C／C++：

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_BOARD\_H\_

#define \_BOARD\_H\_

#include <stdarg.h>

#include "config.h"

#include "sgftree.h"

#include "winsocket.h"

/\* This type is used to store each intersection on the board.

\*

\* On a 486, char is best, since the time taken to push and pop

\* becomes significant otherwise. On other platforms, an int may

\* be better, e.g. if memcpy() is particularly fast, or if

\* character access is very slow.

\*/

typedef unsigned char Intersection;

/\* FIXME: This is very ugly but we can't include hash.h until we have

\* defined Intersection. And we do need to include it before using

\* Hash\_data.

\*/

#include "hash.h"

/\* local versions of absolute value, min and max \*/

#define gg\_abs(x) ((x) < 0 ? -(x) : (x))

#define gg\_min(a, b) ((a)<(b) ? (a) : (b))

#define gg\_max(a, b) ((a)<(b) ? (b) : (a))

/\* Avoid compiler warnings with unused parameters \*/

#define UNUSED(x) (void)x

/\* A string with n stones can have at most 2(n+1) liberties. From this

\* follows that an upper bound on the number of liberties of a string

\* on a board of size N^2 is 2/3 (N^2+1).

\*/

#define MAXLIBS (2\*(MAX\_BOARD\*MAX\_BOARD + 1)/3)

/\* This is a smaller, practical number of liberties that we care to keep track of. \*/

#define MAX\_LIBERTIES 8

/\* This is an upper bound on the number of strings that can exist on

\* the board simultaneously. Since each string must have at least one

\* liberty and each empty point can provide a liberty to at most four

\* strings, at least one out of five board points must be empty.

\*

\* FIXME: This is not sufficiently large. Above stackp==0, the

\* incremental board code doesn't re-use the entries for

\* removed or merged strings, while new strings require new

\* entries. This is a problem only in very pathological cases,

\* and is extremely unlikely to occur in practice.

\*

\* Actually, in the not all that pathological case of a

\* repeated triple ko cycle, each move creates a new string and

\* thus makes use of one more string, which relatively quickly

\* will exhaust the available strings. For a safe upper bound

\* MAX\_STRINGS should be set to

\* MAX\_STACK + 4 \* MAX\_BOARD \* MAX\_BOARD / 5.

\* It's not clear that it's worth the extra memory, however.

\*/

#define MAX\_STRINGS (4 \* MAX\_BOARD \* MAX\_BOARD / 5)

/\* Per gf: Unconditional\_life() can get very close to filling the

\* entire board under certain circumstances. This was discussed in

\* the list around August 21, 2001, in a thread with the subject

\* "gnugo bug logs".

\*/

#define MAXSTACK MAX\_BOARD \* MAX\_BOARD

#define MAXCHAIN 160

#define HASH\_RANDOM\_SEED 12345

/\* ================================================================ \*

\* One-dimensional board \*

\* ================================================================ \*/

/\* Board sizes \*/

#define MIN\_BOARD 1 /\* Minimum supported board size. \*/

#define MAX\_BOARD 19 /\* Maximum supported board size. \*/

#define MAX\_HANDICAP 9 /\* Maximum supported handicap. \*/

#define MAX\_MOVE\_HISTORY 500 /\* Max number of moves remembered. \*/

#define DEFAULT\_BOARD\_SIZE MAX\_BOARD

/\* Colors and komaster states. \*/

enum colors {

EMPTY,

WHITE,

BLACK,

GRAY,

GRAY\_WHITE,

GRAY\_BLACK,

WEAK\_KO,

NUM\_KOMASTER\_STATES

};

#define COLOR\_NAMES \

"empty", \

"white", \

"black", \

"gray", \

"gray\_white", \

"gray\_black", \

"weak\_ko"

const char \*color\_to\_string(int color);

#define OTHER\_COLOR(color) (WHITE+BLACK-(color))

#define IS\_STONE(arg) ((arg) == WHITE || (arg) == BLACK)

/\* Note that POS(-1, -1) == 0

\* DELTA() is defined so that POS(i+di, j+dj) = POS(i, j) + DELTA(di, dj).

\*/

#define BOARDSIZE ((MAX\_BOARD + 2) \* (MAX\_BOARD + 1) + 1)

#define BOARDMIN (MAX\_BOARD + 2)

#define BOARDMAX (MAX\_BOARD + 1) \* (MAX\_BOARD + 1)

#define POS(i, j) ((MAX\_BOARD + 2) + (i) \* (MAX\_BOARD + 1) + (j))

#define DELTA(di, dj) ((di) \* (MAX\_BOARD + 1) + (dj))

#define I(pos) ((pos) / (MAX\_BOARD + 1) - 1)

#define J(pos) ((pos) % (MAX\_BOARD + 1) - 1)

#define PASS\_MOVE 0

#define NO\_MOVE PASS\_MOVE

#define NS (MAX\_BOARD + 1)

#define WE 1

#define SOUTH(pos) ((pos) + NS)

#define WEST(pos) ((pos) - 1)

#define NORTH(pos) ((pos) - NS)

#define EAST(pos) ((pos) + 1)

#define SW(pos) ((pos) + NS - 1)

#define NW(pos) ((pos) - NS - 1)

#define NE(pos) ((pos) - NS + 1)

#define SE(pos) ((pos) + NS + 1)

#define SS(pos) ((pos) + 2 \* NS)

#define WW(pos) ((pos) - 2)

#define NN(pos) ((pos) - 2 \* NS)

#define EE(pos) ((pos) + 2)

#define DIRECT\_NEIGHBORS(pos1, pos2) \

((pos1) == SOUTH(pos2) \

|| (pos1) == WEST(pos2) \

|| (pos1) == NORTH(pos2) \

|| (pos1) == EAST(pos2))

#define DIAGONAL\_NEIGHBORS(pos1, pos2) \

((pos1) == SW(pos2) \

|| (pos1) == NW(pos2) \

|| (pos1) == NE(pos2) \

|| (pos1) == SE(pos2))

#define BOARD(i, j) board[POS(i, j)]

#define MIRROR\_MOVE(pos) POS(board\_size - 1 - I(pos), board\_size - 1 - J(pos))

/\* ================================================================ \*/

/\* global variables \*/

/\* ================================================================ \*/

/\* The board and the other parameters deciding the current position. \*/

extern int board\_size; /\* board size (usually 19) \*/

extern Intersection board[BOARDSIZE]; /\* go board \*/

extern int board\_ko\_pos;

extern int black\_captured; /\* num. of black stones captured \*/

extern int white\_captured;

extern Intersection initial\_board[BOARDSIZE];

extern int initial\_board\_ko\_pos;

extern int initial\_white\_captured;

extern int initial\_black\_captured;

extern int move\_history\_color[MAX\_MOVE\_HISTORY];

extern int move\_history\_pos[MAX\_MOVE\_HISTORY];

extern Hash\_data move\_history\_hash[MAX\_MOVE\_HISTORY];

extern int move\_history\_pointer;

extern float komi;

extern int handicap; /\* used internally in chinese scoring \*/

extern int movenum; /\* movenumber - used for debug output \*/

extern signed char shadow[BOARDMAX]; /\* reading tree shadow \*/

enum suicide\_rules {

FORBIDDEN,

ALLOWED,

ALL\_ALLOWED

};

extern enum suicide\_rules suicide\_rule;

enum ko\_rules {

SIMPLE,

NONE,

PSK,

SSK

};

extern enum ko\_rules ko\_rule;

extern int stackp; /\* stack pointer \*/

extern int count\_variations; /\* count (decidestring) \*/

extern SGFTree \*sgf\_dumptree;

/\* This struct holds the internal board state. \*/

struct board\_state {

int board\_size;

Intersection board[BOARDSIZE];

int board\_ko\_pos;

int black\_captured;

int white\_captured;

Intersection initial\_board[BOARDSIZE];

int initial\_board\_ko\_pos;

int initial\_white\_captured;

int initial\_black\_captured;

int move\_history\_color[MAX\_MOVE\_HISTORY];

int move\_history\_pos[MAX\_MOVE\_HISTORY];

Hash\_data move\_history\_hash[MAX\_MOVE\_HISTORY];

int move\_history\_pointer;

float komi;

int handicap;

int move\_number;

};

/\* This is increased by one anytime a move is (permanently) played or

\* the board is cleared.

\*/

extern int position\_number;

/\* ================================================================ \*/

/\* board.c functions \*/

/\* ================================================================ \*/

/\* Functions handling the permanent board state. \*/

void clear\_board(void);

int test\_gray\_border(void);

void setup\_board(Intersection new\_board[MAX\_BOARD][MAX\_BOARD], int ko\_pos,

int \*last, float new\_komi, int w\_captured, int b\_captured);

void add\_stone(int pos, int color);

void remove\_stone(int pos);

void play\_move(int pos, int color);

int undo\_move(int n);

void store\_board(struct board\_state \*state);

void restore\_board(struct board\_state \*state);

/\* Information about the permanent board. \*/

int get\_last\_move(void);

int get\_last\_player(void);

int get\_last\_opponent\_move(int color);

int stones\_on\_board(int color);

/\* Functions handling the variable board state. \*/

int trymove(int pos, int color, const char \*message, int str);

int tryko(int pos, int color, const char \*message);

void popgo(void);

int komaster\_trymove(int pos, int color,

const char \*message, int str,

int \*is\_conditional\_ko, int consider\_conditional\_ko);

int get\_komaster(void);

int get\_kom\_pos(void);

int move\_in\_stack(int pos, int cutoff);

void get\_move\_from\_stack(int k, int \*move, int \*color);

void dump\_stack(void);

void do\_dump\_stack(void);

void reset\_trymove\_counter(void);

int get\_trymove\_counter(void);

/\* move properties \*/

int is\_pass(int pos);

int is\_legal(int pos, int color);

int is\_suicide(int pos, int color);

int is\_illegal\_ko\_capture(int pos, int color);

int is\_allowed\_move(int pos, int color);

int is\_ko(int pos, int color, int \*ko\_pos);

int is\_ko\_point(int pos);

int does\_capture\_something(int pos, int color);

int is\_self\_atari(int pos, int color);

/\* Purely geometric functions. \*/

int is\_edge\_vertex(int pos);

int is\_corner\_vertex(int pos);

int edge\_distance(int pos);

int square\_dist(int pos1, int pos2);

int rotate1(int pos, int rot);

/\* Basic string information. \*/

int find\_origin(int str);

int chainlinks(int str, int adj[MAXCHAIN]);

int chainlinks2(int str, int adj[MAXCHAIN], int lib);

int chainlinks3(int str, int adj[MAXCHAIN], int lib);

int extended\_chainlinks(int str, int adj[MAXCHAIN], int both\_colors);

int liberty\_of\_string(int pos, int str);

int second\_order\_liberty\_of\_string(int pos, int str);

int neighbor\_of\_string(int pos, int str);

int has\_neighbor(int pos, int color);

int same\_string(int str1, int str2);

int adjacent\_strings(int str1, int str2);

void mark\_string(int str, signed char mx[BOARDMAX], signed char mark);

int are\_neighbors(int pos1, int pos2);

/\* Count and/or find liberties at (pos). \*/

int countlib(int str);

int findlib(int str, int maxlib, int \*libs);

int fastlib(int pos, int color, int ignore\_captures);

int approxlib(int pos, int color, int maxlib, int \*libs);

int accuratelib(int pos, int color, int maxlib, int \*libs);

int count\_common\_libs(int str1, int str2);

int find\_common\_libs(int str1, int str2, int maxlib, int \*libs);

int have\_common\_lib(int str1, int str2, int \*lib);

/\* Count the number of stones in a string. \*/

int countstones(int str);

int findstones(int str, int maxstones, int \*stones);

int count\_adjacent\_stones(int str1, int str2, int maxstones);

/\* Detect a special shape. \*/

int send\_two\_return\_one(int move, int color);

/\* Special function for reading.c \*/

void incremental\_order\_moves(int move, int color, int string,

int \*number\_edges, int \*number\_same\_string,

int \*number\_own, int \*number\_opponent,

int \*captured\_stones, int \*threatened\_stones,

int \*saved\_stones, int \*number\_open);

/\* Board caches initialization functions. \*/

void clear\_approxlib\_cache(void);

void clear\_accuratelib\_cache(void);

/\* Is this point inside the board? \*/

#if 0

#define ON\_BOARD2(i, j) ((i)>=0 && (j)>=0 && (i)<board\_size && (j)<board\_size)

#else

/\*

\* For the case when expr can only be slightly negative,

\* if (expr < 0 || expr > something)

\* is equivalent to

\* if ((unsigned) expr > something)

\*

\* (I think gcc knows this trick, but it does no harm to

\* encode it explicitly since it saves typing !)

\*/

#define ON\_BOARD2(i, j) ((unsigned) (i) < (unsigned) board\_size &&\

(unsigned) (j) < (unsigned) board\_size)

#endif

#define ASSERT\_ON\_BOARD2(i, j) ASSERT2(ON\_BOARD2((i), (j)), (i), (j))

#define ON\_BOARD1(pos) (((unsigned) (pos) < BOARDSIZE) && board[pos] != GRAY)

#define ON\_BOARD(pos) (board[pos] != GRAY)

#define ASSERT\_ON\_BOARD1(pos) ASSERT1(ON\_BOARD1(pos), (pos))

/\* Coordinates for the eight directions, ordered

\* south, west, north, east, southwest, northwest, northeast, southeast.

\* Defined in board.c.

\*/

extern int deltai[8]; /\* = { 1, 0, -1, 0, 1, -1, -1, 1}; \*/

extern int deltaj[8]; /\* = { 0, -1, 0, 1, -1, -1, 1, 1}; \*/

extern int delta[8]; /\* = { NS, -1, -NS, 1, NS-1, -NS-1, -NS+1, NS+1}; \*/

/\* ================================================================ \*/

/\* Other functions \*/

/\* ================================================================ \*/

/\* SGF routines for debugging purposes in sgffile.c \*/

void sgffile\_begindump(struct SGFTree\_t \*tree);

void sgffile\_enddump(const char \*filename);

/\* Hashing and Caching statistics. \*/

struct stats\_data {

int nodes; /\* Number of visited nodes while reading \*/

int read\_result\_entered; /\* Number of read results entered. \*/

int read\_result\_hits; /\* Number of hits of read results. \*/

int trusted\_read\_result\_hits; /\* Number of hits of read results \*/

/\* with sufficient remaining depth. \*/

};

extern struct stats\_data stats;

/\* printutils.c \*/

int gprintf(const char \*fmt, ...);

void vgprintf(FILE \*outputfile, const char \*fmt, va\_list ap);

void mprintf(const char \*fmt, ...);

void gfprintf(FILE \*outfile, const char \*fmt, ...);

const char \*color\_to\_string(int color);

const char \*location\_to\_string(int pos);

void location\_to\_buffer(int pos, char \*buf);

int string\_to\_location(int boardsize, const char \*str);

int is\_hoshi\_point(int m, int n);

void draw\_letter\_coordinates(FILE \*outfile);

void simple\_showboard(FILE \*outfile);

void mark\_goal\_in\_sgf(signed char goal[BOARDMAX]);

/\* ================================================================ \*/

/\* assertions \*/

/\* ================================================================ \*/

/\* Our own abort() which prints board state on the way out.

\* (pos) is a "relevant" board position for info.

\*/

void abortgo(const char \*file, int line, const char \*msg, int pos)

#ifdef \_\_GNUC\_\_

\_\_attribute\_\_ ((noreturn))

#endif

;

#ifdef GG\_TURN\_OFF\_ASSERTS

#define ASSERT2(x, i, j)

#define ASSERT1(x, pos)

#else

/\* avoid dangling else \*/

/\* FIXME: Should probably re-write these using do {...} while (0) idiom. \*/

#define ASSERT2(x, i, j) if (x) ; else abortgo(\_\_FILE\_\_, \_\_LINE\_\_, #x, POS(i, j))

#define ASSERT1(x, pos) if (x) ; else abortgo(\_\_FILE\_\_, \_\_LINE\_\_, #x, pos)

#endif

#define gg\_assert(x) ASSERT1(x, NO\_MOVE)

/\* Are we using valgrind memory checking? \*/

#if USE\_VALGRIND

#include <valgrind/memcheck.h>

#else

#define VALGRIND\_MAKE\_WRITABLE(a, b)

#endif

#endif /\* \_BOARD\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_CACHE\_H\_

#define \_CACHE\_H\_

#include <stdio.h>

#include "hash.h"

/\*

\* This file, together with engine/hash.c implements hashing of go positions

\* using a method known as Zobrist hashing. See the Texinfo documentation

\* (Reading/Hashing) for more information.

\*/

/\* Hashnode: a node stored in the transposition table.

\*

\* In addition to the position, the hash lock encodes the following data,

\* all hashed:

\* komaster

\* kom\_pos

\* routine

\* str1

\* str2

\* extra hashvalue, optional (e.g. encoding a goal array)

\*

\* The data field packs into 32 bits the following

\* fields:

\*

\* RESERVED : 5 bits

\* value1 : 4 bits

\* value2 : 4 bits

\* move : 10 bits

\* cost : 4 bits

\* remaining\_depth: 5 bits (depth - stackp) NOTE: HN\_MAX\_REMAINING\_DEPTH

\*

\* The last 9 bits together give an index for the total costs.

\*/

typedef struct {

Hash\_data key;

unsigned int data; /\* Should be 32 bits, but only wastes 25% if 64 bits. \*/

} Hashnode;

#define HN\_MAX\_REMAINING\_DEPTH 31

/\* Hashentry: an entry, with two nodes of the hash\_table

\*/

typedef struct {

Hashnode deepest;

Hashnode newest;

} Hashentry;

/\* Hn is for hash node. \*/

#define hn\_get\_value1(hn) ((hn >> 23) & 0x0f)

#define hn\_get\_value2(hn) ((hn >> 19) & 0x0f)

#define hn\_get\_move(hn) ((hn >> 9) & 0x3ff)

#define hn\_get\_cost(hn) ((hn >> 5) & 0x0f)

#define hn\_get\_remaining\_depth(hn) ((hn >> 0) & 0x1f)

#define hn\_get\_total\_cost(hn) ((hn >> 0) & 0x1ff)

#define hn\_create\_data(remaining\_depth, value1, value2, move, cost) \

((((value1) & 0x0f) << 23) \

| (((value2) & 0x0f) << 19) \

| (((move) & 0x3ff) << 9) \

| (((cost) & 0x0f) << 5) \

| (((remaining\_depth & 0x1f) << 0)))

/\* Transposition\_table: transposition table used for caching. \*/

typedef struct {

unsigned int num\_entries;

Hashentry \*entries;

int is\_clean;

} Transposition\_table;

extern Transposition\_table ttable;

/\* Number of cache entries to use by default if no cache memory usage

\* has been set explicitly.

\*/

#define DEFAULT\_NUMBER\_OF\_CACHE\_ENTRIES 350000

void tt\_free(Transposition\_table \*table);

int tt\_get(Transposition\_table \*table, enum routine\_id routine,

int target1, int target2, int remaining\_depth,

Hash\_data \*extra\_hash,

int \*value1, int \*value2, int \*move);

void tt\_update(Transposition\_table \*table, enum routine\_id routine,

int target, int target2, int remaining\_depth,

Hash\_data \*extra\_hash,

int value1, int value2, int move);

/\* ================================================================ \*/

/\* Macros used from reading.c, readconnect.c, and owl.c to store and

\* retrieve read results.

