Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 349 (2005) 184–201
Theoretical Computer Science
www.elsevier.com/locate/tcs

Static analysis based on formal models and incremental computation in Go programming ☆

Katsuhiko Nakamura*

College of Science and Engineering, Tokyo Denki University, Hatoyama-machi, Saitama-ken 350-0394, Japan

Abstract

Computer-Go programs have high computational costs for static analysis, even though most intersections of the board remain unchanged by one move. Therefore, incremental computation as well as theoretical models are essential features for static analysis. This paper describes some formal models for static analysis, and explores how incremental computation is applied to the static analysis in Go programs. The static analysis in this paper includes (1) recognizing blocks and groups of stones and evaluating their properties, (2) determining the life and death of a group by numerical features, (3) finding the numbers of regions enclosed by the groups based on Euler's formula, and (4) analysing capturing races (*semeai*) and *sekis* based on an abstract description called the semeai graph. Several operations on the sets of intersections on the board are used for defining the notions on Go boards as well as for describing the analysis methods.

Keywords: Incremental computation; Set operation; Euler's formula; Life and death; Capturing race; Semeai graph

1. Introduction

The strength of computer-Go programs is generally considered as residing at beginners' level despite all efforts by many researchers. [1] Moreover, the progress in the playing strength is considered to be rather slow as compared to that of computer chess and computer Shogi. The game of Shogi is considered to be very difficult too, but we acknowledge that computer Shogi is steadily improving. [2] For Go, we assume that investigating the theoretical foundations of the game as well as applying the results obtained in the last decade in practical Go programming are significant for the progress in computer Go.

It is a widely accepted idea that efficient static analysis is essential for improving the playing strength of computer-Go programs. The idea might be good, but the costs of such an analysis are much higher than those of chess and Shogi. Most of the static analysis needs to be repeated not only at every actual move by a player, but also at every hypothetical move corresponding to a node in the game tree.

☆ Early versions of this paper were partly presented in [16,17].

* Tel.: +81 49 29 629 11; fax: +81 492 966 185.
E-mail address: nakamura@k.dendai.ac.jp.

[1] Many Go players in Japan estimate the current best Go programs as playing at around 4 or 5 *kyu* in amateur rating, although the Japan Go Association recently certified some Go programs as one *dan*, which is stronger than 5 *kyu* by 5 handicap stones.

[2] Iida recently wrote that since computer Shogi is steadily improving, it will defeat the human champion (*Meijin*) by around 2012 [10].

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.046

In this paper, we describe some formal models for static analysis and explore how the incremental computation can be applied to static analysis. By incremental computation,[3] we mean the following form of data processing for reducing the computation time:

(1) The computation is applied to a sequence of objects, each of which is similar to, but partly changed from, the preceding one.
(2) The computation for each object can be restricted to the changed parts of the objects; repeating the same sub-computation for the unchanged parts is avoided as much as possible.

We discuss the static analysis including (1) recognizing blocks and groups of stones and evaluating their properties, (2) determining the life and death of a group by the numerical features, (3) finding the numbers of regions enclosed by the groups based on Euler's formula, and (4) analysing capturing races (*semeai*) and *sekis* based on semeai graphs.

The aim of the static analysis is to obtain a collection of global elements of the board configuration. We mention three types of elements: (1) territories of black and white stones, (2) influence of stones, and (3) the life and death of the groups. In most cases, the change of a board configuration is restricted to one intersection, except in cases of capturing, which seldom occurs. Usually, the global elements remain unchanged by one move, although there are cases where the global elements change considerably by one move. Using incremental computation allows us to restrict the evaluation process to the changed parts of a configuration and to consider the global elements involved. Since the game of Go requires high computational costs for the static analysis, incremental computation seems to be beneficial to computer Go.

The incremental computation can be also applied to the dynamic analysis in general, since the steps of the game tree analysed previously are also steps of the new game tree. However, the incremental computation for the new game tree has three obstacles

(1) The number of branches to be visited in the bottom part of the tree is generally larger than that of the other part of the tree.
(2) The program needs to store a large volume of partial results of the analysis.
(3) It is not easy to resume the correctly stored analysis, since most programs employ depth-first-based search in the dynamic analysis.

The three obstacles seem to be the main reasons why most game programs do not employ the incremental computation in the dynamic analysis. The three obstacles of the incremental computation, can be characterized accordingly by: (1) the efficiency, (2) the overhead, and (3) the difficulty of implementation. We remark that these three notions are also important in the static analysis.

Most previous publications on static analysis in computer Go dealt with determining the life and death of groups of stones. Those works include: the theoretical study of static life [1]; determining the life and death of groups by some local features including perimeters of the empty regions [5], by tactical analysis and eye values [7], by applying combinatorial game theory [12], and by position evaluation [13].

