Specification of the Go Text Protocol, version 2, draft 2

Gunnar Farnebäck

October, 2002

`http://www.lysator.liu.se/~gunnar/gtp`

This is a draft only.

Contents

1 Introduction

This document gives a specification of the Go Text Protocol (GTP), version 2.

1.1 Purpose of the Protocol

The intention of GTP is to provide a flexible and easy to implement communication protocol for go programs. The main purpose is to allow two programs to play each other but it is also useful for regression testing and communication with a GUI or a go server. Most use cases require an external support program, but this can be shared between all programs with GTP support.

1.2 History

The Go Text Protocol was developed within the GNU Go project, initially to create a framework for automated regression testing and to simplify connecting the program to go servers. The first appearance of the protocol was on May 18, 2000, in GNU Go development version 2.7.95. The first stable release of GNU Go with GTP support was GNU Go 3.0.0, released August 24, 2001, which is the reference implementation for version 1 of the protocol. There is no good specification of GTP version 1, however, and this document is intended to provide one for version 2.

1.3 Communication Model

The protocol is asymmetric and involves two parties, which we call controller and engine. The controller is typically some kind of arbiter or relay and the engine is typically a go playing program. All communication is initiated by the controller in form of commands, to which the engine responds.

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 latter can also be established over an error prone modem connection by using PPP (Point to Point Protocol) as a transport layer.

1.4 Typical Use Cases

1. Regression testing.
   controller (regression script) — engine
   The controller sets up a board position and asks the engine to e.g. generate a move.

2. Human vs program.
   controller (GUI) — engine
   The controller relays moves between the human and the engine and asks the engine to generate moves.

3. Program vs program with arbiter.
   engine 1 — controller (arbiter) — 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. There is no direct communication between the two engines. The controller dictates board size, komi, etc.

4. Program vs program without arbiter.
   The same as above except that engine 1 includes the controller functionality and the first GTP link is shortcut.

5. Connection between go server and program.
   go server — controller (relay) — engine
   The controller talks to a go server using whatever protocol is needed and listens for match requests. When one arrives it accepts it, starts the go engine and issues GTP commands to set up board size, komi, etc. and if a game is restarted it also sets up the position. Then it relays moves between the server and the engine and asks the engine to generate new moves when it is in turn.

1.5 Reference Implementation

The reference implementation for GTP version 2 is GNU Go version 3.4. In cases of incompleteness or unclarity in this specification, the reference implementation decides the correct behaviour. Notice, however, that any command available in GNU Go 3.4, but not included in this specification (full list in section 6), is to be considered a private extension (see section 2.13).

Temporary comment: GNU Go 3.4 is currently under development and GTP version 2 has not yet been implemented in the development versions.

2 Protocol Basics

2.1 Character Set

All messages exchanged in this protocol are to be considered as 8-bit character sequences. Only characters in the US-ASCII character set (ANSI X3.4-1986) are used for standardized commands and responses. Other characters may be used in comments (section 2.9) and private extensions (section 2.13) but there is no preferred character set specified for those.

2.2 Control Characters

Character values 0–31 and 127 are control characters in ASCII. The following control characters have a specific meaning in the protocol:

`HT` (dec 9)	Horizontal Tab
`CR` (dec 13)	Carriage Return
`LF` (dec 10)	Line Feed

All other control characters must be discarded on input and should not be used on output.

2.3 Whitespace

The following ASCII characters can be used to indicate whitespace in the protocol:

`SPACE` (dec 32)	Space
`HT` (dec 9)	Horizontal Tab

In the rest of the specification we use 'space' to denote a whitespace character. On input this may be either a SPACE or a HT. On output only a SPACE should be used.

2.4 Newline Convention

A newline is indicated by a single LF character. Any occurence of a CR character must be discarded on input, both by the engine and the controller. On output either LF or some combination of CR and LF can be used. In syntax descriptions we use `\n` to indicate a newline.