\*/

#if TRACE\_READ\_RESULTS

#define TRACE\_CACHED\_RESULT(result, move) \

gprintf("%o%s %1m %d %d %1m (cached) ", read\_function\_name, \

q, stackp, result, move); \

dump\_stack();

#define TRACE\_CACHED\_RESULT2(result1, result2, move) \

gprintf("%o%s %1m %1m %d %d %d %1m (cached) ", read\_function\_name, \

q1, q2, stackp, result1, result2, move); \

dump\_stack();

#define SETUP\_TRACE\_INFO(name, str) \

const char \*read\_function\_name = name; \

int q = find\_origin(str);

#define SETUP\_TRACE\_INFO2(name, str1, str2) \

const char \*read\_function\_name = name; \

int q1 = board[str1] == EMPTY ? str1 : find\_origin(str1); \

int q2 = board[str2] == EMPTY ? str2 : find\_origin(str2);

#else

#define TRACE\_CACHED\_RESULT(result, move)

#define TRACE\_CACHED\_RESULT2(result1, result2, move)

#define SETUP\_TRACE\_INFO(name, str) \

const char \*read\_function\_name = name; \

int q = str;

#define SETUP\_TRACE\_INFO2(name, str1, str2) \

const char \*read\_function\_name = name; \

int q1 = str1; \

int q2 = str2;

#endif

/\* Trace messages in decidestring/decidedragon sgf file. \*/

void sgf\_trace(const char \*func, int str, int move, int result,

const char \*message);

/\* Trace messages in decideconnection sgf file. \*/

void sgf\_trace2(const char \*func, int str1, int str2, int move,

const char \*result, const char \*message);

/\* Trace messages in decidesemeai sgf file. \*/

void sgf\_trace\_semeai(const char \*func, int str1, int str2, int move,

int result1, int result2, const char \*message);

/\* Macro to hide the call to sgf\_trace(). Notice that a little black

\* magic is going on here. Before using this macro, SETUP\_TRACE\_INFO

\* must have been called to provide the variables read\_function\_name

\* and q. These must of course not be used for anything else in

\* the function.

\*/

#define SGFTRACE(move, result, message) \

if (sgf\_dumptree) \

sgf\_trace(read\_function\_name, q, move, result, message)

/\* Corresponding macro for use in connection or semeai reading, where

\* two groups are involved.

\*/

#define SGFTRACE2(move, result, message) \

if (sgf\_dumptree) \

sgf\_trace2(read\_function\_name, q1, q2, move, \

result\_to\_string(result), message)

#define SGFTRACE\_SEMEAI(move, result1, result2, message) \

if (sgf\_dumptree) \

sgf\_trace\_semeai(read\_function\_name, q1, q2, move, \

result1, result2, message)

/\* ================================================================ \*/

/\*

\* These macros should be used in all the places where we want to

\* return a result from a reading function and where we want to

\* store the result in the hash table at the same time.

\*/

#if !TRACE\_READ\_RESULTS

#define READ\_RETURN0(routine, str, remaining\_depth) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

0, 0, NO\_MOVE);\

return 0; \

} while (0)

#define READ\_RETURN(routine, str, remaining\_depth, point, move, value) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

return (value); \

} while (0)

#define READ\_RETURN\_SEMEAI(routine, str1, str2, remaining\_depth, point, move, value1, value2) \

do { \

tt\_update(&ttable, routine, str1, str2, remaining\_depth, NULL, \

value1, value2, move); \

if ((value1) != 0 && (point) != 0) \*(point) = (move); \

return; \

} while (0)

#define READ\_RETURN\_CONN(routine, str1, str2, remaining\_depth, point, move, value) \

do { \

tt\_update(&ttable, routine, str1, str2, remaining\_depth, NULL,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

return (value); \

} while (0)

#define READ\_RETURN\_HASH(routine, str, remaining\_depth, hash, point, move, value) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, hash,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

return (value); \

} while (0)

#define READ\_RETURN2(routine, str, remaining\_depth, point, move, value1, value2) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

value1, value2, move);\

if ((value1) != 0 && (point) != 0) \*(point) = (move); \

return (value1); \

} while (0)

#else /\* !TRACE\_READ\_RESULTS \*/

#define READ\_RETURN0(routine, str, remaining\_depth) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

0, 0, NO\_MOVE);\

gprintf("%o%s %1m %d 0 0 ", read\_function\_name, q, stackp); \

dump\_stack(); \

return 0; \

} while (0)

#define READ\_RETURN(routine, str, remaining\_depth, point, move, value) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

gprintf("%o%s %1m %d %d %1m ", read\_function\_name, q, stackp, \

(value), (move)); \

dump\_stack(); \

return (value); \

} while (0)

#define READ\_RETURN\_SEMEAI(routine, str1, str2, remaining\_depth, point, move, value1, value2) \

do { \

tt\_update(&ttable, routine, str1, str2, remaining\_depth, NULL, \

value1, value2, move); \

if ((value1) != 0 && (point) != 0) \*(point) = (move); \

gprintf("%o%s %1m %1m %d %d %d %1m ", read\_function\_name, q1, q2, stackp, \

(value1), (value2), (move)); \

dump\_stack(); \

return; \

} while (0)

#define READ\_RETURN\_CONN(routine, str1, str2, remaining\_depth, point, move, value) \

do { \

tt\_update(&ttable, routine, str1, str2, remaining\_depth, NULL,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

gprintf("%o%s %1m %1m %d %d %1m ", read\_function\_name, q1, q2, stackp, \

(value), (move)); \

dump\_stack(); \

return (value); \

} while (0)

#define READ\_RETURN\_HASH(routine, str, remaining\_depth, hash, point, move, value) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, hash,\

value, 0, move);\

if ((value) != 0 && (point) != 0) \*(point) = (move); \

gprintf("%o%s %1m %d %d %1m ", read\_function\_name, q, stackp, \

(value), (move)); \

dump\_stack(); \

return (value); \

} while (0)

#define READ\_RETURN2(routine, str, remaining\_depth, point, move, value1, value2) \

do { \

tt\_update(&ttable, routine, str, NO\_MOVE, remaining\_depth, NULL,\

value1, value2, move);\

if ((value1) != 0 && (point) != 0) \*(point) = (move); \

gprintf("%o%s %1m %d %d %1m ", read\_function\_name, q, stackp, \

(value1), (move)); \

dump\_stack(); \

return (value1); \

} while (0)

#endif

/\* ================================================================ \*/

/\* This has actually nothing to do with caching, but is useful in

\* the same places where the caching is.

\*/

/\* Macro to use when saving ko results while continuing to look for an

\* unconditional result. It's assumed that we have tried the move at

\* (move) and then called an attack or defense function giving the

\* result passed in the code parameter.

\*

\* In general we prefer not to have to do the first ko threat. Thus a

\* savecode KO\_A is always better than a savecode KO\_B. Also we always

\* prefer to keep the old move if we get the same savecode once more,

\* on the assumption that the moves have been ordered with the

\* presumably best one first.

\*

\* Notice that the savecode may be either 0 (nothing found so far), KO\_B

\* or KO\_A. Occasionally savecode WIN is also used, indicating an effective

\* but not preferred move, typically because it's either a sacrifice

\* or a backfilling move. If possible, we prefer making non-sacrifice

\* and direct moves. Of course savecode WIN is better than KO\_A or KO\_B.

\*/

#define UPDATE\_SAVED\_KO\_RESULT(savecode, save, code, move) \

if (code != 0 && REVERSE\_RESULT(code) > savecode) { \

save = move; \

savecode = REVERSE\_RESULT(code); \

} \

/\* Same as above, except this should be used when there's no

\* intervening trymove(). Thus we shouldn't reverse the save code.

\*/

#define UPDATE\_SAVED\_KO\_RESULT\_UNREVERSED(savecode, save, code, move) \

if (code != WIN && code > savecode) { \

save = move; \

savecode = code; \

}

/\* This too isn't really related to caching but is convenient to have here.

\* (Needs to be available in reading.c and persistent.c.)

\*

\* Minimum number of nodes for which DEBUG\_READING\_PERFORMANCE reports

\* anything.

\*/

#define MIN\_READING\_NODES\_TO\_REPORT 1000

#endif /\* \_CACHE\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_CLOCK\_H\_

#define \_CLOCK\_H\_

#include "gnugo.h"

/\* initialization and activation \*/

void clock\_settings(int maintime, int byotime, int byostones);

void init\_timers(void);

/\* main access \*/

void clock\_push\_button(int color);

void update\_time\_left(int color, int time\_left, int stones);

void clock\_print(int color);

int have\_time\_settings(void);

void adjust\_level\_offset(int color);

/\* Access to level settings. \*/

int get\_level(void);

void set\_level(int new\_level);

void set\_max\_level(int new\_max);

void set\_min\_level(int new\_min);

#endif /\* \_CLOCK\_H\_ \*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

/\* ---------------------------------------------------------------- \*

\* gnugo.h

\* This file contains the public interface to the GNU Go engine.

\* ---------------------------------------------------------------- \*/

#ifndef \_GNUGO\_H\_

#define \_GNUGO\_H\_

#include "board.h"

#include <stdio.h>

#ifdef HAVE\_CONFIG\_H

#include <config.h>

#endif

#ifdef HAVE\_CRTDBG\_H

#include <crtdbg.h>

#endif

#include "sgftree.h"

#include "clock.h"

#include "winsocket.h"

/\* interface.c \*/

/\* Initialize the whole thing. Should be called once. \*/

void init\_gnugo(float memory, unsigned int random\_seed);

/\* ================================================================ \*/

/\* some public macros used everywhere \*/

/\* ================================================================ \*/

/\* Used in matchpat.c. Have to be different from WHITE, BLACK. \*/

#define ANCHOR\_COLOR 6

#define ANCHOR\_OTHER 7

/\* Return codes for reading functions \*/

#define WIN 5

#define KO\_A 4

#define GAIN 3

#define LOSS 2

#define KO\_B 1

#define LOSE 0

const char \*result\_to\_string(int result);

/\* Used by break\_through(). Must be different from 0 and WIN. \*/

#define CUT 2

/\* Surrounded \*/

#define SURROUNDED 1

#define WEAKLY\_SURROUNDED 2

/\* ================================================================ \*/

/\* Board manipulation \*/

/\* ================================================================ \*/

int check\_boardsize(int boardsize, FILE \*out);

void gnugo\_clear\_board(int boardsize);

void gnugo\_play\_move(int move, int color);

int gnugo\_play\_sgfnode(SGFNode \*node, int to\_move);

int gnugo\_sethand(int desired\_handicap, SGFNode \*node);

float gnugo\_estimate\_score(float \*upper, float \*lower);

/\* ================================================================ \*/

/\* Game handling \*/

/\* ================================================================ \*/

typedef struct {

int handicap;

int to\_move; /\* whose move it currently is \*/

SGFTree game\_record; /\* Game record in sgf format. \*/

int computer\_player; /\* BLACK, WHITE, or EMPTY (used as BOTH) \*/

} Gameinfo;

void gameinfo\_clear(Gameinfo \*ginfo);

void gameinfo\_print(Gameinfo \*ginfo);

int gameinfo\_play\_sgftree\_rot(Gameinfo \*gameinfo, SGFTree \*tree,

const char \*untilstr, int orientation);

int gameinfo\_play\_sgftree(Gameinfo \*gameinfo, SGFTree \*tree,

const char \*untilstr);

/\* ================================================================ \*/

/\* global variables \*/

/\* ================================================================ \*/

/\* Miscellaneous debug options. \*/

extern int quiet; /\* Minimal output. \*/

extern int verbose; /\* Bore the opponent. \*/

extern int allpats; /\* generate all patterns, even small ones \*/

extern int printworms; /\* print full data on each string \*/

extern int printmoyo; /\* print moyo board each move \*/

extern int printdragons; /\* print full data on each dragon \*/

extern int printboard; /\* print board each move \*/

extern int showstatistics; /\* print statistics \*/

extern int profile\_patterns; /\* print statistics of pattern usage \*/

extern char outfilename[128]; /\* output file (-o option) \*/

extern int output\_flags; /\* amount of output to outfile \*/

/\* output flag bits \*/

#define OUTPUT\_MARKDRAGONS 0x0001 /\* mark dead and critical dragons \*/

#define OUTPUT\_MOVEVALUES 0x0002 /\* output values of all moves in list \*/

#define OUTPUT\_DEFAULT 0 /\* no debug output by default \*/

/\* debug flag bits \*/

/\* NOTE : can specify -d0x... \*/

/\* Please keep this list in sync with the DEBUG\_FLAGS string below. \*/

#define DEBUG\_INFLUENCE 0x0001

#define DEBUG\_EYES 0x0002

#define DEBUG\_OWL 0x0004

#define DEBUG\_ESCAPE 0x0008

#define DEBUG\_MATCHER 0x0010

#define DEBUG\_DRAGONS 0x0020

#define DEBUG\_SEMEAI 0x0040

#define DEBUG\_LOADSGF 0x0080

#define DEBUG\_HELPER 0x0100

#define DEBUG\_READING 0x0200

#define DEBUG\_WORMS 0x0400

#define DEBUG\_MOVE\_REASONS 0x0800

#define DEBUG\_OWL\_PERFORMANCE 0x1000

#define DEBUG\_BREAKIN 0x2000

#define DEBUG\_FILLLIB 0x4000

#define DEBUG\_READING\_PERFORMANCE 0x8000

#define DEBUG\_SCORING 0x010000

#define DEBUG\_AFTERMATH 0x020000

#define DEBUG\_ATARI\_ATARI 0x040000

#define DEBUG\_READING\_CACHE 0x080000

#define DEBUG\_TERRITORY 0x100000

#define DEBUG\_PERSISTENT\_CACHE 0x200000

#define DEBUG\_TOP\_MOVES 0x400000

#define DEBUG\_MISCELLANEOUS 0x800000

#define DEBUG\_ORACLE\_STREAM 0x1000000

#define DEBUG\_LARGE\_SCALE 0x1000000

#define DEBUG\_SPLIT\_OWL 0x2000000

#define DEBUG\_TIME 0x4000000

#define DEBUG\_FLAGS "\

DEBUG\_INFLUENCE 0x0001\n\

DEBUG\_EYES 0x0002\n\

DEBUG\_OWL 0x0004\n\

DEBUG\_ESCAPE 0x0008\n\

DEBUG\_MATCHER 0x0010\n\

DEBUG\_DRAGONS 0x0020\n\

DEBUG\_SEMEAI 0x0040\n\

DEBUG\_LOADSGF 0x0080\n\

DEBUG\_HELPER 0x0100\n\

DEBUG\_READING 0x0200\n\

DEBUG\_WORMS 0x0400\n\

DEBUG\_MOVE\_REASONS 0x0800\n\

DEBUG\_OWL\_PERFORMANCE 0x1000\n\

DEBUG\_BREAKIN 0x2000\n\

DEBUG\_FILLLIB 0x4000\n\

DEBUG\_READING\_PERFORMANCE 0x8000\n\

DEBUG\_SCORING 0x010000\n\

DEBUG\_AFTERMATH 0x020000\n\

DEBUG\_ATARI\_ATARI 0x040000\n\

DEBUG\_READING\_CACHE 0x080000\n\

DEBUG\_TERRITORY 0x100000\n\

DEBUG\_PERSISTENT\_CACHE 0x200000\n\

DEBUG\_TOP\_MOVES 0x400000\n\

DEBUG\_MISCELLANEOUS 0x800000\n\

DEBUG\_ORACLE\_STREAM 0x1000000\n\

DEBUG\_LARGE\_SCALE 0x1000000\n\

DEBUG\_SPLIT\_OWL 0x2000000\n\

DEBUG\_TIME 0x4000000\n\

"

extern int debug; /\* debug flags \*/

extern int fusekidb; /\* use fuseki database \*/

extern int disable\_fuseki; /\* do not generate fuseki moves \*/

extern int josekidb; /\* use joseki database \*/

extern int semeai\_variations; /\* max variations considered reading semeai \*/

extern int showtime; /\* print genmove time \*/

extern int showscore; /\* print score \*/

extern int chinese\_rules; /\* use chinese (area) rules for counting \*/

extern int experimental\_owl\_ext; /\* use experimental owl (GAIN/LOSS) \*/

extern int experimental\_connections; /\* use experimental connection module \*/

extern int alternate\_connections; /\* use alternate connection module \*/

extern int owl\_threats; /\* compute owl threats \*/

extern int capture\_all\_dead; /\* capture all dead opponent stones \*/

extern int play\_out\_aftermath; /\* make everything unconditionally settled \*/

extern int resign\_allowed; /\* allows GG to resign hopeless games \*/

extern int play\_mirror\_go; /\* try to play mirror go if possible \*/

extern int mirror\_stones\_limit; /\* but stop at this number of stones \*/

extern int gtp\_version; /\* version of Go Text Protocol \*/

extern int use\_monte\_carlo\_genmove; /\* use Monte Carlo move generation \*/

extern int mc\_games\_per\_level; /\* number of Monte Carlo simulations per level \*/

/\* Mandatory values of reading parameters. Normally -1, if set

\* these override the values derived from the level. \*/

extern int mandated\_depth;

extern int mandated\_backfill\_depth;

extern int mandated\_backfill2\_depth;

extern int mandated\_break\_chain\_depth;

extern int mandated\_superstring\_depth;

extern int mandated\_fourlib\_depth;

extern int mandated\_ko\_depth;

extern int mandated\_branch\_depth;

extern int mandated\_aa\_depth;

extern int mandated\_owl\_distrust\_depth;

extern int mandated\_owl\_branch\_depth;

extern int mandated\_owl\_reading\_depth;

extern int mandated\_owl\_node\_limit;

extern int mandated\_semeai\_node\_limit;

extern int autolevel\_on;

extern float potential\_moves[BOARDMAX];

extern int oracle\_exists; /\* oracle is available for consultation \*/

extern int metamachine; /\* use metamachine\_genmove \*/

/\* ================================================================ \*/

/\* tracing and debugging functions \*/

/\* ================================================================ \*/

/\* Colors. \*/

#define GG\_COLOR\_BLACK 0

#define GG\_COLOR\_RED 1

#define GG\_COLOR\_GREEN 2

#define GG\_COLOR\_YELLOW 3

#define GG\_COLOR\_BLUE 4

#define GG\_COLOR\_MAGENTA 5

#define GG\_COLOR\_CYAN 6

#define GG\_COLOR\_WHITE 7

/\* showbord.c \*/

void start\_draw\_board(void);

void draw\_color\_char(int m, int n, int c, int color);

void draw\_char(int m, int n, int c);

void end\_draw\_board(void);

void showboard(int xo); /\* ascii rep. of board to stderr \*/

/\* influence.c \*/

void debug\_influence\_move(int move);

#define TRACE (!(verbose)) ? (void)0 : (void)gprintf

#ifdef HAVE\_VARIADIC\_DEFINE

/\* gnuc allows variadic macros, so the tests can be done inline \*/

#define DEBUG(level, fmt, args...) \

do { if ((debug & (level))) gprintf(fmt, ##args); } while (0)

#else /\*HAVE\_VARIADIC\_DEFINE\*/

/\* if debug == 0, then can skip the function call. \*/

#define DEBUG (!(debug)) ? (void)0 : (void)DEBUG\_func

int DEBUG\_func(int level, const char \*fmt, ...);

#endif /\*HAVE\_VARIADIC\_DEFINE\*/

/\* genmove.c \*/

#define EXAMINE\_WORMS 1

#define EXAMINE\_INITIAL\_INFLUENCE 2

#define EXAMINE\_DRAGONS\_WITHOUT\_OWL 3

#define EXAMINE\_DRAGONS 4

#define EXAMINE\_OWL\_REASONS 5

#define EXAMINE\_INITIAL\_INFLUENCE2 6

#define FULL\_EXAMINE\_DRAGONS 7

#define EXAMINE\_ALL 99

void reset\_engine(void);

void examine\_position(int how\_much, int aftermath\_play);

void silent\_examine\_position(int how\_much);

/\* ================================================================ \*/

/\* statistics functions \*/

/\* ================================================================ \*/

/\* These are mostly used for GTP examination. \*/

void reset\_owl\_node\_counter(void);

int get\_owl\_node\_counter(void);

void reset\_reading\_node\_counter(void);

int get\_reading\_node\_counter(void);

void reset\_connection\_node\_counter(void);

int get\_connection\_node\_counter(void);

/\* ================================================================ \*/

/\* Low level functions \*/

/\* ================================================================ \*/

/\* utils.c \*/

void who\_wins(int color, FILE \*outfile);

/\* high-level routine to generate the best move for the given color \*/

int genmove(int color, float \*value, int \*resign);

int genmove\_conservative(int color, float \*value);

/\* Play through the aftermath. \*/

float aftermath\_compute\_score(int color, SGFTree \*tree);

/\* Basic information gathering. \*/

/\* worm.c \*/

void make\_worms(void);

void compute\_worm\_influence(void);

/\* dragon.c \*/

void make\_dragons(int stop\_before\_owl);

void initialize\_dragon\_data(void);

void show\_dragons(void);

enum dragon\_status crude\_status(int pos);

enum dragon\_status dragon\_status(int pos);

int same\_dragon(int dr1, int dr2);

/\* debugging functions \*/

void prepare\_pattern\_profiling(void);

void report\_pattern\_profiling(void);

/\* sgffile.c \*/

void sgffile\_add\_debuginfo(SGFNode \*node, float value);

void sgffile\_output(SGFTree \*tree);

void sgffile\_printsgf(int color\_to\_play, const char \*filename);

void sgffile\_printboard(SGFTree \*tree);

void sgffile\_recordboard(SGFNode \*node);

int get\_sgfmove(SGFProperty \*property);

/\* sgfdecide.c \*/

void decide\_string(int pos);

void decide\_connection(int apos, int bpos);

void decide\_owl(int pos);

void decide\_dragon\_data(int pos);

void decide\_semeai(int apos, int bpos);

void decide\_tactical\_semeai(int apos, int bpos);

void decide\_position(void);

void decide\_eye(int pos);

void decide\_combination(int color);

void decide\_surrounded(int pos);

void decide\_oracle(Gameinfo \*gameinfo, char \*infilename, char \*untilstring);

/\*oracle.c\*/

void dismiss\_oracle(void);

void oracle\_clear\_board(int boardsize);

#endif /\* \_GNUGO\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_HASH\_H\_

#define \_HASH\_H\_

#include "config.h"

#include <limits.h>

/\*

\* This file, together with engine/hash.c implements hashing of go positions

\* using a method known as Zobrist hashing. See the Texinfo documentation

\* (Reading/Hashing) for more information.