There are also important previous works on static analysis other than that for life and death. The application of combinatorial game theory to *yose* problems [2] is another theoretical result. It is shown in [18] that the dynamic analysis has sufficient power in the case of Go on as small as 5×5 boards. In [19], a machine learning method is described for applying to score final positions.

Few papers discuss the method of incremental computation in computer Go so far. Most Go-playing programs seem to have some mechanism for incremental computation. Klinger and Mechner [11] and Bouzy [3] describe some methods for incremental updating of data in Go programs. These two publications contain some essential elements of the basics of incremental computation, since they take into account knowledge maintenance and backtracking.

Finally, we discuss the analysis of *capturing race*. A set of black and white groups of stones is in a *capturing race* (or in *semeai*) if a group can be alive only by capturing a group of the opponent [9]. The analysis of the capturing races is closely related to a seki and can be considerably complex, since several groups are related to the race. Although the analysis is essential for determining the life and death of groups, most current Go playing programs do not have sufficient power for analysing capturing races. Müller [13,14] classifies capturing races into nine classes and shows some basic methods of analysing capturing races including a formal model called *conditional combinational games*.

This paper is organized as follows. In Section 2, we describe operations on the set of intersections on the board, which are used for defining the notions on Go boards as well as for describing the analysis methods. In Section 3,

[3] The term "incremental computation" was suggested by H.J. van den Herik. This notion was originally called "differential computation."

we define blocks and groups, and describe methods of recognizing blocks and groups and evaluating the liberties of blocks and groups. Section 4 shows methods of estimating the number of regions enclosed by the groups based on Euler's formula for planar connected graphs. The methods include those for the incremental computation and the application to determining the numbers of eyes in the groups. Section 5 describes a formal model called *semeai* graph for analysing complex capturing races and sekis.

Throughout this paper, we distinguish between *effective* properties and *determinate* properties. A determinate property does not depend on the sequence of moves in the progress of the game. An effective property can be preserved only by responding to the opponent moves with appropriate moves. Static life [1] is an example of the determinate property, whereas the life and death of a group determined by most of the methods in this paper is an effective property.

2. Set operations and incremental computation

In this section, we define (1) a number of constants and (2) a number of operations on the sets of intersections (in 2.1), which are extensions of those introduced in [1]. Moreover, we show how the incremental computation of the operations works (in 2.2). By the operations, we intend to define the basic notions on Go boards as precisely as possible and describe the analysis methods in a formal way rather than to use them directly for the analysis.

2.1. Operations on sets of intersections

The *board* is the set $\mathcal{B} = \{(i, j) \mid 1 \leqslant i, j \leqslant N\}$ of *intersections*. In the standard rule, N is 19. A configuration is represented by two disjoint sets $B \subseteq \mathcal{B}$ and $W \subseteq \mathcal{B}$ of intersections occupied by black and white stones, respectively. The other elements of board in the set $\mathcal{B} - B - W$ are empty intersections. An intersection (i, j) is *adjacent to* an intersection (m, n), if and only if $|i - m| + |j - n| = 1$. An intersection (i, j) is *adjacent to* a set S of intersections, if and only if $(i, j) \notin S$ and there is an intersection $(m, n) \in S$ such that (i, j) is adjacent to (m, n).

The board $\mathcal{B}$ and the empty set $\emptyset$ are constants. Another constant is *edge* defined by $\mathcal{E} \triangleq \{(i, j) |\ i = 1, i = N,$ $j = 1$ or $j = N\}$. We have three types of operations: Boolean, shift, and extended operations. The Boolean operations include union $\cup$, intersection $\cap$, and set difference $-$. There are four shift operations. The operation Shift Left $\overleftarrow{A}$ is defined by

$$\overleftarrow{A} \triangleq \{(i - 1, j) \mid (i, j) \in A,\ i \geqslant 2\}.$$

The value of $\overleftarrow{A}$ is the set of intersections which are shifted left from the intersections in A. The intersections on the left edge in A are eliminated. Other shift operations are: Shift Right $\overrightarrow{A}$, Shift Down $A\downarrow$ and Shift Up $A\uparrow$; they are defined analogously. For a set X, let $|X|$ denote the number of elements in S. The following extended operations are used in the later sections:

$$\begin{aligned} &\mathit{thicken}(X) \triangleq \overleftarrow{X} \cup \overrightarrow{X} \cup X\downarrow \cup X\uparrow \\ &\mathit{ext}(X) \triangleq \mathit{thicken}(X) - X \\ &\#\mathit{adjacent}(X) \triangleq |X \cap \overrightarrow{X}| + |X \cap X\downarrow|. \end{aligned}$$

Note that $\mathit{ext}(X)$ stands for the *exterior* of X and that we can also define $\mathit{ext}(X)$ as the set $\{(i, j) \in \overline{X} \mid (i, j)$ is adjacent to $X\}$. Some examples of these operations are shown in Fig. 1. We represent a configuration (a) by sets B (b) and W (c) of black and white stones, respectively. The value of $\#\mathit{adjacent}(B)$ is 11, and that of $\#\mathit{adjacent}(W)$ is 3.