2.5 Command Structure

A command is exactly one line long, with the syntax

```
[id] command_name [arguments]
```

Here `id` is an optional identity number and `command_name` a string. The rest of the line (up to the first newline) gives the arguments of the command.

2.6 Response Structure

If successful, the engine returns a response of the form

```
=[id] result
```

Here '=' indicates success, `id` is the identity number given in the command, and `result` is a piece of text ending with two consecutive newlines.

2.7 Error Messages

If unsuccessful, the engine returns a response of the form

```
?[id] error_message
```

Here '?' indicates failure, `id` is the identity number given in the command, and `error_message` gives an explanation for the failure, also ending with two consecutive newlines.

2.8 Timing

There are no synchronization requirements between the controller and the engine. The controller may send commands at any time, regardless of whether it has obtained responses for previous commands. The engine may send responses whenever they are ready. It must, however, respond to the commands in the same order as they come in. The engine is allowed to make pauses while sending a response.

2.9 Comments

Comments can be included in the command stream. All text between a hash sign (#) and the following newline is considered as comments and should be discarded on input.

2.10 Empty lines

Empty lines and lines with only whitespace sent by the controller must be ignored by the engine. No response must be generated. Empty lines and lines with only whitespace sent by the engine and occuring outside a response must be ignored by the controller. Notice that pure comment lines will appear as empty lines after the comment has been discarded.

2.11 Board Coordinates

Board intersections, in this document called vertices, are encoded by a letter plus a number. On a 19x19 board the letters go from A to T, excluding I, from the left to the right. The numbers go from 1 to 19, from the bottom to the top. Thus the lower left corner is called A1, the lower right corner T1, the upper left corner A19, and the upper right corner T19. Smaller boards use the obvious subset of these coordinates. Larger boards, up to 25x25, are handled by extending the letters with U to Z as needed. Boards larger than 25x25 are not supported by the protocol.

2.12 Protocol Subsets

An engine does not have to implement all commands listed in this specification. In general, for an engine to be used with some specific controller, it is only required that the engine understands exactly the commands needed by that controller. To simplify this matching of capabilities, there are two predefined protocol subsets called the tournament and the regression subsets. There is also a small set of commands required for all GTP supporting engines.

2.13 Private Extensions

The protocol is trivial to extend with new commands. Obviously there is a risk for conflicts if multiple engines make incompatible private extensions of the protocol or if an engine makes a private extension which turns out to be incompatible with a future extension of the standard protocol.

In order to avoid this problem, standard commands do not include the dash (`-`) character. Private extensions are recommended to be of the form `XXX-YYYYY`, where `XXX` is a prefix which is sufficiently unique for the engine or controller in question, and `YYYYY` describes the command. E.g. a private variant of the `genmove` command used by GNU Go could be called `gg-genmove`.

Engines are allowed to use private extensions without a dash in the name, but then they do it at their own risk and must be prepared to change if the name later becomes used for a standard command.

2.14 Panic Situations

If an engine for some reason, e.g. an internal error, finds itself in a position where it cannot meaningfully continue the session, the correct action is to just close the connection. This is also what typically will happen if the program should happen to encounter an uncontrolled crash.

3 Protocol Details

3.1 Preprocessing

When a command string arrives to an engine, it is expected to perform the following four operations before any further parsing takes place:

1. Remove all occurences of CR and other control characters except for HT and LF.
2. For each line with a hash sign (#), remove all text following and including this character.
3. Convert all occurences of HT to SPACE.
4. Discard any empty or white-space only lines.

When a response arrives to a controller, it is expected only to do steps 1 and 3 above.

Naturally an implementation does not have to actually do this preprocessing as a separate step but may interleave it with other parts of the parsing. For purposes of the following specifications, though, the preprocessing is supposed to have been carried out in full.