\*/

/\* Hash values and the compact board representation should use the

\* longest integer type that the platform can handle efficiently.

\* Typically this would be a 32 bit integer on a 32 bit platform and a

\* 64 bit integer on a 64 bit platform.

\*

\* Our current assumption is that unsigned long has this

\* characteristic. Should it turn out to be false for some platform

\* we'll add conditional code to choose some other type.

\*

\* At the few places in the code where the actual size of these types

\* matter, the code should use sizeof(type) to test for this. Notice

\* that ISO C guarantees a long to be at least 32 bits.

\*

\* On (future) platforms with word length 128 bits or more, it might

\* be a waste to use more than 64 bit hashvalues, since the decreased

\* risk for hash collisions probably isn't worth the increased storage

\* cost.

\*/

typedef unsigned long Hashvalue;

#define SIZEOF\_HASHVALUE SIZEOF\_LONG

#define HASHVALUE\_PRINT\_FORMAT "%0\*lx"

/\* for testing: Enables a lot of checks. \*/

#define CHECK\_HASHING 0

/\* Dump (almost) all read results. \*/

#define TRACE\_READ\_RESULTS 0

/\* How many bits should be used at least for hashing? Set this to 32 for

\* some memory save and speedup, at the cost of occasional irreproducable

\* mistakes (and possibly assertion failures).

\* With 64 bits, there should be less than one such mistake in 10^9 games.

\* Set this to 96 if this is not safe enough for you.

\*/

#define MIN\_HASHBITS 64

#define NUM\_HASHVALUES (1 + (MIN\_HASHBITS - 1) / (CHAR\_BIT \* SIZEOF\_HASHVALUE))

/\* This struct is maintained by the machinery that updates the board

\* to provide incremental hashing. Examples: trymove(), play\_move(), ...

\*/

typedef struct {

Hashvalue hashval[NUM\_HASHVALUES];

} Hash\_data;

extern Hash\_data board\_hash;

Hash\_data goal\_to\_hashvalue(const signed char \*goal);

void hash\_init\_zobrist\_array(Hash\_data \*array, int size);

void hash\_init(void);

#define INIT\_ZOBRIST\_ARRAY(a) \

hash\_init\_zobrist\_array(a, (int) (sizeof(a) / sizeof(a[0])))

void hashdata\_clear(Hash\_data \*hd);

void hashdata\_recalc(Hash\_data \*hd, Intersection \*board, int ko\_pos);

void hashdata\_invert\_ko(Hash\_data \*hd, int pos);

void hashdata\_invert\_stone(Hash\_data \*hd, int pos, int color);

void hashdata\_invert\_komaster(Hash\_data \*hd, int komaster);

void hashdata\_invert\_kom\_pos(Hash\_data \*hd, int kom\_pos);

void hashdata\_calc\_orientation\_invariant(Hash\_data \*hd, Intersection \*board,

int ko\_pos);

char \*hashdata\_to\_string(Hash\_data \*hashdata);

/\* ---------------------------------------------------------------- \*/

/\* There is no need to involve all bits in the remainder computation

\* as long as we only use it to compute a key into a hash table. 32

\* random bits are sufficient to get an even distribution within any

\* hashtable of reasonable size. By never using more than 32 bits we

\* also reduce the platform dependency of the GNU Go engine.

\*/

#define hashdata\_remainder(hd, num) \

(((hd).hashval[0] & 0xffffffffU) % (num))

#if NUM\_HASHVALUES == 1

#define hashdata\_is\_equal(hd1, hd2) \

((hd1).hashval[0] == (hd2).hashval[0])

#define hashdata\_is\_smaller(hd1, hd2) \

((hd1).hashval[0] < (hd2).hashval[0])

#define hashdata\_xor(hd1, hd2) \

(hd1).hashval[0] ^= (hd2).hashval[0]

#elif NUM\_HASHVALUES == 2

#define hashdata\_is\_equal(hd1, hd2) \

((hd1).hashval[0] == (hd2).hashval[0] \

&& (hd1).hashval[1] == (hd2).hashval[1])

#define hashdata\_is\_smaller(hd1, hd2) \

((hd1).hashval[0] < (hd2).hashval[0] \

|| ((hd1).hashval[0] == (hd2).hashval[0] \

&& (hd1).hashval[1] < (hd2).hashval[1]))

#define hashdata\_xor(hd1, hd2) \

do { \

(hd1).hashval[0] ^= (hd2).hashval[0]; \

(hd1).hashval[1] ^= (hd2).hashval[1]; \

} while (0)

#else

int hashdata\_is\_equal\_func(Hash\_data \*hd1, Hash\_data \*hd2);

int hashdata\_is\_smaller\_func(Hash\_data \*hd1, Hash\_data \*hd2);

#define hashdata\_is\_equal(hd1, hd2) \

hashdata\_is\_equal\_func(&(hd1), &(hd2))

#define hashdata\_is\_smaller(hd1, hd2) \

hashdata\_is\_smaller\_func(&(hd1), &(hd2))

#define hashdata\_xor(hd1, hd2) \

do { \

int i; \

for (i = 0; i < NUM\_HASHVALUES; i++) \

(hd1).hashval[i] ^= (hd2).hashval[i]; \

} while (0)

#endif

#endif

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <string.h>

#include "liberty.h"

#include "winsocket.h"

/\* The cosmic style uses more influence than the defaults attenuation

\* coefficients !

\* The "TERR\_.."-values are used in the influence computations used

\* for territory evaluation. (initial\_influence with dragons\_known,

\* move\_influence)

\*/

#define DEFAULT\_ATTENUATION \

(cosmic\_importance \* 2.7 + (1.0 - cosmic\_importance) \* 3.0)

#define TERR\_DEFAULT\_ATTENUATION \

(cosmic\_importance \* 2.15 + (1.0 - cosmic\_importance) \* 2.4)

/\* Extra damping coefficient for spreading influence diagonally. \*/

#define DIAGONAL\_DAMPING \

(cosmic\_importance \* 2.5 + (1.0 - cosmic\_importance) \* 2.0)

#define TERR\_DIAGONAL\_DAMPING \

(cosmic\_importance \* 2.5 + (1.0 - cosmic\_importance) \* 1.7)

/\* Smallest amount of influence that we care about distributing. \*/

#define INFLUENCE\_CUTOFF 0.02

/\* Value in delta\_territory\_cache indicating that the value has not

\* been computed. Arbitrary but unattainable.

\*/

#define NOT\_COMPUTED (-2.0 \* MAX\_BOARD \* MAX\_BOARD)

/\* Maximum number of regions allowed between territory, moyo, and area.

\* FIXME: This number is vastly exaggerated. Should be possible to

\* come up with a much better upper bound.

\*/

#define MAX\_REGIONS (3\*MAX\_BOARD\*MAX\_BOARD + 1)

#define MAX\_INTRUSIONS (2 \* MAX\_BOARD \* MAX\_BOARD)

struct intrusion\_data

{

int source\_pos; /\* Stone from which intrusion originates.\*/

int strength\_pos; /\* Position of the intrusion influence soure. \*/

float strength;

float attenuation;

};

struct influence\_data

{

signed char safe[BOARDMAX];

float white\_influence[BOARDMAX]; /\* Accumulated influence. \*/

float black\_influence[BOARDMAX]; /\* Accumulated influence. \*/

float white\_strength[BOARDMAX]; /\* Strength of influence source. \*/

float black\_strength[BOARDMAX]; /\* Strength of influence source. \*/

float white\_attenuation[BOARDMAX];

float black\_attenuation[BOARDMAX];

float white\_permeability[BOARDMAX];

float black\_permeability[BOARDMAX];

int is\_territorial\_influence; /\* 0 only if computing escape\_influence.\*/

float territory\_value[BOARDMAX];

int non\_territory[BOARDMAX];

int captured;

int color\_to\_move; /\* Which color is in turn to move. \*/

int queue[MAX\_BOARD \* MAX\_BOARD]; /\* Points receiving influence. \*/

int intrusion\_counter;

struct intrusion\_data intrusions[MAX\_INTRUSIONS];

int id;

};

/\* Typedef for pointer to either of the functions whose\_territory(),

\* whose\_loose\_territory(), whose\_moyo(), and whose\_area().

\*/

typedef int (\*owner\_function\_ptr)(const struct influence\_data \*q, int pos);

/\* Used for tuning game advancement algorythm \*/

#define WEIGHT\_TERRITORY 10

#define WEIGHT\_MOYO 3

#define WEIGHT\_AREA 1

/\* cosmic\_importance is a number between 0.0 and 1.0 ;

\* when cosmic\_importance is 0.0, the default influence

\* values are used; when cosmic\_importance is 1.0, GNU Go

\* will try to play an influence-oriented fuseki by

\* over-estimatingthe potential territory values of moyos.

\* In the current implementation, cosmic\_importance decreases

\* slowly for 19\*19 games from 1.0 at move 4 to 0.0 at move 120.

\*/

float cosmic\_importance;

/\* Used in the whose\_moyo() function \*/

struct moyo\_determination\_data

{

float influence\_balance;

float my\_influence\_minimum;

float opp\_influence\_maximum;

};

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_LIBERTY\_H\_

#define \_LIBERTY\_H\_

#include "board.h"

#include "hash.h"

#include "gnugo.h"

#include "winsocket.h"

/\* ================================================================ \*/

/\* public variables \*/

/\* ================================================================ \*/

/\* ================================================================ \*/

#define FALSE\_EYE 1

#define HALF\_EYE 2

#define REVERSE\_RESULT(result) (WIN - result)

void start\_timer(int n);

double time\_report(int n, const char \*occupation, int move, double mintime);

void showstats(void);

void clearstats(void);

void transformation\_init(void);

void ascii\_report\_worm(char \*string);

void report\_dragon(FILE \*outfile, int pos);

void ascii\_report\_dragon(char \*string);

struct dragon\_data2 \*dragon2\_func(int pos);

/\* Routine names used by persistent and non-persistent caching schemes. \*/

enum routine\_id {

OWL\_ATTACK,

OWL\_DEFEND,

SEMEAI,

FIND\_DEFENSE,

ATTACK,

CONNECT,

DISCONNECT,

BREAK\_IN,

BLOCK\_OFF,

OWL\_THREATEN\_ATTACK,

OWL\_THREATEN\_DEFENSE,

OWL\_DOES\_DEFEND,

OWL\_DOES\_ATTACK,

OWL\_CONNECTION\_DEFENDS,

OWL\_SUBSTANTIAL,

OWL\_CONFIRM\_SAFETY,

ANALYZE\_SEMEAI,

NUM\_CACHE\_ROUTINES

};

#define ROUTINE\_NAMES \

"owl\_attack", \

"owl\_defend", \

"semeai", \

"find\_defense", \

"attack", \

"connect", \

"disconnect", \

"break\_in", \

"block\_off", \

"owl\_threaten\_attack", \

"owl\_threatend\_defense", \

"owl\_does\_defend", \

"owl\_does\_attack", \

"owl\_connection\_defends", \

"owl\_substantial", \

"owl\_confirm\_safety", \

"analyze\_semeai"

/\* To prioritize between different types of reading, we give a cost

\* ranking to each of the routines above:

\*

\* 4 semeai

\* 3 owl

\* 2 break-in

\* 1 connection

\* 0 tactical reading

\*

\* -1 is left at the end for a consistency check.

\*/

#define ROUTINE\_COSTS \

3, 3, 4, 0, 0, 1, 1, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, -1

const char \*routine\_id\_to\_string(enum routine\_id routine);

/\* This is used for both the dragon status and safety fields.

\* Also used for unconditional status in struct worm\_data and for the

\* final status computed by the aftermath code.

\*/

enum dragon\_status {

DEAD,

ALIVE,

CRITICAL,

UNKNOWN,

UNCHECKED,

CAN\_THREATEN\_ATTACK,

CAN\_THREATEN\_DEFENSE,

INESSENTIAL,

TACTICALLY\_DEAD,

ALIVE\_IN\_SEKI,

STRONGLY\_ALIVE,

INVINCIBLE,

INSUBSTANTIAL,

WHITE\_TERRITORY,

BLACK\_TERRITORY,

DAME,

NUM\_DRAGON\_STATUS

};

#define DRAGON\_STATUS\_NAMES \

"dead", \

"alive", \

"critical", \

"unknown", \

"unchecked", \

"can threaten attack", \

"can threaten defense", \

"inessential", \

"tactically dead", \

"alive in seki", \

"strongly alive", \

"invincible", \

"insubstantial", \

"white\_territory", \

"black\_territory", \

"dame"

const char \*status\_to\_string(enum dragon\_status status);

/\* Forward struct declarations. \*/

struct pattern;

struct pattern\_db;

struct fullboard\_pattern;

struct corner\_pattern;

struct corner\_db;

struct half\_eye\_data;

struct movelist;

/\*

\* Try to match a pattern in the database to the board. Callbacks for

\* each match.

\*/

typedef void (\*matchpat\_callback\_fn\_ptr)(int anchor, int color,

struct pattern \*, int rotation,

void \*data);

typedef void (\*fullboard\_matchpat\_callback\_fn\_ptr)(int move,

struct fullboard\_pattern \*,

int rotation);

typedef void (\*corner\_matchpat\_callback\_fn\_ptr)(int move, int color,

struct corner\_pattern \*pattern,

int trans,

int \*stones, int num\_stones);

void matchpat(matchpat\_callback\_fn\_ptr callback, int color,

struct pattern\_db \*pdb, void \*callback\_data,

signed char goal[BOARDMAX]);

void matchpat\_goal\_anchor(matchpat\_callback\_fn\_ptr callback, int color,

struct pattern\_db \*pdb, void \*callback\_data,

signed char goal[BOARDMAX], int anchor\_in\_goal);

void fullboard\_matchpat(fullboard\_matchpat\_callback\_fn\_ptr callback,

int color, struct fullboard\_pattern \*pattern);

void corner\_matchpat(corner\_matchpat\_callback\_fn\_ptr callback, int color,

struct corner\_db \*database);

void dfa\_match\_init(void);

void reading\_cache\_init(int bytes);

void reading\_cache\_clear(void);

float reading\_cache\_default\_size(void);

/\* reading.c \*/

int attack(int str, int \*move);

int find\_defense(int str, int \*move);

int attack\_and\_defend(int str,

int \*attack\_code, int \*attack\_point,

int \*defend\_code, int \*defense\_point);

int attack\_either(int astr, int bstr);

int defend\_both(int astr, int bstr);

int break\_through(int apos, int bpos, int cpos);

int attack\_threats(int pos, int max\_points, int moves[], int codes[]);

int restricted\_defend1(int str, int \*move,

int num\_forbidden\_moves, int \*forbidden\_moves);

int restricted\_attack2(int str, int \*move,

int num\_forbidden\_moves, int \*forbidden\_moves);

int simple\_ladder(int str, int \*move);

#define MOVE\_ORDERING\_PARAMETERS 67

void tune\_move\_ordering(int params[MOVE\_ORDERING\_PARAMETERS]);

void draw\_reading\_shadow(void);

/\* persistent.c \*/

void persistent\_cache\_init(void);

void purge\_persistent\_caches(void);

void clear\_persistent\_caches(void);

int search\_persistent\_reading\_cache(enum routine\_id routine, int str,

int \*result, int \*move);

void store\_persistent\_reading\_cache(enum routine\_id routine, int str,

int result, int move, int nodes);

void reading\_hotspots(float values[BOARDMAX]);

int search\_persistent\_connection\_cache(enum routine\_id routine,

int str1, int str2,

int \*result, int \*move);

void store\_persistent\_connection\_cache(enum routine\_id routine,

int str1, int str2,

int result, int move,

int tactical\_nodes,

signed char connection\_shadow[BOARDMAX]);

int search\_persistent\_breakin\_cache(enum routine\_id routine,

int str, Hash\_data \*goal\_hash,

int breakin\_node\_limit,

int \*result, int \*move);

void store\_persistent\_breakin\_cache(enum routine\_id routine,

int str, Hash\_data \*goal\_hash,

int result, int move,

int tactical\_nodes,

int breakin\_node\_limit,

signed char breakin\_shadow[BOARDMAX]);

int search\_persistent\_owl\_cache(enum routine\_id routine,

int apos, int bpos, int cpos,

int \*result, int \*move, int \*move2,

int \*certain);

void store\_persistent\_owl\_cache(enum routine\_id routine,

int apos, int bpos, int cpos,

int result, int move, int move2, int certain,

int tactical\_nodes, signed char goal[BOARDMAX],

int goal\_color);

void owl\_hotspots(float values[BOARDMAX]);

int search\_persistent\_semeai\_cache(enum routine\_id routine,

int apos, int bpos, int cpos, int color,

Hash\_data \*goal\_hash,

int \*resulta, int \*resultb,

int \*move, int \*certain);

void store\_persistent\_semeai\_cache(enum routine\_id routine,

int apos, int bpos, int cpos, int color,

Hash\_data \*goal\_hash,

int resulta, int resultb,

int move, int certain, int tactical\_nodes,

signed char goala[BOARDMAX],

signed char goalb[BOARDMAX]);