2.2. Incremental computation of operations

Let Y be any set of stones of the same colour, and A be a set of one stone of the same colour, such that $Y \cap A = \emptyset$. Incremental computation of an operation Op for $Y \cup A$ means finding the result $Op(Y \cup A)$ from the value $Op(Y)$ and the operations on the four neighbour intersections of A. The results of incremental computation for the basic operations

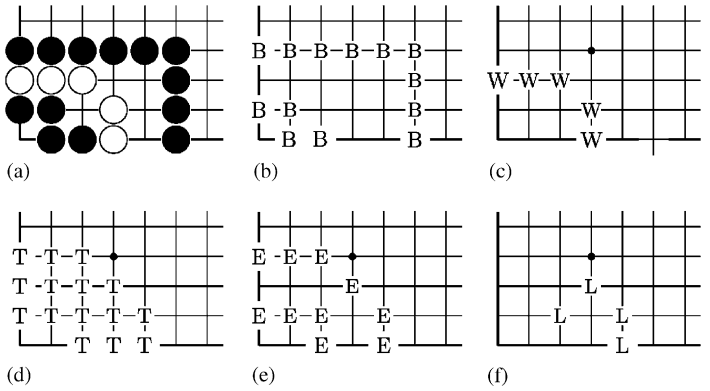


Fig. 1. An example of a configuration and the results of extended operations: (a) A configuration; (b) B; (c) W; (d) $thicken(W)$; (e) $ext(W)$; and (f) $liberty(W, B)$.

are presented as follows:

$$X \cup (Y \cup A) = (X \cup Y) \underline{\cup A},$$

$$X \cap (Y \cup A) = (X \cap Y) \underline{\cup (X \cap A)},$$

$$X - (Y \cup A) = (X - Y) \underline{- A},$$

$$(Y \cup A) - X = (Y - X) \underline{\cup (A - X)},$$

$$\overrightarrow{Y \cup A} = \vec{Y} \ \underline{\cup \ \vec{A}}.$$

Incremental computation for the other shift operations is defined analogously. The underlined parts in the equations represent the increments, which can be obtained by considering only four neighbour states in $ext(A)$. The results of the method for the extended operations are shown below, with $|S|$ being the number of elements in a set S

$$thicken(Y \cup A) = thicken(Y) \underline{\cup thicken(A)}$$

$$ext(Y \cup A) = thicken(Y) \cup thicken(A) - (Y \cup A) = (ext(Y) \underline{- A}) \cup (ext(A) - Y)$$

$$\#adjacent(Y \cup A) \ = |(Y \cup A) \cap (\vec{Y} \cup \vec{A})| + |(Y \cup A) \cap (Y{\downarrow} \cup A{\downarrow})| = \#adjacent(Y) \underline{+ |Y \cap ext(A)|}.$$

3. Blocks and groups

In this section, we define blocks (3.1) and groups of stones (3.2). Moreover, we discuss incremental computation of liberties of blocks (3.3), and the liberties of groups (3.4). Finally, we evaluate life and death of groups enclosing one region (3.5). We remark that in most cases the group is a *effective* property mentioned in Section 1,[4] while a block is a *determinate* property.

3.1. Blocks

We represent a board configuration by sets B_t and W_t of black and white stones. The set E_t of empty intersections is given by $E_t = \mathcal{B} - B_t - W_t$. A *connected* set $S \subseteq \mathcal{B}$ of intersections is recursively defined by:
(1) If $|S| = 1$, then S is connected.
(2) If S is connected, then the union of S and any subset of $ext(S)$ is connected.
A black *block*, also known as a black *string*, is a connected set $B_X \subseteq B_t$ such that $thicken(B_X) \cap B_t = B_X$. The block B_X is the *maximum* connected set in the sense that there is no connected set $B' \subseteq B_t$ with $B_X \subsetneq B'$. *White blocks*

[4] Examples of determinate groups are shown in Fig 10.