3.2 Syntactic Entities

3.2.1 Simple Entities

- **int**
  An `int` is an unsigned integer in the interval $0 <= x <= 2^{31} - 1$.
- **float**
  A `float` is a floating point number representable by a 32 bit IEEE 754 float.
- **string**
  A `string` is a sequence of printable, non-whitespace characters. Strings are case sensitive.
- **vertex**
  A `vertex` is a board coordinate consisting of one letter and one number, as defined in section 2.11, or the string “pass”. Vertices are not case sensitive. Examples: “B13”, “j11”.
- **color**
  A `color` is one of the strings “white” or “w” to denote white, or “black” or “b” to denote black. Colors are not case sensitive.
- **move**
  A `move` is the combination of one `color` and one `vertex`, separated by space. Moves are not case sensitive. Examples: “white h10”, “B F5”, “w pass”.
- **boolean**
  A `boolean` is one of the strings “false” and “true”.

3.2.2 Compound Entities

- **Collection**
  An {x y} is an x followed by a y, separated by a space. x and y may be any combination of simple entities. The construction can be generalized to any fixed number of entities.

- **List**
  An x* is a space separated list of entities of type x, where x may be any of the entities specified so far. The list can have an arbitrary number of elements and goes on until an LF is encountered.

- **Alternatives**
  An x|y is either an x or a y.

- **Multiline list**
  An x& is an LF separated list of entities of type x, where x may be any of the entities specified so far. The multiline list can have an arbitrary number of lines and goes on until two consecutive LFs are encountered.

3.3 Commands

A command has one of the syntaxes

```
id command_name arguments\n
id command_name\n
command_name arguments\n
command_name\n
```

- `id` is an optional `int`.

- `command_name` is a `string`.

- `arguments` is a space separated list of some collection of entities, the composition of which varies with the command If `arguments` is missing it counts as empty.

3.4 Success Responses

A successful response has one of the syntaxes

```
=id response\n\n
=id\n\n
= response\n\n
=\n\n
```

- `id` is an optional `int` and must be the same number as in the corresponding command. It may be omitted if and only if it was omitted in the command.

- `response` is some collection of entities, separated by space or a single LF, the composition of which varies with the command. The response may be empty.

3.5 Failure Responses

An unsuccessful response has one of the syntaxes

```
?id error_message\n\n
? error_message\n\n
```

- `id` is an optional `int` and must be the same number as in the corresponding command. It may be omitted if and only if it was omitted in the command.
- `error_message` is a `string*&`.

3.6 Standard Error Messages

If the engine receives an unknown or unimplemented command, use the error message "unknown command". Some commands fail in certain cases with standardized error messages. Those are listed in the command descriptions in section 6.3. For other failures the engine can freely choose error message.

4 Important Concepts

4.1 Handicap Placement

The protocol supports both fixed placement of handicap stones and free placement. The handicap stones are always black.

4.1.1 Fixed Handicap Placement

With fixed placement the handicap stones are set in predetermined positions. The maximum number of fixed handicap stones varies with the board size but is never larger than 9. On a 19x19 board, the positions for the handicap stones are given by this table:

Handicap	Vertices
2	D4 Q16
3	D4 Q16 D16
4	D4 Q16 D16 Q4
5	D4 Q16 D16 Q4 K10
6	D4 Q16 D16 Q4 D10 Q10
7	D4 Q16 D16 Q4 D10 Q10 K10
8	D4 Q16 D16 Q4 D10 Q10 K4 K16
9	D4 Q16 D16 Q4 D10 Q10 K4 K16 K10

The placement of handicap stones on other board sizes mirrors that of 19x19 with stones at a specific distance from the edges and on the middle lines of the board, with the following caveats:

- For boards smaller than 13x13, the edge stones are placed on the third line instead of on the fourth line.
- For boards of even size there is no middle line and therefore no handicaps larger than 4.
- Boards of size 7x7 have at most 4 handicap stones.
- No handicap for boards smaller than 7x7.