/\* readconnect.c \*/

int string\_connect(int str1, int str2, int \*move);

int disconnect(int str1, int str2, int \*move);

int fast\_disconnect(int str1, int str2, int \*move);

int non\_transitivity(int str1, int str2, int str3, int \*move);

int break\_in(int str, const signed char goal[BOARDMAX], int \*move);

int block\_off(int str1, const signed char goal[BOARDMAX], int \*move);

int obvious\_false\_eye(int pos, int color);

void estimate\_lunch\_eye\_value(int lunch, int \*min, int \*probable, int \*max,

int appreciate\_one\_two\_lunches);

int owl\_topological\_eye(int pos, int color);

int vital\_chain(int pos);

int confirm\_safety(int move, int color, int \*defense\_point,

signed char safe\_stones[BOARDMAX]);

int dragon\_weak(int pos);

float dragon\_weakness(int pos, int ignore\_dead\_dragons);

int size\_of\_biggest\_critical\_dragon(void);

void change\_dragon\_status(int dr, enum dragon\_status status);

float blunder\_size(int move, int color, int \*defense\_point,

signed char safe\_stones[BOARDMAX]);

void set\_depth\_values(int level, int report\_levels);

void modify\_depth\_values(int n);

void increase\_depth\_values(void);

void decrease\_depth\_values(void);

int get\_depth\_modification(void);

int safe\_move(int move, int color);

int does\_secure(int color, int move, int pos);

void compute\_new\_dragons(int dragon\_origins[BOARDMAX]);

void join\_dragons(int d1, int d2);

int dragon\_escape(signed char goal[BOARDMAX], int color,

signed char escape\_value[BOARDMAX]);

void compute\_refined\_dragon\_weaknesses(void);

void compute\_strategic\_sizes(void);

struct eyevalue;

void compute\_dragon\_genus(int d, struct eyevalue \*genus, int eye\_to\_exclude);

float crude\_dragon\_weakness(int safety, struct eyevalue \*genus, int has\_lunch,

float moyo\_value, float escape\_route);

int is\_same\_dragon(int d1, int d2);

int are\_neighbor\_dragons(int d1, int d2);

void mark\_dragon(int pos, signed char mx[BOARDMAX], signed char mark);

int first\_worm\_in\_dragon(int d);

int next\_worm\_in\_dragon(int w);

int lively\_dragon\_exists(int color);

void compute\_dragon\_influence(void);

void set\_strength\_data(int color, signed char safe\_stones[BOARDMAX],

float strength[BOARDMAX]);

void mark\_inessential\_stones(int color, signed char safe\_stones[BOARDMAX]);

void add\_cut(int apos, int bpos, int move);

void cut\_reasons(int color);

void get\_lively\_stones(int color, signed char safe\_stones[BOARDMAX]);

int is\_same\_worm(int w1, int w2);

int is\_worm\_origin(int w, int pos);

void propagate\_worm(int pos);

void find\_cuts(void);

void find\_connections(void);

/\* movelist.c \*/

int movelist\_move\_known(int move, int max\_points, int points[], int codes[]);

void movelist\_change\_point(int move, int code, int max\_points,

int points[], int codes[]);

/\* surround.c \*/

int compute\_surroundings(int pos, int apos, int showboard,

int \*surround\_size);

int is\_surrounded(int pos);

int does\_surround(int move, int dragon);

void reset\_surround\_data(void);

int surround\_map(int dr, int pos);

/\* functions to add (or remove) move reasons \*/

void collect\_move\_reasons(int color);

void clear\_move\_reasons(void);

void add\_lunch(int eater, int food);

void add\_attack\_move(int pos, int ww, int code);

void add\_defense\_move(int pos, int ww, int code);

void add\_attack\_threat\_move(int pos, int ww, int code);

void remove\_attack\_threat\_move(int pos, int ww);

void add\_defense\_threat\_move(int pos, int ww, int code);

void add\_connection\_move(int pos, int dr1, int dr2);

void add\_cut\_move(int pos, int dr1, int dr2);

void add\_antisuji\_move(int pos);

void add\_semeai\_move(int pos, int dr);

void add\_potential\_semeai\_attack(int pos, int dr1, int dr2);

void add\_potential\_semeai\_defense(int pos, int dr1, int dr2);

void add\_semeai\_threat(int pos, int dr);

void add\_owl\_attack\_move(int pos, int dr, int kworm, int code);

void add\_owl\_defense\_move(int pos, int dr, int code);

void add\_owl\_attack\_threat\_move(int pos, int dr, int code);

void add\_owl\_defense\_threat\_move(int pos, int dr, int code);

void add\_owl\_prevent\_threat\_move(int pos, int dr);

void add\_owl\_uncertain\_defense\_move(int pos, int dr);

void add\_owl\_uncertain\_attack\_move(int pos, int dr);

void add\_gain\_move(int pos, int target1, int target2);

void add\_loss\_move(int pos, int target1, int target2);

void add\_my\_atari\_atari\_move(int pos, int size);

void add\_your\_atari\_atari\_move(int pos, int size);

void add\_vital\_eye\_move(int pos, int eyespace, int color);

void add\_invasion\_move(int pos);

void add\_expand\_territory\_move(int pos);

void add\_expand\_moyo\_move(int pos);

void add\_strategical\_attack\_move(int pos, int dr);

void add\_strategical\_defense\_move(int pos, int dr);

void add\_worthwhile\_threat\_move(int pos);

void add\_replacement\_move(int from, int to, int color);

/\* Parameters to add\_either\_move and add\_all\_move \*/

#define ATTACK\_STRING 1

#define DEFEND\_STRING 2

void add\_either\_move(int pos, int reason1, int target1,

int reason2, int target2);

void add\_all\_move(int pos, int reason1, int target1,

int reason2, int target2);

int set\_minimum\_move\_value(int pos, float value);

void set\_maximum\_move\_value(int pos, float value);

void set\_minimum\_territorial\_value(int pos, float value);

void set\_maximum\_territorial\_value(int pos, float value);

void add\_shape\_value(int pos, float value);

void add\_followup\_value(int pos, float value);

void add\_reverse\_followup\_value(int pos, float value);

int list\_move\_reasons(FILE \*out, int pos);

void print\_all\_move\_values(FILE \*output);

void record\_top\_move(int move, float val);

void remove\_top\_move(int move);

void scale\_randomness(int pos, float scaling);

void compute\_move\_probabilities(float probabilities[BOARDMAX]);

void register\_good\_attack\_threat(int move, int target);

int is\_known\_good\_attack\_threat(int move, int target);

void register\_known\_safe\_move(int move);

int is\_known\_safe\_move(int move);

int get\_attack\_threats(int pos, int max\_strings, int strings[]);

int get\_defense\_threats(int pos, int max\_strings, int strings[]);

void get\_saved\_worms(int pos, signed char saved[BOARDMAX]);

void get\_saved\_dragons(int pos, signed char saved[BOARDMAX]);

void mark\_safe\_stones(int color, int move\_pos,

const signed char saved\_dragons[BOARDMAX],

const signed char saved\_worms[BOARDMAX],

signed char safe\_stones[BOARDMAX]);

int owl\_lively(int pos);

int owl\_escape\_value(int pos);

int owl\_goal\_dragon(int pos);

int owl\_eyespace(int pos);

int owl\_big\_eyespace(int pos);

int owl\_mineye(int pos);

int owl\_maxeye(int pos);

int owl\_proper\_eye(int pos);

int owl\_eye\_size(int pos);

int owl\_lunch(int str);

int owl\_strong\_dragon(int pos);

void owl\_reasons(int color);

void unconditional\_life(int unconditional\_territory[BOARDMAX], int color);

void clear\_unconditionally\_meaningless\_moves(void);

void find\_unconditionally\_meaningless\_moves(int unconditional\_territory[BOARDMAX],

int color);

int unconditionally\_meaningless\_move(int pos, int color,

int \*replacement\_move);

void unconditional\_move\_reasons(int color);

void find\_superstring(int str, int \*num\_stones, int \*stones);

void find\_superstring\_conservative(int str, int \*num\_stones, int \*stones);

void find\_superstring\_liberties(int str, int \*liberties, int \*libs,

int liberty\_cap);

void find\_proper\_superstring\_liberties(int str, int \*liberties, int \*libs,

int liberty\_cap);

void find\_superstring\_stones\_and\_liberties(int str, int \*num\_stones,

int \*stones, int \*liberties,

int \*libs, int liberty\_cap);

void superstring\_chainlinks(int str, int \*num\_adj, int adj[MAXCHAIN],

int liberty\_cap);

void proper\_superstring\_chainlinks(int str, int \*num\_adj,

int adj[MAXCHAIN], int liberty\_cap);

int place\_fixed\_handicap(int handicap); /\* place stones on board only \*/

int place\_free\_handicap(int handicap); /\* place stones on board only \*/

int free\_handicap\_remaining\_stones(void);

int free\_handicap\_total\_stones(void);

/\* Various different strategies for finding a move \*/

void fuseki(int color);

void semeai(void);

void semeai\_move\_reasons(int color);

void shapes(int color);

void endgame(int color);

void endgame\_shapes(int color);

void combinations(int color);

int atari\_atari(int color, int \*attack\_move,

signed char defense\_moves[BOARDMAX],

int save\_verbose);

int atari\_atari\_confirm\_safety(int color, int tpos, int \*move, int minsize,

const signed char saved\_dragons[BOARDMAX],

const signed char saved\_worms[BOARDMAX]);

int atari\_atari\_blunder\_size(int color, int tpos,

signed char defense\_moves[BOARDMAX],

const signed char safe\_stones[BOARDMAX]);

int review\_move\_reasons(int \*move, float \*value, int color,

float pure\_threat\_value, float our\_score,

int allowed\_moves[BOARDMAX],

int use\_thrashing\_dragon\_heuristics);

void prepare\_move\_influence\_debugging(int pos, int color);

int fill\_liberty(int \*move, int color);

int aftermath\_genmove(int color, int do\_capture\_dead\_stones,

int allowed\_moves[BOARDMAX]);

enum dragon\_status aftermath\_final\_status(int color, int pos);

int mc\_get\_size\_of\_pattern\_values\_table(void);

int mc\_load\_patterns\_from\_db(const char \*filename, unsigned int \*values);

void mc\_init\_patterns(const unsigned int \*values);

int choose\_mc\_patterns(char \*name);

void list\_mc\_patterns(void);

void uct\_genmove(int color, int \*move, int \*forbidden\_moves,

int \*allowed\_moves, int nodes, float \*move\_values,

int \*move\_frequencies);

int owl\_attack(int target, int \*attack\_point, int \*certain, int \*kworm);

int owl\_defend(int target, int \*defense\_point, int \*certain, int \*kworm);

int owl\_threaten\_attack(int target, int \*attack1, int \*attack2);

int owl\_threaten\_defense(int target, int \*defend1, int \*defend2);

int owl\_does\_defend(int move, int target, int \*kworm);

int owl\_confirm\_safety(int move, int target, int \*defense\_point, int \*kworm);

int owl\_does\_attack(int move, int target, int \*kworm);

int owl\_connection\_defends(int move, int target1, int target2);

int owl\_substantial(int str);

void owl\_analyze\_semeai(int apos, int bpos,

int \*resulta, int \*resultb, int \*semeai\_move,

int owl, int \*semeai\_result\_certain);

void owl\_analyze\_semeai\_after\_move(int move, int color, int apos, int bpos,

int \*resulta, int \*resultb,

int \*semeai\_move, int owl,

int \*semeai\_result\_certain,

int recompute\_dragons);

void set\_limit\_search(int value);

void set\_search\_diamond(int pos);

void reset\_search\_mask(void);

void set\_search\_mask(int pos, int value);

int oracle\_play\_move(int pos, int color);

void consult\_oracle(int color);

void summon\_oracle(void);

void oracle\_loadsgf(char \*infilename, char \*untilstring);

int oracle\_threatens(int move, int target);

int within\_search\_area(int pos);

int metamachine\_genmove(int color, float \*value);

void draw\_search\_area(void);

int genmove\_restricted(int color, int allowed\_moves[BOARDMAX]);

void change\_attack(int str, int move, int acode);

void change\_defense(int str, int move, int dcode);

void change\_attack\_threat(int str, int move, int acode);

void change\_defense\_threat(int str, int move, int dcode);

int attack\_move\_known(int move, int str);

int defense\_move\_known(int move, int str);

int attack\_threat\_move\_known(int move, int str);

int defense\_threat\_move\_known(int move, int str);

void worm\_reasons(int color);

int semeai\_move\_reason\_known(int move, int dr);

int does\_attack(int move, int str);

int does\_defend(int move, int str);

int double\_atari(int move, int color, float \*value,

signed char safe\_stones[BOARDMAX]);

int playing\_into\_snapback(int move, int color);

int play\_attack\_defend\_n(int color, int do\_attack, int num\_moves, ...);

int play\_attack\_defend2\_n(int color, int do\_attack, int num\_moves, ...);

int play\_break\_through\_n(int color, int num\_moves, ...);

int play\_connect\_n(int color, int do\_connect, int num\_moves, ...);

int play\_lib\_n(int color, int num\_moves, ...);

int cut\_possible(int pos, int color);

int defend\_against(int move, int color, int apos);

int somewhere(int color, int check\_alive, int num\_moves, ...);

int visible\_along\_edge(int color, int apos, int bpos);

int test\_symmetry\_after\_move(int move, int color, int strict);

/\* Printmoyo values, specified by -m flag. \*/

#define PRINTMOYO\_TERRITORY 0x01

#define PRINTMOYO\_MOYO 0x02

#define PRINTMOYO\_AREA 0x04

/\* The following have been borrowed by the influence functions below. \*/

#define PRINTMOYO\_INITIAL\_INFLUENCE 0x08

#define PRINTMOYO\_PRINT\_INFLUENCE 0x10

#define PRINTMOYO\_NUMERIC\_INFLUENCE 0x20

#define PRINTMOYO\_PERMEABILITY 0x40

#define PRINTMOYO\_STRENGTH 0x80

#define PRINTMOYO\_ATTENUATION 0x100

#define PRINTMOYO\_VALUE\_TERRITORY 0x200

/\* These values are used to communicate whether stones are safe or

\* have been saved, when computing influence.

\*/

#define INFLUENCE\_SAFE\_STONE 1

#define INFLUENCE\_SAVED\_STONE 2

/\* These values are used to communicate the status of stones when analyzing

\* a move for potentially being a blunder.

\*/

/\* dead 0 \*/

#define SAFE\_STONE 1

#define OWL\_SAVED\_STONE 2

/\* This format is used when exporting the moyo segmentation. \*/

#define MAX\_MOYOS MAX\_BOARD\*MAX\_BOARD

struct moyo\_data

{

int number; /\* Number of moyos. \*/

int segmentation[BOARDMAX]; /\* Numbers the moyos. \*/

int size[MAX\_MOYOS];

int owner[MAX\_MOYOS];

float territorial\_value[MAX\_MOYOS];

};

/\* We use a forward declaration of influence\_data so that the rest

\* of the engine can reference influence data. It can only be accessed

\* in influence.c, however!

\*/

struct influence\_data;

extern struct influence\_data initial\_black\_influence;

extern struct influence\_data initial\_white\_influence;

extern struct influence\_data move\_influence;

extern struct influence\_data followup\_influence;

#define INITIAL\_INFLUENCE(color) ((color) == WHITE ? \

&initial\_white\_influence \

: &initial\_black\_influence)

#define OPPOSITE\_INFLUENCE(color) (INITIAL\_INFLUENCE(OTHER\_COLOR(color)))

#define DEFAULT\_STRENGTH 100.0

/\* Influence functions. \*/

void compute\_influence(int color, const signed char safe\_stones[BOARDMAX],

const float strength[BOARDMAX],

struct influence\_data \*q,

int move, const char \*trace\_message);

void compute\_followup\_influence(const struct influence\_data \*base,

struct influence\_data \*q,

int move, const char \*trace\_message);

void compute\_escape\_influence(int color,

const signed char safe\_stones[BOARDMAX],

const signed char goal[BOARDMAX],

const float strength[BOARDMAX],

signed char escape\_value[BOARDMAX]);

float influence\_delta\_territory(const struct influence\_data \*base,

const struct influence\_data \*q, int color,

int move);

int retrieve\_delta\_territory\_cache(int pos, int color, float \*move\_value,

float \*followup\_value,

const struct influence\_data \*base,

Hash\_data safety\_hash);

void store\_delta\_territory\_cache(int pos, int color, float move\_value,

float followup\_value,

const struct influence\_data \*base,

Hash\_data safety\_hash);

int whose\_territory(const struct influence\_data \*q, int pos);

int whose\_moyo(const struct influence\_data \*q, int pos);

int whose\_moyo\_restricted(const struct influence\_data \*q, int pos);

int whose\_area(const struct influence\_data \*q, int pos);

float influence\_territory(const struct influence\_data \*q, int pos, int color);

void influence\_get\_territory\_segmentation(struct influence\_data \*q,

struct moyo\_data \*moyo);

void get\_influence(const struct influence\_data \*q,

float white\_influence[BOARDMAX],

float black\_influence[BOARDMAX],

float white\_strength[BOARDMAX],

float black\_strength[BOARDMAX],

float white\_attenuation[BOARDMAX],

float black\_attenuation[BOARDMAX],

float white\_permeability[BOARDMAX],

float black\_permeability[BOARDMAX],

float territory\_value[BOARDMAX],

int influence\_regions[BOARDMAX],

int non\_territory[BOARDMAX]);

float influence\_score(const struct influence\_data \*q, int chinese\_rules);

float game\_status(int color);

void influence\_mark\_non\_territory(int pos, int color);

int influence\_considered\_lively(const struct influence\_data \*q, int pos);

void influence\_erase\_territory(struct influence\_data \*q, int pos, int color);

void break\_territories(int color\_to\_move, struct influence\_data \*q,

int store, int pos);

void clear\_break\_in\_list(void);

void break\_in\_move\_reasons(int color);

void choose\_strategy(int color, float our\_score, float game\_status);

/\* Eye space functions. \*/

int is\_eye\_space(int pos);

int is\_proper\_eye\_space(int pos);

int is\_marginal\_eye\_space(int pos);

int max\_eye\_value(int pos);

void test\_eyeshape(int eyesize, int \*eye\_vertices);

int analyze\_eyegraph(const char \*coded\_eyegraph, struct eyevalue \*value,

char \*analyzed\_eyegraph);

/\* debugging support \*/

void goaldump(const signed char goal[BOARDMAX]);

void move\_considered(int move, float value);

/\* Transformation stuff. \*/

#define MAX\_OFFSET (2\*MAX\_BOARD - 1) \* (2\*MAX\_BOARD - 1)

#define OFFSET(dx, dy)\

((dy + MAX\_BOARD - 1) \* (2\*MAX\_BOARD - 1) + (dx + MAX\_BOARD - 1))

#define OFFSET\_DELTA(dx, dy) (OFFSET(dx, dy) - OFFSET(0, 0))

#define CENTER\_OFFSET(offset) (offset - OFFSET(0, 0))

#define TRANSFORM(offset, trans) (transformation[offset][trans])

#define AFFINE\_TRANSFORM(offset, trans, delta)\

(transformation[offset][trans] + delta)

#define TRANSFORM2(x, y, tx, ty, trans)\

do {\

\*tx = transformation2[trans][0][0] \* (x) + transformation2[trans][0][1] \* (y);\

\*ty = transformation2[trans][1][0] \* (x) + transformation2[trans][1][1] \* (y);\

} while (0)

/\* ================================================================ \*/

/\* global variables \*/

/\* ================================================================ \*/

extern int disable\_threat\_computation;

extern int disable\_endgame\_patterns;

extern int doing\_scoring;

/\* Reading parameters \*/

extern int depth; /\* deep reading cutoff \*/

extern int backfill\_depth; /\* deep reading cutoff \*/

extern int backfill2\_depth; /\* deep reading cutoff \*/

extern int break\_chain\_depth; /\* deep reading cutoff \*/

extern int superstring\_depth; /\* deep reading cutoff \*/

extern int branch\_depth; /\* deep reading cutoff \*/

extern int fourlib\_depth; /\* deep reading cutoff \*/

extern int ko\_depth; /\* deep ko reading cutoff \*/

extern int aa\_depth; /\* deep global reading cutoff \*/

extern int depth\_offset; /\* keeps track of temporary depth changes \*/

extern int owl\_distrust\_depth; /\* below this owl trusts the optics code \*/

extern int owl\_branch\_depth; /\* below this owl tries only one variation \*/

extern int owl\_reading\_depth; /\* owl does not read below this depth \*/

extern int owl\_node\_limit; /\* maximum number of nodes considered \*/

extern int semeai\_branch\_depth;

extern int semeai\_branch\_depth2;

extern int semeai\_node\_limit;

extern int connect\_depth;

extern int connect\_depth2;

extern int connection\_node\_limit;

extern int breakin\_depth;

extern int breakin\_node\_limit;

extern int semeai\_variations; /\* max variations considered reading semeai \*/

extern float best\_move\_values[10];

extern int best\_moves[10];

extern int experimental\_owl\_ext; /\* use experimental owl (GAIN/LOSS) \*/

extern int experimental\_semeai; /\* use experimental semeai module \*/

extern int experimental\_connections; /\* use experimental connection module \*/

extern int alternate\_connections; /\* use alternate connection module \*/

extern int owl\_threats; /\* compute owl threats \*/

extern int experimental\_break\_in; /\* use experimental module breakin.c \*/

extern int cosmic\_gnugo; /\* use center oriented influence \*/

extern int large\_scale; /\* seek large scale captures \*/

extern int thrashing\_dragon; /\* Dead opponent's dragon trying to live \*/

extern signed char thrashing\_stone[BOARDMAX]; /\* All thrashing stones. \*/

extern int transformation[MAX\_OFFSET][8];

extern const int transformation2[8][2][2];

/\* Arrays pointing out the closest worms from each vertex. The first

\* one is the closest worms of either color, the last two ones ignore

\* worms of the other color. Beyond a certain distance from any worm

\* no close worm is listed at all. Only the closest worm is listed

\* and if more than one are equally close they are all listed. The

\* number of equally close worms is given in the number\_\*\_worms

\* arrays. If more than MAX\_CLOSE\_WORMS are equally close, none is

\* listed.

