msg := `plan\` *eol*
(`plan:` *bot loc ptype world eol*)*
`plan/` *eol*

Voting ($C \rightarrow S$) After each player has submitted a plan, they are sent to the cops using a from-msg. The voting is done by ranking, so listing a player's name first means that you think they have submitted the best plan and listing a player last means you think they submitted the worst plan. A legal ballot must contain each Cop-Bot's name, exactly once.

vote-msg := `vote\` *eol*
(`vote:` *bot eol*)*
`vote/` *eol*

Vote Tally ($S \rightarrow C$) After all players have voted, the server computes which is the elected plan and reports the winner, if any.

vote-res-msg := `winner:` *bot eol*
| `nowinner:` *eol*

From: Messages from other cops ($S \rightarrow C$) When a player sends an inform-msg or a plan-msg to the server, the server makes sure it is grammatically correct, and then forwards it to the other clients.

These message are unlike other messages since they contain repeated elements that are longer than a single line. Each of the repeated messages is prefixed by a `from:` token, indicating who originally sent the message, and then the message they sent (with the spaces and newlines normalized).

Unlike the other repeated elements in the protocol spec, the repeated elements below may contain more than 1000 lines, but each repeated element from the *inform-msg* or *plan-msg* non-terminals can only appear at most 1000 times. Overall, this means that entire message may be as long as 5000 lines roughly (plus a few more structuring lines), since the server only sends out one encapsulated message per Cop-Bot.

from-msg-inform := `from\` *eol*
(`from:` *bot eol*
inform-msg)*
`from/` *eol*

from-msg-plan := `from\` *eol*
(`from:` *bot eol*
plan-msg)*
`from/` *eol*

3.3 Time limits and protocol violations

Whenever a player is required to submit information to the server, that player has 5 seconds to submit it from the time the server begins sending the most recent communication. For example, after the server starts sending the first byte of a world-update message, a Cop-Bot has 5 seconds to finish sending an inform message.

The penalty for not submitting a message within the time limit or for submitting a malformed or illegal message is disqualification from the tournament; see section 4.

4 Tournament Structure

Your Cop-Bot and Robber-Bot progress through the tournament together. The tournament takes place in two rounds: a regular season followed by the playoffs.

In the tournament, contestants will be grouped into pods. Each pod will have six pairs of Cop-Bot and Robber-Bots. The pod plays six games and in each game five contestant's Cop-Bots play against the sixth's contestant's Robber-Bot. Each game uses a different contestant's Robber-Bot program will be used Each game is scored according to the rules explained in section 2.4 and the score for each contestant in the pod is the sum of the scores of the games.

In the regular season pods, each contestant's Cop-Bot will play with five sets of the Judges' entries. The Judges' entries will consists of a mediocre Cop-Bot and a mediocre Robber-Bot. They Cop-Bots will be slightly special, because they will act like McGruffs when your entry is playing a Cop. For the purposes of the regular season, if a McGruff is fed back information and makes an illegal move, it will stand still. When

your entry is playing the robber, cops will be the Judges' own mediocre Cop-Bot. The scores are summed as usual. Any submission that does not win the pod during the regular season is eliminated.

The playoffs will be a pod-based single-elimination tournament. The three contestants with the top three scores in each pod proceed to the next round. When only six contestants remain, they will form a final pod. The top three contestants in that pod are the first, second, and third place winners of the tournament. Ties are broken based on performance in prior rounds of the tournament. Ties in the first round will be broken by performance in the regular season. Ties beyond that are unlikely (but if they happen, we will find a fair way to break them).

All programs will be run on a AMD Athlon 2700+ computer with 512MB RAM. They will run as users that have no network access, read-only access to the directory in which their program resides, read-only access to standard Unix paths, and a 100 megabyte quota in `/tmp`. All files in `/tmp` will be deleted between games. During each game, each cop and robber program will run on its own machine. The machines running non-LiveCD-based submissions will have slightly different specifications, but will achieve the same effect: no permanent, cross-game storage and no network access.

Any contestant whose Cop-Bot or Robber-Bot is disqualified in any game (e.g., by sending an illegal message; see section 3.3) is disqualified from the tournament. The particular pod where a disqualification occurs is with a new sixth member. (The aborted pod's partial results are ignored). Otherwise, the contestant's score for the pod is the sum of its scores for each game in the pod.

In the event that the contestants cannot be grouped into groups of six evenly, an unspecified process takes place to form groups of six. No games ever take place in which there are not five cop programs competing against one robber program.

5 Version History

7 Improved timeout description in section 3.3.

6 Fixed bug in description of Bank transfers in 2.1 (transfer timing in 2.3.1 is correct). Fixed typo in first para of 3.2.

5 Added `edg:` directionality explanation. Improved description of tournament structure.

4 Fixed bug in voting algorithm spec. Improved smell description.

3 Whitespace conventions clarified in 3.2.

2 Typo fixed in 2.3.2.

1 Initial release.