More explicitly we obtain the following table:

board size	max handicap	edge distance
25	9	4
24	4	4
23	9	4
22	4	4
21	9	4
20	4	4
19	9	4
18	4	4
17	9	4
16	4	4
15	9	4
14	4	4
13	9	4
12	4	3
11	9	3
10	4	3
9	9	3
8	4	3
7	4	3
6	-	-
5	-	-
4	-	-
3	-	-
2	-	-

4.1.2 Free Handicap Placement

With free placement the handicap stones are set as chosen by the controller or by one of the engines (for normal tournament use the engine playing the black stones would make the choice). The smallest number of handicap stones is 2. The highest number is one less the number of vertices on the board. However, when the number of handicap stones becomes very high there is no benefit in additional stones. Therefore, when asked to choose handicap placement, an engine is allowed to return a smaller number of stones than requested. This provision should only be used if the requested number of stones is so high that a smaller number of stones is believed to guarantee that the engine cannot possibly lose against any opponent.

4.2 Time Handling

The protocol has support for Canadian byo yomi, including absolute time (no byo yomi) as a special case. Canadian byo yomi is characterized by the three parameters

- Main time m,
- Byo yomi time b,
- Byo yomi stones s.

The semantics is that the clock is first set to m. The engine has no requirements on the number of stones while this time is running. When it is up, the clock is reset to b and the engine has to play s stones before this time is up. When s stones have been played, the clock is reset to b, regardless of remaining time. Then the engine has to play another s stones before the time is up. This procedure repeats until the game is over. If an engine fails to play s stones before its byo yomi time is up, it loses on time.

Setting $m = 0$ means that the engine immediately starts in byo yomi. Setting $b = 0$ means that if the main time is up before the game is over, the engine loses on time. Setting $b > 0$ and $s = 0$ means no time limits.

4.3 Scoring

Depending on the exact choice of rules (see also section 8.3), scoring a finished game may be more or less complex. With a few exceptions it is critical to determine which stones are dead and which are alive. Sometimes it is also necessary to distinguish between life in seki and independent life.

This protocol provides two commands to query the engines about score and group status. They are both valid only when the game is finished.

The first command, `final_score`, asks for the engine's opinion about the score. The result is returned as a string of the form `W+2.5` if white wins, `B+31` if black wins, and just `0` if the game ends in a draw. The number in the result is of course the difference between the number of points for each player, including komi.

The second command, `final_status_list`, is used to query an engine about the status of the stones. This command takes a string argument which may be one of `alive`, `seki`, and `dead`. The result is reported by listing all stones having the requested status. The list is organized with one string per line. If an engine cannot distinguish between life in seki and independent life, all those stones should be reported as alive.

The protocol does not include any support for resolving disagreement about status or score.

5 Internal State

5.1 State Variables

An engine is expected to keep track of the following state information:

- board size
- board configuration
- number of captured stones of either color
- move history
- komi
- time settings

5.2 Default State

There is no default state for any state variable. When first started, the engine may set these as it likes. A controller which has some specific opinion about these values must set them explicitly with the appropriate commands, including clearing the board.

5.3 State Maintenance

The state is changed by certain commands, as specified in their description in section 6. State which is not explicitly modified must remain unchanged. A failed command must never change any state.

6 Commands

6.1 Required Commands

All implementations are required to support the following commands:

```
protocol_version
name
version
known_command
list_commands
quit
boardsize
clear_board
komi
play
genmove
```

6.2 Protocol Subsets

6.2.1 Tournament

The tournament subset adds the commands:

```
fixed_handicap
place_free_handicap
set_free_handicap
```

6.2.2 Regression

The regression subset adds the commands:

```
loadsgf
reg_genmove
```

6.3 List of All Commands

6.3.1 Adminstrative Commands

- **protocol_version**

arguments	none
effects	none
output	`version_number` `int version_number` - Version of the GTP Protocol
fails	never
comments	For this specification 2.