\*

\* See compute\_effective\_worm\_sizes() in worm.c for details.

\*/

#define MAX\_CLOSE\_WORMS 4

extern int close\_worms[BOARDMAX][MAX\_CLOSE\_WORMS];

extern int number\_close\_worms[BOARDMAX];

extern int close\_black\_worms[BOARDMAX][MAX\_CLOSE\_WORMS];

extern int number\_close\_black\_worms[BOARDMAX];

extern int close\_white\_worms[BOARDMAX][MAX\_CLOSE\_WORMS];

extern int number\_close\_white\_worms[BOARDMAX];

extern int false\_eye\_territory[BOARDMAX];

extern int forced\_backfilling\_moves[BOARDMAX];

extern double slowest\_time; /\* Timing statistics \*/

extern int slowest\_move;

extern int slowest\_movenum;

extern double total\_time;

struct eyevalue {

unsigned char a; /\* number of eyes if attacker plays first twice \*/

unsigned char b; /\* number of eyes if attacker plays first \*/

unsigned char c; /\* number of eyes if defender plays first \*/

unsigned char d; /\* number of eyes if defender plays first twice \*/

};

struct half\_eye\_data {

float value; /\* Topological eye value. \*/

unsigned char type; /\* HALF\_EYE or FALSE\_EYE; \*/

int num\_attacks; /\* number of attacking points \*/

int attack\_point[4]; /\* the moves to attack a topological halfeye \*/

int num\_defenses; /\* number of defending points \*/

int defense\_point[4]; /\* the moves to defend a topological halfeye \*/

};

/\* array of half-eye data \*/

extern struct half\_eye\_data half\_eye[BOARDMAX];

/\*

\* data concerning a worm. A copy is kept at each vertex of the worm.

\*/

#define MAX\_TACTICAL\_POINTS 10

struct worm\_data {

int color; /\* its color \*/

int size; /\* its cardinality \*/

float effective\_size; /\* stones and surrounding spaces \*/

int origin; /\* the origin of the string. Two vertices are in \*/

/\* the same worm iff they have same origin. \*/

int liberties; /\* number of liberties \*/

int liberties2; /\* number of second order liberties \*/

int liberties3; /\* third order liberties (empty vertices at distance 3) \*/

int liberties4; /\* fourth order liberties \*/

int lunch; /\* if lunch != 0 then lunch points to a boundary \*/

/\* worm which can be captured easily. \*/

int cutstone; /\* 1=potential cutting stone; 2=cutting stone \*/

int cutstone2; /\* Number of potential cuts involving the worm. \*/

int genus; /\* number of connected components of the complement, less one \*/

int inessential; /\* 1=inessential worm \*/

int invincible; /\* 1=strongly unconditionally non-capturable \*/

enum dragon\_status unconditional\_status; /\* ALIVE, DEAD, WHITE\_TERRITORY,

BLACK\_TERRITORY, UNKNOWN \*/

/\* The following arrays keeps track of up to MAX\_TACTICAL\_POINTS

\* different attack, defense, attack threat, and defense threat

\* points with corresponding result codes. (0 = loss, 1 = bad ko, 2

\* = good ko, 3 = win). The arrays are guaranteed to be sorted with

\* respect to the codes so that the first element contains the best

\* result.

\*/

int attack\_points[MAX\_TACTICAL\_POINTS];

int attack\_codes[MAX\_TACTICAL\_POINTS];

int defense\_points[MAX\_TACTICAL\_POINTS];

int defense\_codes[MAX\_TACTICAL\_POINTS];

int attack\_threat\_points[MAX\_TACTICAL\_POINTS];

int attack\_threat\_codes[MAX\_TACTICAL\_POINTS];

int defense\_threat\_points[MAX\_TACTICAL\_POINTS];

int defense\_threat\_codes[MAX\_TACTICAL\_POINTS];

};

extern struct worm\_data worm[BOARDMAX];

/\* Unconditionally meaningless moves. \*/

int meaningless\_black\_moves[BOARDMAX];

int meaningless\_white\_moves[BOARDMAX];

/\* Surround cache (see surround.c) \*/

#define MAX\_SURROUND 10

struct surround\_data {

int dragon\_number; /\* number of the (surrounded) beast \*/

signed char surround\_map[BOARDMAX]; /\* surround map \*/

};

extern struct surround\_data surroundings[MAX\_SURROUND];

extern int surround\_pointer;

/\*

\* data concerning a dragon. A copy is kept at each stone of the string.

\*/

struct dragon\_data {

int color; /\* its color \*/

int id; /\* the index into the dragon2 array \*/

int origin; /\* the origin of the dragon. Two vertices are in the same \*/

/\* dragon iff they have same origin. \*/

int size; /\* size of the dragon \*/

float effective\_size; /\* stones and surrounding spaces \*/

enum dragon\_status crude\_status; /\* (ALIVE, DEAD, UNKNOWN, CRITICAL) \*/

enum dragon\_status status; /\* best trusted status \*/

};

extern struct dragon\_data dragon[BOARDMAX];

/\* Supplementary data concerning a dragon. Only one copy is stored per

\* dragon in the dragon2 array.

\*/

#define MAX\_NEIGHBOR\_DRAGONS 10

struct dragon\_data2 {

int origin; /\* the origin of the dragon \*/

int adjacent[MAX\_NEIGHBOR\_DRAGONS]; /\* adjacent dragons \*/

int neighbors; /\* number of adjacent dragons \*/

int hostile\_neighbors; /\* neighbors of opposite color \*/

int moyo\_size; /\* size of surrounding influence moyo, \*/

float moyo\_territorial\_value; /\* ...and its territorial value \*/

enum dragon\_status safety; /\* a more detailed status estimate \*/

float weakness; /\* a continuous estimate of the dragon's safety \*/

float weakness\_pre\_owl; /\* dragon safety based on pre-owl computations \*/

float strategic\_size; /\* An effective size including weakness of neighbors \*/

int escape\_route; /\* a measurement of likelihood of escape \*/

struct eyevalue genus; /\* the number of eyes (approximately) \*/

int heye; /\* coordinates of a half eye \*/

int lunch; /\* if lunch != 0 then lunch points to a boundary worm which \*/

/\* can be captured easily. \*/

int surround\_status; /\* Is it surrounded? \*/

int surround\_size; /\* Size of the surrounding area \*/

int semeais; /\* number of semeais in which the dragon is involved \*/

int semeai\_defense\_code ;/\* Result code for semeai defense. \*/

int semeai\_defense\_point;/\* Move found by semeai code to rescue dragon \*/

int semeai\_defense\_certain;

int semeai\_defense\_target; /\* The opponent dragon involved in the semeai \*/

int semeai\_attack\_code ; /\* Result code for semeai attack. \*/

int semeai\_attack\_point; /\* Move found by semeai code to kill dragon \*/

int semeai\_attack\_certain;

int semeai\_attack\_target; /\* The opponent dragon involved in the semeai \*/

enum dragon\_status owl\_threat\_status; /\* CAN\_THREATEN\_ATTACK/DEFENSE \*/

enum dragon\_status owl\_status; /\* (ALIVE, DEAD, UNKNOWN, CRITICAL, UNCHECKED) \*/

int owl\_attack\_point; /\* vital point for attack \*/

int owl\_attack\_code; /\* ko result code \*/

int owl\_attack\_certain; /\* 0 if owl reading node limit is reached \*/

int owl\_attack\_node\_count;

int owl\_second\_attack\_point;/\* if attacker gets both attack points, wins \*/

int owl\_defense\_point; /\* vital point for defense \*/

int owl\_defense\_code; /\* ko result code \*/

int owl\_defense\_certain; /\* 0 if owl reading node limit is reached \*/

int owl\_second\_defense\_point;/\* if defender gets both attack points, wins \*/

int owl\_attack\_kworm; /\* only valid when owl\_attack\_code is GAIN \*/

int owl\_defense\_kworm; /\* only valid when owl\_defense\_code is LOSS \*/

};

/\* dragon2 is dynamically allocated \*/

extern int number\_of\_dragons;

extern struct dragon\_data2 \*dragon2;

/\* Macros for accessing the dragon2 data with board coordinates and

\* the dragon data with a dragon id.

\*/

#if 1 /\* Trust DRAGON2 accesses \*/

#define DRAGON2(pos) dragon2[dragon[pos].id]

#else

struct dragon\_data2 \*dragon2\_func(int pos);

#define DRAGON2(pos) (\*dragon2\_func(pos))

#endif

#define DRAGON(d) dragon[dragon2[d].origin]

extern float white\_score, black\_score;

/\* Global variables to tune strategy. \*/

extern float minimum\_value\_weight;

extern float maximum\_value\_weight;

extern float invasion\_malus\_weight;

extern float strategical\_weight;

extern float territorial\_weight;

extern float attack\_dragon\_weight;

extern float followup\_weight;

struct aftermath\_data {

int white\_captured;

int black\_captured;

int white\_prisoners;

int black\_prisoners;

int white\_territory;

int black\_territory;

int white\_area;

int black\_area;

int white\_control[BOARDMAX];

int black\_control[BOARDMAX];

enum dragon\_status final\_status[BOARDMAX];

};

#define MAX\_EYE\_ATTACKS 3

struct eye\_data {

int color; /\* BLACK, WHITE, or GRAY \*/

int esize; /\* size of the eyespace \*/

int msize; /\* number of marginal vertices \*/

int origin; /\* The origin \*/

struct eyevalue value; /\* Number of eyes. \*/

/\* The above fields are constant on the whole eyespace. \*/

/\* ---------------------------------------------------------------- \*/

/\* The below fields are not. \*/

unsigned char marginal; /\* This vertex is marginal \*/

unsigned char neighbors; /\* number of neighbors in eyespace \*/

unsigned char marginal\_neighbors; /\* number of marginal neighbors \*/

};

struct vital\_eye\_points {

int attack\_points[MAX\_EYE\_ATTACKS];

int defense\_points[MAX\_EYE\_ATTACKS];

};

extern struct vital\_eye\_points black\_vital\_points[BOARDMAX];

extern struct vital\_eye\_points white\_vital\_points[BOARDMAX];

extern struct eye\_data white\_eye[BOARDMAX];

extern struct eye\_data black\_eye[BOARDMAX];

/\* Array with the information which was previously stored in the cut

\* field and in the INHIBIT\_CONNECTION bit of the type field in struct

\* eye\_data.

\*/

extern int cutting\_points[BOARDMAX];

/\* The following declarations have to be postponed until after the

\* definition of struct eye\_data or struct half\_eye\_data.

\*/

void compute\_eyes(int pos, struct eyevalue \*value,

int \*attack\_point, int \*defense\_point,

struct eye\_data eye[BOARDMAX],

struct half\_eye\_data heye[BOARDMAX],

int add\_moves);

void compute\_eyes\_pessimistic(int pos, struct eyevalue \*value,

int \*pessimistic\_min,

int \*attack\_point, int \*defense\_point,

struct eye\_data eye[BOARDMAX],

struct half\_eye\_data heye[BOARDMAX]);

void propagate\_eye(int pos, struct eye\_data eye[BOARDMAX]);

int find\_eye\_dragons(int origin, struct eye\_data eye[BOARDMAX], int eye\_color,

int dragons[], int max\_dragons);

void make\_domains(struct eye\_data b\_eye[BOARDMAX],

struct eye\_data w\_eye[BOARDMAX],

int owl\_call);

void partition\_eyespaces(struct eye\_data eye[BOARDMAX], int color);

void find\_half\_and\_false\_eyes(int color, struct eye\_data eye[BOARDMAX],

struct half\_eye\_data heye[BOARDMAX],

int find\_mask[BOARDMAX]);

void set\_eyevalue(struct eyevalue \*e, int a, int b, int c, int d);

int min\_eye\_threat(struct eyevalue \*e);

int min\_eyes(struct eyevalue \*e);

int max\_eyes(struct eyevalue \*e);

int max\_eye\_threat(struct eyevalue \*e);

void add\_eyevalues(struct eyevalue \*e1, struct eyevalue \*e2,

struct eyevalue \*sum);

int eye\_move\_urgency(struct eyevalue \*e);

char \*eyevalue\_to\_string(struct eyevalue \*e);

int is\_halfeye(struct half\_eye\_data heye[BOARDMAX], int pos);

int is\_false\_eye(struct half\_eye\_data heye[BOARDMAX], int pos);

#endif /\* \_LIBERTY\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

/\* values for move\_reason.type \*/

#define THREAT\_BIT 1

/\* Only use even values for non-threat move reasons! \*/

#define ATTACK\_MOVE 2

#define ATTACK\_MOVE\_GOOD\_KO 4

#define ATTACK\_MOVE\_BAD\_KO 6

#define ATTACK\_THREAT (ATTACK\_MOVE | THREAT\_BIT)

#define DEFEND\_MOVE 8

#define DEFEND\_MOVE\_GOOD\_KO 10

#define DEFEND\_MOVE\_BAD\_KO 12

#define DEFEND\_THREAT (DEFEND\_MOVE | THREAT\_BIT)

#define CONNECT\_MOVE 14

#define CUT\_MOVE 16

#define SEMEAI\_MOVE 18

#define SEMEAI\_THREAT (SEMEAI\_MOVE | THREAT\_BIT)

#define EXPAND\_TERRITORY\_MOVE 20

#define EXPAND\_MOYO\_MOVE 22

#define INVASION\_MOVE 24

#define OWL\_ATTACK\_MOVE 26

#define OWL\_ATTACK\_MOVE\_GOOD\_KO 28

#define OWL\_ATTACK\_MOVE\_BAD\_KO 30

#define OWL\_ATTACK\_THREAT (OWL\_ATTACK\_MOVE | THREAT\_BIT)

#define OWL\_DEFEND\_MOVE 32

#define OWL\_DEFEND\_MOVE\_GOOD\_KO 34

#define OWL\_DEFEND\_MOVE\_BAD\_KO 36

#define OWL\_DEFEND\_THREAT (OWL\_DEFEND\_MOVE | THREAT\_BIT)

#define OWL\_PREVENT\_THREAT 38

#define UNCERTAIN\_OWL\_ATTACK 40

#define UNCERTAIN\_OWL\_DEFENSE 42

#define STRATEGIC\_ATTACK\_MOVE 44

#define STRATEGIC\_DEFEND\_MOVE 46

#define MY\_ATARI\_ATARI\_MOVE 50

#define YOUR\_ATARI\_ATARI\_MOVE 52

#define VITAL\_EYE\_MOVE 54

#define OWL\_ATTACK\_MOVE\_GAIN 60

#define OWL\_DEFEND\_MOVE\_LOSS 62

#define POTENTIAL\_SEMEAI\_ATTACK 64

#define POTENTIAL\_SEMEAI\_DEFENSE 66

#define ANTISUJI\_MOVE 70

#define EITHER\_MOVE 100

#define ALL\_MOVE 102

/\* Bitmap values for move\_reason.status \*/

#define ACTIVE 0

#define TERRITORY\_REDUNDANT 1

#define STRATEGICALLY\_REDUNDANT 2

#define REDUNDANT (TERRITORY\_REDUNDANT | STRATEGICALLY\_REDUNDANT)

#define SECONDARY 4

#define MAX\_REASONS 120

#define MAX\_TRACE\_LENGTH 160

#define HUGE\_MOVE\_VALUE 10.0\*MAX\_BOARD\*MAX\_BOARD

struct move\_reason {

int type; /\* e.g. attack, defend, or connect \*/

int what; /\* pointer into list of strings, list of pair of dragons,

or similar \*/

int status; /\* This is a bitmap to mark redundant or secondary

move reasons. \*/

};

struct move\_data {

float value; /\* total comparison value, computed at the very end \*/

float final\_value; /\* value after point redistribution. \*/

float additional\_ko\_value; /\* Additional threat value if ko fight going on.\*/

float territorial\_value; /\* Value in terms of actual profit. \*/

float strategical\_value; /\* Value with respect to strength, weakness, and

safety of all groups on the board. \*/

float maxpos\_shape; /\* Maximal positive contribution to shape \*/

float maxneg\_shape; /\* Maximal negative contribution to shape \*/

int numpos\_shape; /\* Number of positive contributions to shape \*/

int numneg\_shape; /\* Number of negative contributions to shape \*/

float followup\_value; /\* Value of followup move (our sente). \*/

float influence\_followup\_value; /\* Followup value of move as reported by

experimental influence. \*/

float reverse\_followup\_value; /\* Value of opponents followup move

(reverse sente). \*/

float secondary\_value; /\* Secondary move value. \*/

float min\_value; /\* Minimum allowed value for the move. \*/

float max\_value; /\* Maximum allowed value for the move. \*/

float min\_territory; /\* Minimum territorial value. \*/

float max\_territory; /\* Maximum territorial value. \*/

float randomness\_scaling; /\* Increase to randomize this move. \*/

int reason[MAX\_REASONS]; /\* List of reasons for a move. \*/

int move\_safety; /\* Whether the move seems safe. \*/

int worthwhile\_threat; /\* Play this move as a pure threat. \*/

float random\_number; /\* Random number connected to this move. \*/

};

/\*

\* Some sizes.