- **name**

arguments	none
effects	none
output	`name` `string* name` - Name of the engine
fails	never
comments	E.g. “GNU Go”, “GoLois”, “Many Faces of Go”. The name does not include any version information, which is provided by the `version` command.

- **version**

arguments	none
effects	none
output	`version` `string* version` - Version of the engine
fails	never
comments	E.g. “3.1.33”, “10.5”. Engines without a sense of version number should return the empty string.

- **known_command**

arguments	`command_name` `string command_name` - Name of a command
effects	none
output	`known` `boolean known` - “true” if the command is known by the engine, “false” otherwise
fails	never
comments	The protocol makes no distinction between unknown commands and known but unimplemented ones. Do not declare a command as known if it is known not to work.

- **list_commands**

arguments	none
effects	none
output	`commands` `string& commands` - List of commands, one per row
fails	never
comments	Include all known commands, including required ones and private extensions.

- **quit**

arguments	none
effects	The session is terminated and the connection is closed.
output	none
fails	never
comments	The full response of this command must be sent before the engine closes the connection. The controller must receive the response before the connection is closed on its side.

6.3.2 Setup Commands

- **boardsize**

  arguments `size`
  `int size` - New size of the board.

  effects The board size is changed. The board configuration, number of captured stones, and move history become arbitrary.

  output none

  fails Syntax error. If the engine cannot handle the new size, fails with the error message "unacceptable size".

  comments In GTP version 1 this command also did the work of `clear_board`. This may or may not be true for implementations of GTP version 2. Thus the controller must call `clear_board` explicitly. Even if the new board size is the same as the old one, the board configuration becomes arbitrary.

- **clear_board**

  arguments none

  effects The board is cleared, the number of captured stones is reset to zero for both colors and the move history is reset to empty.

  output none

  fails never

  comments

- **komi**

  arguments `new_komi`
  `float new_komi` - New value of komi.

  effects Komi is changed.

  output none

  fails syntax error

  comments The engine must accept the komi even if it should be ridiculous.

- **fixed_handicap**

  arguments `number_of_stones`
  `int number_of_stones` - Number of handicap stones.

  effects Handicap stones are placed on the board according to the specification in section 4.1.1.

  output `vertices`
  `vertex* vertices` - A list of the vertices where handicap stones have been placed.

  fails syntax error, invalid number of stones, board not empty

  comments This command is only valid if the board is empty. See section 4.1.1 for valid number of handicap stones. The handicap stones are *not* included in the move history.

- **place_free_handicap**

  arguments `number_of_stones`
  `int number_of_stones` - Number of handicap stones.

  effects Handicap stones are placed on the board on the vertices the engine prefers. See also section 4.1.2.

  output `vertices`
  `vertex* vertices` - A list of the vertices where handicap stones have been placed.

  fails syntax error, invalid number of stones, board not empty, bad vertex list

  comments This command is only valid if the board is empty. The engine may place fewer than the requested number of stones on the board under certain circumstances, as discussed in section 4.1.2. The controller can check this by counting the number of vertices in the response. The handicap stones are *not* included in the move history. Vertices must not be repeated or include "pass".

- **set_free_handicap**

  arguments `vertices`
  `vertex* vertices -` A list of vertices where handicap stones should be placed on the board.

  effects Handicap stones are placed on the vertices as requested.

  output none

  fails syntax error, board not empty, bad vertex list

  comments This command is only valid if the board is empty. The list must have at least two elements and no more than the number of board vertices minus one. The engine must accept the handicap placement. The handicap stones are *not* included in the move history. Vertices must not be repeated or include "pass".

6.3.3 Core Play Commands

- **play**

  arguments `move`
  `move move` - Color and vertex of the move

  effects A stone of the requested color is played at the requested vertex. The number of captured stones is updated if needed and the move is added to the move history.

  output none

  fails syntax error, illegal move. In the latter case, fails with the error message "illegal move".

  comments Consecutive moves of the same color are not considered illegal from the protocol point of view.

- **genmove**

  arguments `color`
  `color color` - Color for which to generate a move.

  effects A stone of the requested color is played where the engine chooses. The number of captured stones is updated if needed and the move is added to the move history.

  output `vertex`
  `vertex|string vertex` - Vertex where the move was played or the string "resign".

  fails never

  comments Notice that "pass" is a valid vertex and should be returned if the engine wants to pass. Use "resign" if you want to give up the game. The controller is allowed to use this command for either color, regardless who played the last move.

- **undo**

  arguments none

  effects The board configuration and the number of captured stones are reset to the state before the last move. The last move is removed from the move history.

  output none

  fails If the engine is unable to take back the last move, fails with the error message "cannot undo".

  comments If you want to take back multiple moves, use this command multiple times. The engine may fail to undo if the move history is empty or if the engine only maintains a partial move history, which has been exhausted by previous undos. It is never possible to undo handicap placements. Use clear_board if you want to start over. An engine which never is able to undo should not include this command among its known commands.

6.3.4 Tournament Commands

- **time_settings**

  arguments `main_time byo_yomi_time byo_yomi_stones`
  `int main_time` - Main time measured in seconds.
  `int byo_yomi_time` - Byo yomi time measured in seconds.
  `int byo_yomi_stones` - Number of stones per byo yomi period.

  effects The time settings are changed.

  output none

  fails syntax error

  comments The interpretation of the parameters is discussed in section 4.2. The engine must accept the requested values. This command gives no provision for negotiation of the time settings.

- **time_left**

arguments	`color time stones` `color color` - Color for which the information applies. `int time` - Number of seconds remaining. `int stones` - Number of stones remaining.
effects	none
output	none
fails	syntax error
comments	While the main time is counting, the number of remaining stones is given as 0.

- **final_score**

arguments	none
effects	none
output	`score` `string score` - Score as described in section 4.3.
fails	If the engine is unable to determine the score, fails with error message "cannot score".
comments	If the engine never is able to determine the score, leave the command unimplemented.

- **final_status_list**

arguments	`status` `string status` - Requested status.
effects	none
output	`stones` `vertex*& stones` - Stones with the requested status.
fails	syntax error
comments	See section 4.3 for details.

6.3.5 Regression Commands

- **loadsgf**

  arguments `filename move_number`
  `string filename` - Name of an sgf file.
  `int move_number` - Optional move number.

  effects Board size and komi are set to the values given in the sgf file. Board configuration, number of captured stones, and move history are found by replaying the game record up to the position before `move_number` or until the end if omitted.

  output none

  fails Syntax error. If the file does not exist or cannot be read in because it is broken in some way, fails with the error message "cannot load file".

  comments Due to the syntactical limitations of this protocol, the filename cannot include spaces, hash signs (#), or control characters. The command requires the controller and the engine to share file system, or at least that the controller has sufficient knowledge about the file system of the engine. If `move_number` is larger than the number of moves in the file, read until the end of the file. This command has no support for sgf files with variations or game collections.

- **reg_genmove**

  arguments `color`
  `color color` - Color for which to generate a move.

  effects none

  output `vertex`
  `vertex|string vertex` - Vertex where the engine would want to play a move or the string "resign".

  fails never

  comments This command differs from `genmove` in that it does not play the generated move. It is also advisable to turn off any move randomization since that may cause meaningless regression fluctuations.

6.3.6 Debug Commands

- **showboard**

  arguments none

  effects none

  output `board`
  `string*& board` - A diagram of the board position.

  fails never

  comments The engine may draw the board as it likes. It is, however, required to place the coordinates as described in section 2.11. This command is only intended to help humans with debugging and the output should never need to be parsed by another program.