\*

\* FIXME: Many of these could be optimized more for size (e.g. MAX\_EYES)

\*/

#define MAX\_MOVE\_REASONS 1000

#define MAX\_WORMS 2\*MAX\_BOARD\*MAX\_BOARD/3

#define MAX\_DRAGONS MAX\_WORMS

#define MAX\_CONNECTIONS 4\*MAX\_WORMS

#define MAX\_POTENTIAL\_SEMEAI 50

#define MAX\_EYES MAX\_BOARD\*MAX\_BOARD/2

#define MAX\_LUNCHES MAX\_WORMS

#define MAX\_EITHER 100

#define MAX\_ALL 100

#define MAX\_ATTACK\_THREATS 6

extern struct move\_data move[BOARDMAX];

extern struct move\_reason move\_reasons[MAX\_MOVE\_REASONS];

extern int next\_reason;

/\* Connections \*/

extern int conn\_worm1[MAX\_CONNECTIONS];

extern int conn\_worm2[MAX\_CONNECTIONS];

extern int next\_connection;

extern int semeai\_target1[MAX\_POTENTIAL\_SEMEAI];

extern int semeai\_target2[MAX\_POTENTIAL\_SEMEAI];

/\* Unordered sets (currently pairs) of move reasons / targets \*/

typedef struct {

int reason1;

int what1;

int reason2;

int what2;

} Reason\_set;

extern Reason\_set either\_data[MAX\_EITHER];

extern int next\_either;

extern Reason\_set all\_data[MAX\_ALL];

extern int next\_all;

/\* Eye shapes \*/

extern int eyes[MAX\_EYES];

extern int eyecolor[MAX\_EYES];

extern int next\_eye;

/\* Lunches \*/

extern int lunch\_dragon[MAX\_LUNCHES]; /\* eater \*/

extern int lunch\_worm[MAX\_LUNCHES]; /\* food \*/

extern int next\_lunch;

/\* Point redistribution \*/

extern int replacement\_map[BOARDMAX];

/\* The color for which we are evaluating moves. \*/

extern int current\_color;

int find\_worm(int str);

int find\_dragon(int str);

int move\_reason\_known(int pos, int type, int what);

int attack\_move\_reason\_known(int pos, int what);

int defense\_move\_reason\_known(int pos, int what);

int owl\_attack\_move\_reason\_known(int pos, int what);

int owl\_defense\_move\_reason\_known(int pos, int what);

int owl\_move\_reason\_known(int pos, int what);

int semeai\_move\_reason\_known(int pos, int what);

int get\_biggest\_owl\_target(int pos);

int is\_antisuji\_move(int pos);

void discard\_redundant\_move\_reasons(int pos);

void mark\_changed\_dragon(int pos, int color, int affected, int affected2,

int move\_reason\_type,

signed char safe\_stones[BOARDMAX],

float strength[BOARDMAX], float \*effective\_size);

void mark\_changed\_string(int affected, signed char changed\_stones[BOARDMAX],

float strength[BOARDMAX], signed char new\_status);

int adjacent\_to\_nondead\_stone(int pos, int color);

int find\_connection(int worm1, int worm2);

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

struct heap\_entry;

struct connection\_data;

/\* Expensive functions that we try to evaluate as late as possible

\* when spreading connection distances.

\*/

typedef void (\*connection\_helper\_fn\_ptr) (struct connection\_data \*conn,

int color);

/\* This heap contains a list of positions where we have delayed a

\* decision whether to "spread a connection distance". The function

\* helper() will be called when we finally need the decision. See

\* push\_connection\_heap\_entry() for organization of the heap.

\*/

struct heap\_entry {

int distance;

int coming\_from;

int target;

connection\_helper\_fn\_ptr helper;

};

/\* Fixed-point arithmetic helper macros \*/

#define FIXED\_POINT\_BASIS 10000

#define FP(x) ((int) (0.5 + FIXED\_POINT\_BASIS \* (x)))

#define FIXED\_TO\_FLOAT(x) ((x) / (float) FIXED\_POINT\_BASIS)

#define HUGE\_CONNECTION\_DISTANCE FP(100.0)

struct connection\_data {

int distances[BOARDMAX];

int deltas[BOARDMAX];

int coming\_from[BOARDMAX];

int vulnerable1[BOARDMAX];

int vulnerable2[BOARDMAX];

int queue[BOARDMAX];

int queue\_start;

int queue\_end;

int heap\_data\_size;

int heap\_size;

struct heap\_entry heap\_data[4 \* BOARDMAX];

struct heap\_entry \*heap[BOARDMAX];

int target;

int cutoff\_distance;

int speculative;

};

void compute\_connection\_distances(int str, int target, int cutoff,

struct connection\_data \*conn,

int speculative);

void init\_connection\_data(int color, const signed char goal[BOARDMAX],

int target, int cutoff,

struct connection\_data \*conn, int speculative);

void spread\_connection\_distances(int color, struct connection\_data \*conn);

void sort\_connection\_queue\_tail(struct connection\_data \*conn);

void expand\_connection\_queue(struct connection\_data \*conn);

void print\_connection\_distances(struct connection\_data \*conn);

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\*

\* src/gmp.h

\* Copyright (C) 1995-1996 William Shubert.

\*

\* Parts of this file are taken from "protocol.c", which is covered under

\* the copyright notice below.

\* Any code that is not present in the original "proocol.c" is covered under

\* the copyright notice above.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

protocol.c 1.00

JCGA Go Communication Protocol

copyright(c) Shuji Sunaga 95.7.9

original Standard Go Modem Protocol 1.0

by David Fotland

\* Permission granted to use this code for any

\* commercial or noncommercial purposes as long as this

\* copyright notice is not removed.

\* This code was written for Borland C++ 4.0J

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* You may use this code in any way you wish as long as you retain the

\* above copyright notices.

\*/

/\* Modified by Paul Pogonyshev on 10.07.2003.

\* Added support for Simplified GTP.

\*/

#ifndef \_GMP\_H\_

#define \_GMP\_H\_ 1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Data types

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef enum {

gmp\_nothing, gmp\_move, gmp\_pass, gmp\_reset, gmp\_newGame, gmp\_undo, gmp\_err

} GmpResult;

#ifndef \_GMP\_C\_

typedef struct Gmp\_struct Gmp;

#endif /\* \_GMP\_C\_ \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Fuctions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern Gmp \*gmp\_create(int inFile, int outFile);

extern void gmp\_destroy(Gmp \*ge);

/\*

\* This starts a game up.

\* If you want, you can pass in -1 for size, handicap, and chineseRules,

\* and it will query. You can also pass in -1 for iAmWhite, but if you do

\* this then it will send a RESET command. If the other machine is black

\* and doesn't support arbitration, then this could screw things up.

\* Komi must be specified since GMP doesn't let you exchange komi information.

\* After calling this function, you should call gmp\_check until you get a

\* "gmp\_newGame" returned. Then you know that the size, etc. have all been

\* verified, you can call "gmp\_size()" or whatever to find out values you

\* set to -1, and you can start the game.

\*/

extern void gmp\_startGame(Gmp \*ge, int boardsize, int handicap,

float komi, int chineseRules, int iAmWhite,

int simplified);

/\*

\* Pretty self-explanatory set of routines. For sendMove, (0,0) is the

\* corner.

\*/

extern void gmp\_sendMove(Gmp \*ge, int x, int y);

extern void gmp\_sendPass(Gmp \*ge);

extern void gmp\_sendUndo(Gmp \*ge, int numUndos);

/\*

\* gmp\_check() process all data queued up until the next command that needs

\* to be returned to the application. If sleep is nonzero, then it will

\* stay here until a command arrives. If sleep is zero, then it will

\* return immediately if no command is ready.

\* It should be called about once per second to prevent the connection

\* between the programs from timing out.

\* If you get a move, "out1" will be the X and "out2" will be the y.

\*/

extern GmpResult gmp\_check(Gmp \*ge, int sleepy,

int \*out1, int \*out2, const char \*\*error);

/\*

\* These routines return the configuration of the game that you are playing

\* in. They should all be set up by the time you get a gmpResult\_newGame

\* from gmp\_read().

\* If you get a -1 back from these, it means that you didn't set the value

\* when you called gmp\_startGame() and your opponent wouldn't say what

\* they had the parameter set to.

\*/

extern int gmp\_size(Gmp \*ge);

extern int gmp\_handicap(Gmp \*ge);

extern float gmp\_komi(Gmp \*ge);

extern int gmp\_chineseRules(Gmp \*ge);

extern int gmp\_iAmWhite(Gmp \*ge);

/\*

\* This is handy if you want to print out, as an ascii string, the result

\* that you got bock from gmp\_read().

\*/

extern const char \*gmp\_resultString(GmpResult result);

#endif /\* \_GMP\_H\_ \*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* To facilitate development of the Go Text Protocol, the two \*

\* files gtp.c and gtp.h are licensed under less restrictive \*

\* terms than the rest of GNU Go. \*

\* \*

\* Copyright 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 and \*

\* 2009 by the Free Software Foundation. \*

\* \*

\* Permission is hereby granted, free of charge, to any person \*

\* obtaining a copy of this file gtp.h, to deal in the Software \*

\* without restriction, including without limitation the rights \*

\* to use, copy, modify, merge, publish, distribute, and/or \*

\* sell copies of the Software, and to permit persons to whom \*

\* the Software is furnished to do so, provided that the above \*

\* copyright notice(s) and this permission notice appear in all \*

\* copies of the Software and that both the above copyright \*

\* notice(s) and this permission notice appear in supporting \*

\* documentation. \*

\* \*

\* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY \*

\* KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE \*

\* WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR \*

\* PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO \*

\* EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS \*

\* NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR \*

\* CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING \*

\* FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF \*

\* CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT \*

\* OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS \*

\* SOFTWARE. \*

\* \*

\* Except as contained in this notice, the name of a copyright \*

\* holder shall not be used in advertising or otherwise to \*

\* promote the sale, use or other dealings in this Software \*

\* without prior written authorization of the copyright holder. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

/\* NOTE: GNU Go specific, workarounds dumb Windows sockets. \*/

#include "winsocket.h"

#include <stdarg.h>

#include <stdio.h>

/\* Maximum allowed line length in GTP. \*/

#define GTP\_BUFSIZE 1000

/\* Status returned from callback functions. \*/

#define GTP\_QUIT -1

#define GTP\_OK 0

#define GTP\_FATAL 1

/\* Whether the GTP command was successful. \*/

#define GTP\_SUCCESS 0

#define GTP\_FAILURE 1

/\* Function pointer for callback functions. \*/

typedef int (\*gtp\_fn\_ptr)(char \*s);

/\* Function pointer for vertex transform functions. \*/

typedef void (\*gtp\_transform\_ptr)(int ai, int aj, int \*bi, int \*bj);

/\* Elements in the array of commands required by gtp\_main\_loop. \*/

struct gtp\_command {

const char \*name;

gtp\_fn\_ptr function;

};

void gtp\_main\_loop(struct gtp\_command commands[],

FILE \*gtp\_input, FILE \*gtp\_output, FILE \*gtp\_dump\_commands);

void gtp\_internal\_set\_boardsize(int size);

void gtp\_set\_vertex\_transform\_hooks(gtp\_transform\_ptr in,

gtp\_transform\_ptr out);

void gtp\_mprintf(const char \*format, ...);

void gtp\_printf(const char \*format, ...);

void gtp\_start\_response(int status);

int gtp\_finish\_response(void);

int gtp\_success(const char \*format, ...);

int gtp\_failure(const char \*format, ...);

void gtp\_panic(void);

int gtp\_decode\_color(char \*s, int \*color);

int gtp\_decode\_coord(char \*s, int \*m, int \*n);

int gtp\_decode\_move(char \*s, int \*color, int \*i, int \*j);

void gtp\_print\_vertices(int n, int movei[], int movej[]);

void gtp\_print\_vertex(int i, int j);

extern FILE \*gtp\_output\_file;

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

/\*-------------------------------------------------------------------------

\* interface.h

\* This file contains all headers for interfaces

\*-------------------------------------------------------------------------\*/

#ifndef \_PLAY\_INTERFACE\_H

#define \_PLAY\_INTERFACE\_H

#include "gnugo.h"

#include "sgftree.h"

void play\_ascii(SGFTree \*tree, Gameinfo \*gameinfo,

char \*filename, char \*until);

void play\_gtp(FILE \*gtp\_input, FILE \*gtp\_output, FILE \*gtp\_dump\_commands,

int gtp\_initial\_orientation);

void play\_gmp(Gameinfo \*gameinfo, int simplified);

void play\_solo(Gameinfo \*gameinfo, int benchmark);

void play\_replay(SGFTree \*tree, int color\_to\_test);

void load\_and\_analyze\_sgf\_file(Gameinfo \*gameinfo);

void load\_and\_score\_sgf\_file(SGFTree \*tree, Gameinfo \*gameinfo,

const char \*scoringmode);

#endif

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 \*

\* or (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_DFA\_MKPAT\_H\_

#define \_DFA\_MKPAT\_H\_

#include "dfa.h"

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Parameters \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define DFA\_RESIZE\_STEP 20000

#define DFA\_INIT\_SIZE 250

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Data types definition \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Intersections. \*/

typedef unsigned short Intersection\_t;

/\* Attribute list. \*/

typedef struct attrib

{

int val;

int next;

} attrib\_t;

/\* DFA state. \*/

typedef struct state

{

int att;

int next[4];

} state\_t;

/\* DFA. \*/

typedef struct dfa

{

/\* File header. \*/

char name[80];

/\* Transition graph. \*/

state\_t \*states;

int max\_states;

int last\_state;

/\* Attributes sets. \*/

attrib\_t \*indexes;

int max\_indexes;

int last\_index;

} dfa\_t;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Functions declaration \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dfa\_init(void); /\* Every call to a DFA function must be done \*/

void dfa\_end(void); /\* between calls to these two functions. \*/

/\* Basic DFA manipulation. \*/

void print\_c\_dfa(FILE \*of, const char \*name, dfa\_t \*pdfa);

void new\_dfa(dfa\_t \*pdfa, const char \*name);

void copy\_dfa(dfa\_t \*p\_to, dfa\_t \*p\_from);

void kill\_dfa(dfa\_t \*pdfa);

int dfa\_size(dfa\_t \*pdfa); /\* in kB \*/

void save\_dfa(const char \*f\_name, dfa\_t \*pdfa);

dfa\_t \*load\_dfa(const char \*f\_path, const char \*f\_name, dfa\_t \*\*ppdfa);

void dfa\_finalize(dfa\_t \*pdfa);

void dfa\_shuffle(dfa\_t \*pdfa);

int dfa\_calculate\_max\_matched\_patterns(dfa\_t \*pdfa);

int dfa\_minmax\_delta(dfa\_t \*pdfa, int next\_index, int isMin);

void dump\_dfa(FILE \*f, dfa\_t \*pdfa);

struct pattern;

/\* Conversion between a GNU Go pattern struct into a DFA string. \*/

void pattern\_2\_string(struct pattern \*pat, struct patval\_b \*elements,

char \*str, int ci, int cj);

void dfa\_rotate\_string(char \*strrot, const char \*str, int ll);

/\* Add a string with attribute `att\_val' into a DFA. \*/

float dfa\_add\_string(dfa\_t \*pdfa, const char \*str, int pattern\_index, int ll);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Global variables \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern int dfa\_verbose; /\* Verbiage level. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Experimental DFA builder \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define DFA\_ATTRIB\_BLOCK\_SIZE 150000

#define DFA\_NODE\_BLOCK\_SIZE 50000

typedef struct \_dfa\_attrib dfa\_attrib;

typedef struct \_dfa\_attrib\_block dfa\_attrib\_block;

typedef struct \_dfa\_attrib\_array dfa\_attrib\_array;

typedef struct \_dfa\_node dfa\_node;

typedef struct \_dfa\_node\_block dfa\_node\_block;

typedef struct \_dfa\_graph dfa\_graph;

struct \_dfa\_attrib {

dfa\_attrib \*next;

int string\_index;

};

struct \_dfa\_attrib\_block {

dfa\_attrib\_block \*previous;

dfa\_attrib attrib[DFA\_ATTRIB\_BLOCK\_SIZE];

};

struct \_dfa\_attrib\_array {

dfa\_attrib\_block \*last\_block;

int allocated;

};

struct \_dfa\_node {

dfa\_node \*branch[4];

dfa\_attrib \*attributes;

dfa\_attrib \*passing\_strings;

};

struct \_dfa\_node\_block {

dfa\_node\_block \*previous;

dfa\_node node[DFA\_NODE\_BLOCK\_SIZE];

};

struct \_dfa\_graph {

int num\_nodes;

dfa\_node \*root;

dfa\_node\_block \*last\_block;

int allocated;

dfa\_attrib\_array attributes;

};

#define DFA\_HASH\_BLOCK\_SIZE 10000

#define DFA\_HASH\_TABLE\_SIZE 4096

#define DFA\_HASH\_VALUE\_1 1

#define DFA\_HASH\_VALUE\_2 79

#define DFA\_HASH\_VALUE\_3 2971

typedef struct \_dfa\_hash\_entry dfa\_hash\_entry;

typedef struct \_dfa\_hash\_block dfa\_hash\_block;

struct \_dfa\_hash\_entry {

dfa\_hash\_entry \*next;

dfa\_attrib \*key;

dfa\_node \*value;

};

struct \_dfa\_hash\_block {

dfa\_hash\_block \*previous;

dfa\_hash\_entry entry[DFA\_HASH\_BLOCK\_SIZE];

};

typedef struct \_dfa\_pattern dfa\_pattern;

typedef struct \_dfa\_patterns dfa\_patterns;

struct \_dfa\_pattern {

dfa\_pattern \*next;

int num\_variations;

int current\_variation;

char \*variation[8];

};

struct \_dfa\_patterns {

int num\_patterns;

dfa\_pattern \*patterns;

dfa\_pattern \*last\_pattern;

dfa\_graph graph;

};

void dfa\_graph\_reset(dfa\_graph \*graph);

void dfa\_patterns\_reset(dfa\_patterns \*patterns);

void dfa\_patterns\_clear(dfa\_patterns \*patterns);

void dfa\_patterns\_add\_pattern(dfa\_patterns \*patterns,

const char \*string, int index);

void dfa\_patterns\_set\_last\_pattern\_variation(dfa\_patterns \*patterns,

int variation);

void dfa\_patterns\_select\_shortest\_variation(dfa\_patterns \*patterns);

void dfa\_patterns\_build\_graph(dfa\_patterns \*patterns);

int \*dfa\_patterns\_optimize\_variations(dfa\_patterns \*patterns, int iterations);

#endif /\* \_DFA\_MKPAT\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_DFA\_H\_

#define \_DFA\_H\_

#if MAX\_BOARD > 11

#define DFA\_MAX\_BOARD MAX\_BOARD

#else

#define DFA\_MAX\_BOARD 11

#endif

#define DFA\_MAX\_ORDER ((2 \* DFA\_MAX\_BOARD - 1) \

\* (2 \* DFA\_MAX\_BOARD - 1))

#define DFA\_BASE (3 \* DFA\_MAX\_BOARD)

#define DFA\_POS(i, j) (((i) + DFA\_MAX\_BOARD) \* DFA\_BASE \

+ ((j) + DFA\_MAX\_BOARD))

#ifndef EMPTY

#define EMPTY 0 /\* . \*/

#define WHITE 1 /\* O \*/

#define BLACK 2 /\* X \*/

#endif

#define OUT\_BOARD 3 /\* # \*/

/\* Maximum pattern matched at one positions. \*/

#define DFA\_MAX\_MATCHED (8 \* 24)

/\* DFA spiral order. \*/

extern int spiral[DFA\_MAX\_ORDER][8];

void build\_spiral\_order(void);

/\* The run-time data structures declared here are different from those

\* used internally to build the DFA. \*/

/\* Attribute list. \*/

typedef struct attrib\_rt

{

short val;

short next;

} attrib\_rt\_t;

/\* DFA state. \*/

typedef struct state\_rt

{

short next[4];

short att;

} state\_rt\_t;

typedef struct dfa\_rt

{

/\* File header. \*/

const char name[80];

/\* Transition graph. \*/

const state\_rt\_t \*states;

/\* Attributes sets. \*/

const attrib\_rt\_t \*indexes;

} dfa\_rt\_t;

#endif /\* \_DFA\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#include "liberty.h"

#define CAN\_BE\_EMPTY 1

#define CAN\_CONTAIN\_STONE 2

#define EYE\_DEFENSE\_POINT 4

#define EYE\_ATTACK\_POINT 8

/\*

\* The vertices of each eye are defined by an array of struct eye\_vertex.

\*/

struct eye\_vertex {

signed char marginal; /\* 1 if marginal vertex, 0 otherwise \*/

signed char edge; /\* 0 = center, 1 = edge, 2 = corner \*/

/\* A corner vertex may only be matched at the corner.

\* An edge vertex may be matched at the corner or on the edge.

\* A center vertex may be matched anywhere.

\*/

signed char flags; /\* see the #defines above \*/

signed char neighbors; /\* number of neighbors \*/

signed char n[4]; /\* position in array of vertex neighbors \*/

};

/\*

\* Each eye is described by one struct eye\_graph and the vertices

\* in the struct eye\_vertex array.

\*/

struct eye\_graph {

struct eye\_vertex \*vertex;

int patnum; /\* number of pattern \*/

int esize; /\* number of vertices \*/

int msize; /\* number of marginal vertices \*/

int ends; /\* number of vertices with one neighbor \*/

int two\_neighbors; /\* number of vertices with 2 neighbors \*/

int three\_neighbors; /\* number of vertices with 3 neighbors \*/

struct eyevalue value; /\* eye value \*/

};

extern struct eye\_graph graphs[];

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

/\* This file describes the compiled form of the pattern database.

\* mkpat is used to compile various source files <name>.db into

\* intermediate files <name>.c which define data structures

\* describing the patterns.

\*/

#ifndef \_PATTERN\_H\_

#define \_PATTERN\_H\_

#ifdef HAVE\_CONFIG\_H

#include <config.h>

#endif

/\* local versions of absolute value, min and max \*/

#define gg\_abs(x) ((x) < 0 ? -(x) : (x))

#define gg\_min(a, b) ((a)<(b) ? (a) : (b))

#define gg\_max(a, b) ((a)<(b) ? (b) : (a))

/\* This tells Alpha OSF/1 not to define a getopt prototype in <stdio.h>.

\* Ditto for AIX 3.2 and <stdlib.h>.

\*/

#ifndef \_NO\_PROTO

#define \_NO\_PROTO

#endif

#ifdef HAVE\_CONFIG\_H

#include <config.h>

#else

#define GRID\_OPT 0

#endif

#ifndef GRID\_OPT

#error GRID\_OPT should be defined as 0, 1 or 2

#endif

/\* Include support for pattern profiling. May be turned off in stable

\* releases to save some memory.

\*

\* FIXME: should probably be included in config.h

\*/

#define PROFILE\_PATTERNS 0

/\* this trick forces a compile error if ints are not at least 32-bit \*/

struct \_unused\_patterns\_h {

int unused[sizeof(unsigned int) >= 4 ? 1 : -1];

};

#define ATTACK\_MACRO(pos) ((stackp == 0) ? (worm[pos].attack\_codes[0]) : attack(pos, NULL))

#define DEFEND\_MACRO(pos) ((stackp == 0) ? (worm[pos].defense\_codes[0]) : find\_defense(pos, NULL))

struct pattern; /\* forward reference to keep gcc happy \*/

/\* this is the type of a function which the matcher can

\* call to evaluate the score of a move.

\* parameters:

\* pattern and rotation are the current pattern being considered

\* ti, tj: IN = posn of the 7, 8 or 9 marker

\* OUT = recommended move

\* return value : weight of move, or 0 if match failed

\*/

typedef int (\*pattern\_helper\_fn\_ptr)(struct pattern \*, int rotation,

int move, int color);

typedef int (\*autohelper\_fn\_ptr)(int rotation, int move,

int color, int action);

/\* each pattern is compiled into a sequence of these elements.

\* Each describes a relative x, y from the pattern origin,

\* and a description of what should be there.

\* Current attributes are

\* 0 = .

\* 1 = X

\* 2 = O

\* 3 = x

\* 4 = o

\* 5 = , (barriers only)

\* 6 = a (half-eye only, OBSOLETE)

\* 7 = ! (connection and barriers only)

\*/

#define ATT\_dot 0

#define ATT\_X 1

#define ATT\_O 2

#define ATT\_x 3

#define ATT\_o 4

#define ATT\_comma 5

#define ATT\_a 6

#define ATT\_not 7

/\* Pattern classes. The semantics of these varies between different

\* databases. The descriptions here mostly relate to patterns in

\* patterns.db and other databases which are handled by shapes.c.

\*/

#define CLASS\_O 0x0001 /\* O stones must be alive or unknown \*/

#define CLASS\_o 0x0002 /\* O stones must be dead or unknown \*/

#define CLASS\_X 0x0004 /\* X stones must be alive or unknown \*/

#define CLASS\_x 0x0008 /\* X stones must be dead or unknown \*/

#define CLASS\_s 0x0010 /\* move is a sacrifice \*/

#define CLASS\_n 0x0020 /\* X could also make this move if we do not \*/

#define CLASS\_D 0x0040 /\* defense pattern \*/

#define CLASS\_C 0x0080 /\* move connects two worms \*/

#define CLASS\_c 0x0100 /\* move weakly connects two worms \*/

/\* for owl databases: combinable pattern \*/

#define CLASS\_B 0x0200 /\* move breaks connection between enemy worms \*/

#define CLASS\_A 0x0400 /\* attack pattern \*/

#define CLASS\_b 0x0800 /\* move is intended to block opponent \*/

#define CLASS\_e 0x1000 /\* move is intended to expand territory \*/

#define CLASS\_E 0x2000 /\* move is intended to expand moyo \*/

#define CLASS\_a 0x4000 /\* strategical level attack \*/

#define CLASS\_d 0x8000 /\* strategical level defense \*/

#define CLASS\_I 0x00010000 /\* invasions patterns (influence.db) \*/

#define CLASS\_J 0x00020000 /\* joseki standard move \*/

#define CLASS\_j 0x00040000 /\* joseki move, slightly less urgent \*/

#define CLASS\_t 0x00080000 /\* minor joseki move (tenuki OK) \*/

#define CLASS\_U 0x00100000 /\* very urgent joseki move \*/

#define CLASS\_T 0x00200000 /\* joseki trick move \*/

#define CLASS\_W 0x00400000 /\* worthwhile threat move \*/

#define CLASS\_F 0x00800000 /\* for joseki moves: a fuseki pattern \*/

#define CLASS\_N 0x01000000 /\* antisuji move (do \_not\_ play) \*/

#define CLASS\_Y 0x80000000 /\* used for experimental patterns \*/

/\* Collection of the classes inducing move reasons. \*/

#define CLASS\_MOVE\_REASONS (CLASS\_C | CLASS\_B | CLASS\_b | \

CLASS\_e | CLASS\_E | CLASS\_I | CLASS\_a | CLASS\_d | \

CLASS\_J | CLASS\_j | CLASS\_U | CLASS\_T | CLASS\_t | \

CLASS\_W | CLASS\_c | CLASS\_F)

/\* directions for applying edge-constraints \*/

#define NORTH\_EDGE 1

#define SOUTH\_EDGE 2

#define EAST\_EDGE 4

#define WEST\_EDGE 8

/\* different kinds of autohelpers \*/

#define HAVE\_CONSTRAINT 1

#define HAVE\_ACTION 2

/\* Values of the action parameter to indicate where an influence autohelper

\* is called from.

\*/

#define INFLUENCE\_CALLBACK 1

#define FOLLOWUP\_INFLUENCE\_CALLBACK 2

typedef struct patval {

short offset;

unsigned char att;

} Patval;

/\* Build-time version of patval structure. \*/

typedef struct patval\_b {

int x;

int y;

int att;

} Patval\_b;

enum attribute\_type {

MIN\_VALUE,

MAX\_VALUE,

MIN\_TERRITORY,

MAX\_TERRITORY,

SHAPE,

FOLLOWUP,

REVERSE\_FOLLOWUP,

/\* For `mkpat'. \*/

FIRST\_OFFSET\_ATTRIBUTE,

THREATENS\_TO\_CAPTURE = FIRST\_OFFSET\_ATTRIBUTE,

THREATENS\_EYE,

REVERSE\_SENTE,

NUM\_ATTRIBUTES,

LAST\_ATTRIBUTE = NUM\_ATTRIBUTES

};

#ifdef HAVE\_TRANSPARENT\_UNIONS

struct pattern\_attribute {

enum attribute\_type type;

/\* GCC allows unnamed (and transparent) unions. \*/

union {

float value;

int offset;

};

};

#else

struct pattern\_attribute {

enum attribute\_type type;

float value;

int offset;

};

#endif

/\*

\* Each pattern as a whole is compiled to an instance of this structure.

\*/

struct pattern {

struct patval \*patn; /\* array of elements \*/

int patlen; /\* number of elements \*/

int trfno; /\* number of transformations (rotations and reflections) \*/

const char \*name; /\* short description of pattern (optional) \*/

int mini, minj; /\* min and max (relative to anchor) extent of ... \*/

int maxi, maxj; /\* ...the pattern \*/

int height, width; /\* differences between max and min extents \*/

unsigned int edge\_constraints; /\* and combinations of NORTH, EAST etc.

\* for edges \*/

int move\_offset; /\* offset of the suggested move (relative to anchor) \*/

#if GRID\_OPT

unsigned int and\_mask[8]; /\* for each rotation, masks for a \*/

unsigned int val\_mask[8]; /\* 4x4 grid around anchor \*/

#endif

unsigned int class; /\* classification of pattern \*/

/\* Value (owl-style, used for pattern sorting) is not stored as an

\* attribute, because it is very common.

\*/

float value;

/\* Pattern attributes like shape, followup etc. \*/

struct pattern\_attribute \*attributes;

int autohelper\_flag; /\* whether autohelper has constraint and/or action \*/

pattern\_helper\_fn\_ptr helper; /\* helper function, or NULL \*/

autohelper\_fn\_ptr autohelper; /\* automatically generated helper \*/

/\* function, or NULL \*/

int anchored\_at\_X; /\* 3 if the pattern has 'X' at the anchor posn \*/

float constraint\_cost; /\* mkpat's estimate of the constraint complexity.\*/

#if PROFILE\_PATTERNS

int hits;

int dfa\_hits;

int reading\_nodes;

#endif

};

struct pattern\_db {

int fixed\_for\_size;

const int fixed\_anchor;

struct pattern \*patterns;

struct dfa\_rt \*pdfa;

};

struct fullboard\_pattern {

Hash\_data fullboard\_hash; /\* Hash of the full board position. \*/

int number\_of\_stones; /\* Number of stones on board. \*/

const char \*name; /\* Pattern identifier. \*/

int move\_offset; /\* position of the move relative to tengen \*/

int value; /\* value for pattern, if matched \*/

};

/\* Monte Carlo local patterns. \*/

struct mc\_pattern\_database {

const char \*name;

const unsigned int \*values;

};

/\* helper functions \*/

#define DECLARE(x) int x(struct pattern \*pattern, int transformation, int move, int color)

DECLARE(jump\_out\_helper);

DECLARE(jump\_out\_far\_helper);

DECLARE(high\_handicap\_helper);

DECLARE(reinforce\_helper);

DECLARE(throw\_in\_atari\_helper);

DECLARE(cutstone2\_helper);

DECLARE(thrash\_around\_helper);

/\* autohelper fns \*/

int seki\_helper(int str);

void threaten\_to\_save\_helper(int move, int str);

void threaten\_to\_capture\_helper(int move, int str);

void prevent\_attack\_threat\_helper(int move, int str);

void defend\_against\_atari\_helper(int move, int str);

void amalgamate\_most\_valuable\_helper(int apos, int bpos, int cpos);

int finish\_ko\_helper(int apos);

int squeeze\_ko\_helper(int apos);

int backfill\_helper(int apos, int bpos, int cpos);

int owl\_threatens\_attack(int apos, int bpos);

int connect\_and\_cut\_helper(int Apos, int bpos, int cpos);

int connect\_and\_cut\_helper2(int Apos, int bpos, int cpos, int color);

int edge\_double\_sente\_helper(int move, int apos, int bpos, int cpos);

void test\_attack\_either\_move(int move, int color, int worma, int wormb);

int adjacent\_to\_stone\_in\_atari(int str);

int adjacent\_to\_defendable\_stone\_in\_atari(int str);

void backfill\_replace(int move, int str);

int break\_mirror\_helper(int str, int color);

int distrust\_tactics\_helper(int str);

int disconnect\_helper(int apos, int bpos);

/\* pattern arrays themselves \*/

extern struct pattern\_db pat\_db;

extern struct pattern\_db aa\_attackpat\_db;

extern struct pattern\_db owl\_attackpat\_db;

extern struct pattern\_db owl\_vital\_apat\_db;

extern struct pattern\_db owl\_defendpat\_db;

extern struct pattern\_db conn\_db;

extern struct pattern\_db attpat\_db;

extern struct pattern\_db defpat\_db;

extern struct pattern\_db endpat\_db;

extern struct pattern\_db influencepat\_db;

extern struct pattern\_db barrierspat\_db;

extern struct pattern\_db fusekipat\_db;

extern struct pattern\_db handipat\_db;

extern struct pattern\_db oracle\_db;

extern struct corner\_db joseki\_db;

extern struct fullboard\_pattern fuseki19[];

extern struct fullboard\_pattern fuseki13[];

extern struct fullboard\_pattern fuseki9[];

extern struct mc\_pattern\_database mc\_pattern\_databases[];

struct corner\_db;

struct corner\_variation;

struct corner\_pattern;

struct corner\_db {

int max\_width; /\* Largest possible width and... \*/

int max\_height; /\* ... largest possible height of database patterns. \*/

unsigned char num\_top\_variations; /\* Number of top level variations. \*/

struct corner\_variation \*top\_variations;

};

struct corner\_variation {

int move\_offset; /\* Offset of the move in this variation. \*/

signed char xor\_att; /\* 0 - the same color as the first matched stone,

\* 3 - the opposite color.

\*/

unsigned char num\_stones; /\* Number of stones in the `move\_offset' rectangle. \*/

unsigned char num\_variations; /\* Number of subvariations. \*/

struct corner\_variation \*variations; /\* Pointer to subvariation array. \*/

struct corner\_pattern \*pattern; /\* Address of matched pattern (if any). \*/

};

struct corner\_pattern {

int second\_corner\_offset; /\* Offset of pattern's second corner. \*/

int symmetric; /\* If the pattern is symmetric ('/' symmetry). \*/

unsigned int class; /\* Pattern class. \*/

const char \*name; /\* Pattern name (optional). \*/

/\* Pattern attributes like shape (the only one used currently). \*/

struct pattern\_attribute \*attributes;

int autohelper\_flag; /\* Whether autohelper has constraint and/or action. \*/

autohelper\_fn\_ptr autohelper; /\* Automatically generated helper (or NULL). \*/

};

/\* Build time version of corner\_variation structure. \*/

struct corner\_variation\_b {

int move\_offset;

signed char xor\_att;

unsigned char num\_stones;

unsigned char num\_variations;

struct corner\_variation\_b \*next;

struct corner\_variation\_b \*child;

int child\_num;

int pattern\_num;

};

#endif /\* \_PATTERN\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* # \*/

/\* # This document was taken from the SGF Specfication. See: \*/

/\* # http://www.red-bean.com/sgf/ \*/

/\* # \*/

/\* # [SGF FF[4] - Smart Game Format] \*/

/\* # \*/

/\* # FF[4] property index \*/

/\* # \*/

/\* # This is an alphabetical index to all properties defined in FF[4]. \*/

/\* # New properties are marked with '\*', changed properties are marked with '!'. \*/

/\* # \*/

/\* #ID Description property type property value \*/

/\* #---- --------------- --------------- -------------------------------------- \*/

/\* Add Black setup list of stone \*/

#define SGFAB 16961

/\* Add Empty setup list of point \*/

#define SGFAE 17729

/\* Annotation game-info simpletext \*/

#define SGFAN 20033

/\* Application root composed simpletext ':' simpletext \*/

#define SGFAP 20545

/\* Arrow - list of composed point ':' point \*/

#define SGFAR 21057

/\* Who adds stones - (LOA) simpletext \*/

#define SGFAS 21313

/\* Add White setup list of stone \*/

#define SGFAW 22337

/\* Black move move \*/

#define SGFB 8258

/\* Black time left move real \*/

#define SGFBL 19522

/\* Bad move move double \*/

#define SGFBM 19778

/\* Black rank game-info simpletext \*/

#define SGFBR 21058

/\* Black team game-info simpletext \*/

#define SGFBT 21570

/\* Comment - text \*/

#define SGFC 8259

/\* Charset root simpletext \*/

#define SGFCA 16707

/\* Copyright game-info simpletext \*/

#define SGFCP 20547

/\* Circle - list of point \*/

#define SGFCR 21059

/\* Dim points - (inherit) elist of point \*/

#define SGFDD 17476

/\* Even position - double \*/

#define SGFDM 19780

/\* Doubtful move none \*/

#define SGFDO 20292

/\* Date game-info simpletext \*/

#define SGFDT 21572

/\* Event game-info simpletext \*/

#define SGFEV 22085

/\* Fileformat root number (range: 1-4) \*/

#define SGFFF 17990

/\* Figure - none | composed number ":" simpletext \*/

#define SGFFG 18246

/\* Good for Black - double \*/

#define SGFGB 16967

/\* Game comment game-info text \*/

#define SGFGC 17223

/\* Game root number (range: 1-5,7-16) \*/

#define SGFGM 19783

/\* Game name game-info simpletext \*/

#define SGFGN 20039

/\* Good for White - double \*/

#define SGFGW 22343

/\* Handicap game-info (Go) number \*/

#define SGFHA 16712

/\* Hotspot - double \*/

#define SGFHO 20296

/\* Initial pos. game-info (LOA) simpletext \*/

#define SGFIP 20553

/\* Interesting move none \*/

#define SGFIT 21577

/\* Invert Y-axis game-info (LOA) simpletext \*/

#define SGFIY 22857

/\* Komi game-info (Go) real \*/

#define SGFKM 19787

/\* Ko move none \*/

#define SGFKO 20299

/\* Label - list of composed point ':' simpletext \*/

#define SGFLB 16972

/\* Line - list of composed point ':' point \*/

#define SGFLN 20044

/\* Mark - list of point \*/

#define SGFMA 16717

/\* set move number move number \*/

#define SGFMN 20045

/\* Nodename - simpletext \*/

#define SGFN 8270

/\* OtStones Black move number \*/

#define SGFOB 16975

/\* Opening game-info text \*/

#define SGFON 20047

/\* Overtime game-info simpletext \*/

#define SGFOT 21583

/\* OtStones White move number \*/

#define SGFOW 22351

/\* Player Black game-info simpletext \*/

#define SGFPB 16976

/\* Place game-info simpletext \*/

#define SGFPC 17232

/\* Player to play setup color \*/

#define SGFPL 19536

/\* Print move mode - (inherit) number \*/

#define SGFPM 19792

/\* Player White game-info simpletext \*/

#define SGFPW 22352

/\* Result game-info simpletext \*/

#define SGFRE 17746

/\* Round game-info simpletext \*/

#define SGFRO 20306

/\* Rules game-info simpletext \*/

#define SGFRU 21842

/\* Markup - (LOA) point \*/

#define SGFSE 17747

/\* Selected - list of point \*/

#define SGFSL 19539

/\* Source game-info simpletext \*/

#define SGFSO 20307

/\* Square - list of point \*/

#define SGFSQ 20819

/\* Style root number (range: 0-3) \*/

#define SGFST 21587

/\* Setup type game-info (LOA) simpletext \*/

#define SGFSU 21843

/\* Size root (number | composed number ':' number) \*/

#define SGFSZ 23123

/\* Territory Black - (Go) elist of point \*/

#define SGFTB 16980

/\* Tesuji move double \*/

#define SGFTE 17748

/\* Timelimit game-info real \*/

#define SGFTM 19796

/\* Triangle - list of point \*/

#define SGFTR 21076

/\* Territory White - (Go) elist of point \*/

#define SGFTW 22356

/\* Unclear pos - double \*/

#define SGFUC 17237

/\* User game-info simpletext \*/

#define SGFUS 21333

/\* Value - real \*/

#define SGFV 8278

/\* View - (inherit) elist of point \*/

#define SGFVW 22358

/\* White move move \*/

#define SGFW 8279

/\* White time left move real \*/

#define SGFWL 19543

/\* White rank game-info simpletext \*/

#define SGFWR 21079

/\* White team game-info simpletext \*/

#define SGFWT 21591

/\* # \*/

/\* # These are additions to the SGF spec- old commands and some others \*/

/\* # \*/

/\* # \*/

/\* # outdated FF3 properties \*/

/\* # \*/

#define SGFBS 21314

#define SGFWS 21335

#define SGFID 17481

#define SGFRG 18258

#define SGFSC 17235

/\* # \*/

/\* # some random ones used by CGoban \*/

/\* # \*/

#define SGFSY 22867

/\* # \*/

/\* # nonstandard SGF property used by GNU Go to mark illegal moves \*/

/\* # \*/

#define SGFIL 19529

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_SGFTREE\_H\_

#define \_SGFTREE\_H\_

#include <stdio.h>

#include "sgf\_properties.h"

#ifndef \_BOARD\_H\_

/\*

\* NOTE: These MUST coincide with the definitions for the engine that we

\* are using. In this case they are defined in engine/gnugo.h.