7 Example

8 Comments on the Specification

8.1 Design Principles

1. The protocol is primarily intended for machine-machine communication. At the same time we want it to be reasonably human readable as well. There are two principal reasons for this. The first one is to make it easy to debug a protocol implementation or to find the error if the communication breaks down, e.g. if two engines involved in a game get their boards out of sync. The second reason is to make it easy to online issue GTP commands, or write scripts, for engine testing.

   The protocol is *not* intended as a user interface for playing games though, even if it can be done for testing purposes.

2. The protocol intentionally does not include any negotiation options. The controller dictates everything and the engine has to comply, unless it is technically unable to, in which case it has to fail. While this to some extent limits the power of the protocol, it considerably simplifies implementation of both engines and controllers.

   Arguably an engine could fail on purpose as some kind of attempt to force negotiation. This is not encouraged and is considered bad style. A controller has absolutely no obligation to try to work around such failures.

8.2 Detail Comments

- **1.3 Communication Model**
  That the controller initiates all communication does not imply that it has to set up the communication *channel*.

- **2.1 Character Set**
  ASCII specifies characters in the interval 0–127, some of which are control characters. "Other characters" refer to characters in the interval 128–255, which are available in various character sets such as the ISO-8859-x series.

- **2.3 Whitespace**
  The requirement to accept both SPACE and HT but only produce SPACE applies to both engine and controller. The reason for this asymmetry is that SPACE is the preferred character but since it for some purposes can be convenient to write scripts of GTP commands manually in text files it is conceivable that an HT may occur occasionally.

- **2.4 Newline Convention**
  The newline convention is easy to implement and allows interoperability between platforms using LF, CRLF, or LFCR to indicate newlines. It does not interoperate with the CR only convention used in text files on MacOS, prior to MacOS X. This is not expected to be a substantial problem.

- **2.8 Timing**
  While the communication channels are required to be free from errors, we do not assume that they are free from delays. For communication over the internet, random delays are a reality and it would be unrealistic not to take this into account.

- **2.9 Comments**
  Comments are mainly useful in regression test suites.

- **2.11 Board Coordinates**
  The choice of board coordinates is guided by design principle 1 of the previous section. The alternative to instead use a pair of integers is slightly easier for a machine to parse and does not impose any limitations on the maximum board size. However, the better human readibility of the chosen format is considered significant enough.

  This coordinate convention is identical to the one used on IGS and many other go servers, and is also used in the specification of the Go Modem Protocol.

- **3.2.1 Simple Entities**
  The only purpose of the `float` entity is to specify komi values. In practice it would suffice to restrict these to small integers and half-integers, but it is probably better to allow general floating point numbers anyway.

- **3.2.2 Compound Entities**
  Since a list can only be stopped by an LF, an entity like {`int* color*`} is invalid, although it would technically be possible to tell the ints from the colors in this particular case.

  For similar reasons constructions of the form {`x& y&`} or `x&*` are also invalid.

  The multiline list construction cannot be used for commands since these are terminated by a single LF.

  In constructions of the form `x*&`, there is no requirement that every `x*` has the same length. I.e., there may be varying number of elements on each line.

- **3.6 Standard Error Messages**
  Failures can be divided into two classes. In the first class we have syntactically incorrect commands and other failures of the controller to follow the specification. In the second class we have technical limitations of the engine, which the controller cannot easily foresee in advance.

  Only failures in the second class have standard error messages, on the assumption that these are the only ones the controller need to take action on. Failures in the first class indicate programming errors and will require debugging. Thus those error messages may freely be chosen to best help the debugging.

- **4.1.1 Fixed Handicap Placement**
  This fixed handicap placement is compatible with the Go Modem Protocol and many go servers.

- **4.1.2 Free Handicap Placement**
  Filling all vertices but one with handicap stones leave them in atari and is obviously not attractive for black. Moreover black has the trivial placement of half the number of stones in a chessboard pattern which leaves white without a single legal move. It is also possible to place a smaller

number of stones than that so that white cannot possibly form a single living group. Thus there clearly exists thresholds above which it is not meaningful to add more stones.

With this in mind it is recommended that every engine has a threshold of at least 40 stones for 19x19 and a proportional number for other board sizes. Extremely weak engines are recommended to provide handicap placements all the way up to the full chessboard pattern.

- **4.2 Time Handling**
  The controller is responsible for measuring time and deciding whether an engine has run out of time. The information given by the `time_settings` and `time_left` commands is only to guide the engines about how fast they need to play.

- **4.3 Scoring**
  With most rulesets scoring is difficult for computer programs. Only sophisticated engines can be expected to have reliable scoring while many engines probably will not implement the scoring commands at all. A controller is recommended to have external methods available to decide the winner (e.g. an independent trusted program or a human reviewer) but there is no reason to invoke them if both engines implement the scoring commands and agree about the result.

  The format of the score string is identical to the RE property in SGF FF[4].

- **5.1 State Variables**
  An engine which never uses some state for anything does not have to keep track of it just because it should. The point of the rule is that the engine is not allowed to suddenly change any state which it does use, unless instructed to.

- **5.2 Default State**
  It may seem natural to require e.g. an empty board when the engine is started. However, it can also be convenient to be able to start the engine with e.g. an sgf file already loaded.

- **6.3.5 Regression Commands**
  GNU Go has a wide array of commands used in regression testing, such as `attack` to test whether a string can be tactically captured. However, except for simple move generation it is not at all clear to what extent such commands can be defined in ways which are meaningful across multiple programs. It is desired to increase the set of standard regression commands in future protocol revisions, but it must be done with care.

  The reason for a separate `reg_genmove` command is that it is usually desirable to get consistent regression results. I.e. for a given position the same move should always be generated. In actual play it is useful to have a random variability between moves of similar value, in particular in the opening, to avoid playing too predictably.

8.3 Missing Features

1. **Ruleset Commands and Scoring Options**
   This version of the protocol has no provisions to specify what ruleset and/or scoring options to use. This is planned for future revisions but has been omitted here due to the complexity of the issue.

   The reason why this is considered complex is that there are numerous rulesets (using the term loosely) such as Japanese, Chinese, AGA, Ing, IGS, and New Zealand, which differ with respect to one or more of ko rule, area or territory scoring, scoring of seki, legality of suicide, effect of handicap stones on scoring, and so on.

   As a workaround this kind of information has to be passed through other channels than GTP, e.g. as command line options when starting the engine.

   In practice this is not all that much of a problem since these settings rarely vary between games, e.g. within a tournament. Still it is desirable to have this functionality in the protocol, but it is worth waiting for a well thought through design of the commands.

2. **Introspective Commands**
   GNU Go includes a large number of commands to query the board, e.g. list legal moves, find connected strings, count liberties, and so on. These can be useful when writing a “stupid” user interface which does not itself know anything about the board logic. They have been omitted from this specification mainly to keep it shorter and make it look less imposing. They are under consideration for inclusion in later revisions.

8.4 Licensing

Anyone may use this protocol for any purpose without any restrictions.

This *document* may be copied, modified, and distributed according to the terms of the GNU Free Documentation License, included in section 10.

While this in theory allows anyone to create modified protocol specifications, which could potentially lead to great chaos, that would benefit noone and we trust people not be that stupid.

The reason why we allow modification at all is to make sure that new authors can continue evolving the protocol if previous authors disappear, without having to rewrite everything from scratch.

People who want to use this protocol as a basis for development of some other protocol are most welcome to start from this protocol specification.

9 Credits

10 GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

10.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

10.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

10.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

10.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

- State on the Title page the name of the publisher of the Modified Version, as the publisher.

- Preserve all the copyright notices of the Document.

- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

10.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

10.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

10.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

10.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

10.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have no Invariant Sections, write “with no Invariant Sections” instead of saying which ones are invariant. If you have no Front-Cover Texts, write “no Front-Cover Texts” instead of “Front-Cover Texts being LIST”; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.