\*

\* The reason that we put them here within the #ifndef clause is because

\* we want to decouple the GNU Go engine from SGF library, but we don't

\* want to redefine these symbols if we include this file into board.h.

\*/

#define EMPTY 0

#define WHITE 1

#define BLACK 2

#endif

void \*xalloc(unsigned int);

/\*

\* A property of an SGF node. An SGF node is described by a linked

\* list of these.

\*/

typedef struct SGFProperty\_t {

struct SGFProperty\_t \*next;

short name;

char \*value;

} SGFProperty;

typedef struct SGFNode\_t {

SGFProperty \*props;

struct SGFNode\_t \*parent;

struct SGFNode\_t \*child;

struct SGFNode\_t \*next;

} SGFNode;

/\* low level functions \*/

SGFNode \*sgfPrev(SGFNode \*node);

SGFNode \*sgfRoot(SGFNode \*node);

SGFNode \*sgfNewNode(void);

void sgfFreeNode(SGFNode \*node);

int sgfGetIntProperty(SGFNode \*node, const char \*name, int \*value);

int sgfGetFloatProperty(SGFNode \*node, const char \*name, float \*value);

int sgfGetCharProperty(SGFNode \*node, const char \*name, char \*\*value);

void sgfAddProperty(SGFNode \*node, const char \*name, const char \*value);

void sgfAddPropertyInt(SGFNode \*node, const char \*name, long val);

void sgfAddPropertyFloat(SGFNode \*node, const char \*name, float val);

void sgfOverwriteProperty(SGFNode \*node, const char \*name, const char \*text);

void sgfOverwritePropertyFloat(SGFNode \*node, const char \*name, float val);

void sgfOverwritePropertyInt(SGFNode \*node, const char \*name, int val);

void \*xrealloc(void \*pt, unsigned int size);

SGFProperty \*sgfMkProperty(const char \*name, const char \*value,

SGFNode \*node, SGFProperty \*last);

void sgfFreeProperty(SGFProperty \*prop);

SGFNode \*sgfAddStone(SGFNode \*node, int color, int movex, int movey);

SGFNode \*sgfAddPlay(SGFNode \*node, int who, int movex, int movey);

SGFNode \*sgfAddPlayLast(SGFNode \*node, int who, int movex, int movey);

void sgfWriteResult(SGFNode \*node, float score, int overwrite);

void sgf\_write\_header(SGFNode \*root, int overwrite, int seed, float komi,

int handicap, int level, int rules);

SGFNode \*sgfLabel(SGFNode \*node, const char \*label, int i, int j);

SGFNode \*sgfLabelInt(SGFNode \*node, int num, int i, int j);

SGFNode \*sgfCircle(SGFNode \*node, int i, int j);

SGFNode \*sgfSquare(SGFNode \*node, int i, int j);

SGFNode \*sgfTriangle(SGFNode \*node, int i, int j);

SGFNode \*sgfMark(SGFNode \*node, int i, int j);

SGFNode \*sgfAddComment(SGFNode \*node, const char \*comment);

SGFNode \*sgfBoardText(SGFNode \*node, int i, int j, const char \*text);

SGFNode \*sgfBoardChar(SGFNode \*node, int i, int j, char c);

SGFNode \*sgfBoardNumber(SGFNode \*node, int i, int j, int number);

SGFNode \*sgfStartVariant(SGFNode \*node);

SGFNode \*sgfStartVariantFirst(SGFNode \*node);

SGFNode \*sgfAddChild(SGFNode \*node);

SGFNode \*sgfCreateHeaderNode(int boardsize, float komi, int handicap);

/\* Read SGF tree from file. \*/

SGFNode \*readsgffile(const char \*filename);

/\* Specific solution for fuseki \*/

SGFNode \*readsgffilefuseki(const char \*filename, int moves\_per\_game);

/\* Write SGF tree to a file. \*/

int writesgf(SGFNode \*root, const char \*filename);

/\* ---------------------------------------------------------------- \*/

/\* --- SGFTree --- \*/

/\* ---------------------------------------------------------------- \*/

typedef struct SGFTree\_t {

SGFNode \*root;

SGFNode \*lastnode;

} SGFTree;

void sgftree\_clear(SGFTree \*tree);

int sgftree\_readfile(SGFTree \*tree, const char \*infilename);

int sgftreeBack(SGFTree \*tree);

int sgftreeForward(SGFTree \*tree);

void sgftreeAddPlay(SGFTree \*tree, int color, int movex, int movey);

void sgftreeAddPlayLast(SGFTree \*tree, int color, int movex, int movey);

void sgftreeAddStone(SGFTree \*tree, int color, int movex, int movey);

void sgftreeWriteResult(SGFTree \*tree, float score, int overwrite);

SGFNode \*sgftreeNodeCheck(SGFTree \*tree);

void sgftreeCircle(SGFTree \*tree, int i, int j);

void sgftreeSquare(SGFTree \*tree, int i, int j);

void sgftreeTriangle(SGFTree \*tree, int i, int j);

void sgftreeMark(SGFTree \*tree, int i, int j);

void sgftreeAddComment(SGFTree \*tree, const char \*comment);

void sgftreeBoardText(SGFTree \*tree, int i, int j, const char \*text);

void sgftreeBoardChar(SGFTree \*tree, int i, int j, char c);

void sgftreeBoardNumber(SGFTree \*tree, int i, int j, int number);

void sgftreeStartVariant(SGFTree \*tree);

void sgftreeStartVariantFirst(SGFTree \*tree);

void sgftreeCreateHeaderNode(SGFTree \*tree, int boardsize, float komi,

int handicap);

void sgftreeSetLastNode(SGFTree \*tree, SGFNode \*lastnode);

/\* ---------------------------------------------------------------- \*/

/\* --- Utilities --- \*/

/\* ---------------------------------------------------------------- \*/

int get\_moveX(SGFProperty \*property, int boardsize);

int get\_moveY(SGFProperty \*property, int boardsize);

int get\_moveXY(SGFProperty \*property, int \*i, int \*j, int boardsize);

int show\_sgf\_properties(SGFNode \*node);

int show\_sgf\_tree(SGFNode \*node);

int is\_markup\_node(SGFNode \*node);

int is\_move\_node(SGFNode \*node);

int is\_pass\_node(SGFNode \*node, int boardsize);

int find\_move(SGFNode \*node);

#endif

/\* Declarations for getopt.

Copyright (C) 1989,90,91,92,93,94,96,97,98 Free Software Foundation, Inc.

This file is part of the GNU C Library.

The GNU C Library is free software; you can redistribute it and/or

modify it under the terms of the GNU Library General Public License as

published by the Free Software Foundation; either version 3 of the

License, or (at your option) any later version.

The GNU C Library is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Library General Public License for more details.

You should have received a copy of the GNU Library General Public

License along with the GNU C Library; see the file COPYING.LIB. If not,

write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor,

Boston, MA 02110-1301, USA. \*/

#ifndef GG\_GETOPT\_H

#ifndef \_\_need\_getopt

# define GG\_GETOPT\_H 1

#endif

#ifdef \_\_cplusplus

extern "C" {

#endif

/\* For communication from `getopt' to the caller.

When `getopt' finds an option that takes an argument,

the argument value is returned here.

Also, when `ordering' is RETURN\_IN\_ORDER,

each non-option ARGV-element is returned here. \*/

extern char \*gg\_optarg;

/\* Index in ARGV of the next element to be scanned.

This is used for communication to and from the caller

and for communication between successive calls to `getopt'.

On entry to `getopt', zero means this is the first call; initialize.

When `getopt' returns -1, this is the index of the first of the

non-option elements that the caller should itself scan.

Otherwise, `optind' communicates from one call to the next

how much of ARGV has been scanned so far. \*/

extern int gg\_optind;

/\* Callers store zero here to inhibit the error message `getopt' prints

for unrecognized options. \*/

extern int gg\_opterr;

/\* Set to an option character which was unrecognized. \*/

extern int gg\_optopt;

#ifndef \_\_need\_getopt

/\* Describe the long-named options requested by the application.

The LONG\_OPTIONS argument to getopt\_long or getopt\_long\_only is a vector

of `struct option' terminated by an element containing a name which is

zero.

The field `has\_arg' is:

no\_argument (or 0) if the option does not take an argument,

required\_argument (or 1) if the option requires an argument,

optional\_argument (or 2) if the option takes an optional argument.

If the field `flag' is not NULL, it points to a variable that is set

to the value given in the field `val' when the option is found, but

left unchanged if the option is not found.

To have a long-named option do something other than set an `int' to

a compiled-in constant, such as set a value from `optarg', set the

option's `flag' field to zero and its `val' field to a nonzero

value (the equivalent single-letter option character, if there is

one). For long options that have a zero `flag' field, `getopt'

returns the contents of the `val' field. \*/

struct gg\_option

{

# if defined \_\_STDC\_\_ && \_\_STDC\_\_

const char \*name;

# else

char \*name;

# endif

/\* has\_arg can't be an enum because some compilers complain about

type mismatches in all the code that assumes it is an int. \*/

int has\_arg;

int \*flag;

int val;

};

/\* Names for the values of the `has\_arg' field of `struct option'. \*/

# define no\_argument 0

# define required\_argument 1

# define optional\_argument 2

#endif /\* need getopt \*/

/\* Get definitions and prototypes for functions to process the

arguments in ARGV (ARGC of them, minus the program name) for

options given in OPTS.

Return the option character from OPTS just read. Return -1 when

there are no more options. For unrecognized options, or options

missing arguments, `optopt' is set to the option letter, and '?' is

returned.

The OPTS string is a list of characters which are recognized option

letters, optionally followed by colons, specifying that that letter

takes an argument, to be placed in `optarg'.

If a letter in OPTS is followed by two colons, its argument is

optional. This behavior is specific to the GNU `getopt'.

The argument `--' causes premature termination of argument

scanning, explicitly telling `getopt' that there are no more

options.

If OPTS begins with `--', then non-option arguments are treated as

arguments to the option '\0'. This behavior is specific to the GNU

`getopt'. \*/

#if defined \_\_STDC\_\_ && \_\_STDC\_\_

# ifdef \_\_GNU\_LIBRARY\_\_

/\* Many other libraries have conflicting prototypes for getopt, with

differences in the consts, in stdlib.h. To avoid compilation

errors, only prototype getopt for the GNU C library. \*/

extern int gg\_getopt (int argc, char \*const \*argv, const char \*shortopts);

# else /\* not \_\_GNU\_LIBRARY\_\_ \*/

extern int gg\_getopt (int argc, char \*const \*argv, const char \*shortopts);

# endif /\* \_\_GNU\_LIBRARY\_\_ \*/

# ifndef \_\_need\_getopt

extern int gg\_getopt\_long (int argc, char \*const \*argv, const char \*shortopts,

const struct gg\_option \*longopts, int \*longind);

extern int gg\_getopt\_long\_only (int argc, char \*const \*argv,

const char \*shortopts,

const struct gg\_option \*longopts, int \*longind);

/\* Internal only. Users should not call this directly. \*/

extern int \_getopt\_internal (int argc, char \*const \*argv,

const char \*shortopts,

const struct gg\_option \*longopts, int \*longind,

int long\_only);

# endif

#else /\* not \_\_STDC\_\_ \*/

extern int gg\_getopt ();

# ifndef \_\_need\_getopt

extern int gg\_getopt\_long ();

extern int gg\_getopt\_long\_only ();

extern int \_getopt\_internal ();

# endif

#endif /\* \_\_STDC\_\_ \*/

#ifdef \_\_cplusplus

}

#endif

/\* Make sure we later can get all the definitions and declarations. \*/

#undef \_\_need\_getopt

#endif /\* getopt.h \*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_GG\_UTILS\_H\_

#define \_GG\_UTILS\_H\_

#include <stdarg.h>

#ifdef HAVE\_CONFIG\_H

#include <config.h>

#endif

#if TIME\_WITH\_SYS\_TIME

# include <sys/time.h>

# include <time.h>

#else

# if HAVE\_SYS\_TIME\_H

# include <sys/time.h>

# else

# include <time.h>

# endif

#endif

#ifdef \_\_MINGW32\_\_

#include <windows.h>

#include <winsock.h>

#include <io.h>

#endif

#ifdef HAVE\_WINSOCK\_IO\_H

#include <winsock.h>

#include <io.h>

#endif

void gg\_init\_color(void);

void write\_color\_char(int c, int x);

void write\_color\_string(int c, const char \*str);

void gg\_vsnprintf(char \*dest, unsigned long len, const char \*fmt,

va\_list args);

void gg\_snprintf(char \*dest, unsigned long len, const char \*fmt, ...);

double gg\_gettimeofday(void);

double gg\_cputime(void);

float gg\_normalize\_float(float x, float a);

int gg\_normalize\_float2int(float x, float a);

void gg\_sort(void \*base, size\_t nel, size\_t width,

int (\*compar)(const void \*, const void \*));

#define MAX\_INTERPOLATION\_STEPS 20

struct interpolation\_data

{

int sections;

float range\_lowerbound;

float range\_upperbound;

float values[MAX\_INTERPOLATION\_STEPS + 1];

};

float gg\_interpolate(struct interpolation\_data \*f, float x);

float soft\_cap(float a, float b);

const char \*gg\_version(void);

/\* prototypes for basic reorientation functions \*/

void rotate(int i, int j, int \*ri, int \*rj, int bs, int rot);

void inv\_rotate(int i, int j, int \*ri, int \*rj, int bs, int rot);

void update\_random\_seed(void);

void set\_random\_seed(unsigned int seed);

unsigned int get\_random\_seed(void);

void reuse\_random\_seed(void);

#endif /\* \_GG\_UTILS\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef \_RANDOM\_H\_

#define \_RANDOM\_H\_

/\* This random number generator produces 32 bit unsigned integers, no

\* more, no less. Internally in the algorithm and for storing the

\* state we need a type that is at least 32 bits wide. A longer type

\* doesn't hurt but means a waste of bits.

\*

\* ISO C guarantees that an unsigned long always is at least 32 bits.

\* It is not uncommon, however, that it is longer. An unsigned int is

\* not guaranteed to be more than 16 bits wide, but on modern

\* platforms we can be certain that this type too is 32 bits (or

\* more). Also the GNU Coding Standards explicitly state that the

\* possibility of ints shorter than 32 bits should be ignored.

\*

\* We could make a typedef here to choose exactly which type to use.

\* In order to avoid various complications in the interface to the

\* random number generator, however, we prefer to consistently use

\* unsigned int internally and we assume this type to be at least 32

\* bits wide.

\*/

/\* Internal state of the random number generator. \*/

struct gg\_rand\_state {

unsigned int x[25]; /\* Internal state. \*/

int k; /\* Word counter. \*/

};

/\* Seed the random number generator. If an unsigned int is larger than

\* 32 bits, only the 32 least significant bits are used for seeding.

\*/

void gg\_srand(unsigned int seed);

/\* Obtain one random integer value in the interval [0, 2^31-1]. \*/

int gg\_rand(void);

/\* Obtain one random integer value in the interval [0, 2^32-1]. \*/

unsigned int gg\_urand(void);

/\* Obtain one random floating point value in the half open interval

\* [0.0, 1.0).

\*

\* If the value is converted to a floating point type with less than

\* 32 bits mantissa (or if the double type should happen to be

\* unusually short), the value 1.0 may be attained.

\*/

double gg\_drand(void);

/\* Retrieve the internal state of the random generator. \*/

void gg\_get\_rand\_state(struct gg\_rand\_state \*state);

/\* Set the internal state of the random number generator. \*/

void gg\_set\_rand\_state(struct gg\_rand\_state \*state);

#endif /\* \_RANDOM\_H\_ \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\

\* This is GNU Go, a Go program. Contact gnugo@gnu.org, or see \*

\* http://www.gnu.org/software/gnugo/ for more information. \*

\* \*

\* Copyright 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, \*

\* 2008 and 2009 by the Free Software Foundation. \*

\* \*

\* This program is free software; you can redistribute it and/or \*

\* modify it under the terms of the GNU General Public License as \*

\* published by the Free Software Foundation - version 3 or \*

\* (at your option) any later version. \*

\* \*

\* This program is distributed in the hope that it will be useful, \*

\* but WITHOUT ANY WARRANTY; without even the implied warranty of \*

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the \*

\* GNU General Public License in file COPYING for more details. \*

\* \*

\* You should have received a copy of the GNU General Public \*

\* License along with this program; if not, write to the Free \*

\* Software Foundation, Inc., 51 Franklin Street, Fifth Floor, \*

\* Boston, MA 02111, USA. \*

\\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#ifndef GNU\_GO\_WINSOCKET\_H

#define GNU\_GO\_WINSOCKET\_H

#ifdef HAVE\_CONFIG\_H

#include <config.h>

#endif

#define USE\_WINDOWS\_SOCKET\_CLUDGE \

((defined(\_WIN32) || defined(\_WIN32\_WCE)) \

&& defined(ENABLE\_SOCKET\_SUPPORT))

#if USE\_WINDOWS\_SOCKET\_CLUDGE

#include <stdarg.h>

#include <stdio.h>

#ifndef WINSOCKET\_H\_INTERNAL\_INCLUSION

/\* (sic.) Teh cludge. \*/

/\* At least in some versions of `stdio.h' on Windows, feof() is a

\* macro, not a function.

\*/

#ifdef feof

#undef feof

#endif

#define setbuf winsocket\_setbuf

#define fflush winsocket\_fflush

#define feof winsocket\_feof

#define fclose winsocket\_fclose

#define fread winsocket\_fread

#define fgets winsocket\_fgets

#define fwrite winsocket\_fwrite

#define fputc winsocket\_fputc

#define fputs winsocket\_fputs

#define fprintf winsocket\_fprintf

#define vfprintf winsocket\_vfprintf

#endif /\* WINSOCKET\_H\_INTERNAL\_INCLUSION \*/

void winsocket\_activate(int \_socket\_handle);

void winsocket\_setbuf(FILE \*file, char \*buffer);

int winsocket\_fflush(FILE \*file);

int winsocket\_feof(FILE \*file);

int winsocket\_fclose(FILE \*file);

size\_t winsocket\_fread(void \*buffer,

size\_t size, size\_t num\_items, FILE \*file);

char \* winsocket\_fgets(char \*buffer, int size, FILE \*file);

size\_t winsocket\_fwrite(const void \*buffer,

size\_t size, size\_t num\_items, FILE \*file);

int winsocket\_fputc(int character, FILE \*file);

int winsocket\_fputs(const char \*string, FILE \*file);

int winsocket\_fprintf(FILE \*file, const char \*format\_string, ...);

int winsocket\_vfprintf(FILE \*file, const char \*format\_string,

va\_list arguments);

#endif /\* USE\_WINDOWS\_SOCKET\_CLUDGE \*/

#endif /\* GNU\_GO\_WINSOCKET\_H \*/

/\*

\* Local Variables:

\* tab-width: 8

\* c-basic-offset: 2

\* End:

\*/

Dockerfile:

# Pull base image.

FROM ubuntu:16.04

MAINTAINER yangboz <z@smartkit.info>

# Install cmake.

RUN apt-get update; \

apt-get -y upgrade

RUN apt-get -y install g++ cmake

# Install boost.

RUN apt-get --yes install libboost-all-dev

# Define working directory.

COPY . /app

WORKDIR /app

#gnugo install

RUN ./configure \

&& make \

&& make install

# Define default command.

#ENTRYPOINT ["gnugo"]

#CMD ["./configure"]

#CMD ["make"]

#CMD ["make","install"]

#CMD ["gnugo"]

#CMD ["./estimator", "test\_games/1776378.game"]

CMD gnugo --score estimate -l /sgfs/\*.sgf