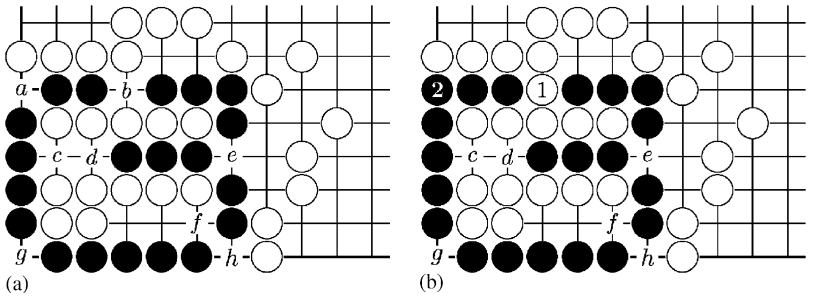


Fig. 2. An example of static groups. Symbols *a–h* denote connecting points.

are defined analogously. An *empty region* is a connected set $E_X \subseteq E_t$ such that $thicken(E_X) \cap E_t = E_X$. A *region* enclosed by black stones, or black-*enclosed region*, is a connected set $R \subseteq W_t \cup E$ such that $ext(R) \subseteq B_t$. White-enclosed regions are defined similarly. A *liberty*, or *dame*, of a black (or white) block B_X (W_X) is an empty intersection in the exterior of the block. Hence, we have

$$liberty(B_X, W_t) \triangleq ext(B_X) \cap E_t = ext(B_X) - W_t.$$

We remark that every block has at least one non-empty liberty, since a block without any liberty is dead and is removed from the board. The configuration in Fig. 1(a) contains two black blocks, two white blocks, and three small empty regions. The inner black block has two liberties. The two black blocks enclose the region of five white stones and five empty intersections.

3.2. Groups

A *group* is an important notion that is defined as either a block or a union of blocks in the same colour such that the opponent cannot cut, or separate, the blocks. Although some groups, such as blocks jointed by *kosumi* (diagonal)  and/or *takefu* (bamboo joint)  can be recognized by static analysis, precise recognition needs dynamic analysis. [5]

The *static group* is a group in the narrow sense. We define it as follows. For two blocks X and Y of the same colour, any empty intersection in $ext(X) \cap ext(Y)$ is a *connecting point*. A connecting point is *safe*, if the opponent can neither place a stone at that point nor cut the group at that point. The safe connecting point is either a false eye, or the intersection is such that even if the opponent places a stone at that point, the stone can be captured in the next move. The unsafe connecting points include those of *kou* and snapback (*utte-gaeshi*), which we do not discuss further in this paper.

A black *subgroup* is a set $G \subseteq B_t$ of stones defined by the following rules:

(1) A block is a subgroup.
(2) For a subgroup $G \subseteq B_t$ and a block $X \subseteq B_t$, if both G and X have either one safe connecting point or two or more connecting points, then $G \cup X$ is a subgroup.

The black *static group* is the maximum subgroup G such that there is no subgroup G' with $G \subsetneq G'$. White static groups are defined analogously. The set of blocks connected by *kosumi* and/or *takefu* relations are static groups. For example, all black blocks in Fig. 2(a) form a static group, and two inner white blocks also form a static group. The outer white stones form a group that is not static. Among the connecting points from *a* to *h*, *e* and *g* are safe.

Usually the opponent cannot separate the static group. There is, however, an exception: some block(s) with few liberties of a static group can be captured by *oi-otoshi*, i.e., capturing by causing a shortage of liberties through a sequence of blocks. For example, if a white stone occupies point *c* in 2(b) and White move at *b*, then Black cannot connect point *a* because of the oi-otoshi. Hence, for maintaining groups, it is important to observe the liberties of the groups and its parts (as discussed in Section 3.4).

Intuitively, a group or a block is effectively dead, if the opponent can capture it, and effectively alive otherwise. A group is in determinate life, if either it has two eyes that are small enclosed regions or it is in a seki. A more

[5] It is shown in [17] that a static analysis based on the electric charge model is useful for recognizing groups in the broad sense.

Table 1
Symbols representing states of regions enclosed by a group

T:	the enclosed region can form two eyes
S:	the group enclosing the region can form a seki
O:	the region cannot form two eyes but only one eye
E:	the region cannot form any eye
K_T:	the enclosed region can form two eyes, if the group side wins the *ko*, and one eye otherwise
K_O:	the enclosed region can form one eye, if the group side wins the *ko*, and no eye otherwise

strict definition of determinate life is given by the Chinese rule as described in [2]. A related subject is discussed in Section 5.2. We define effective life of a group as the property that the group side has a proof tree of moves so that the group can be considered to be in the determinate life.

The effective life and death of a group enclosing one region depends on the size and the shape of the region and the positions of the opponent stones in the region. In a case where the opponent is to move, the group enclosing a region of n empty intersections is dead, if $n \leqslant 3$, and alive, if $n \geqslant 8$. When a group enclosing a region cannot form two eyes and cannot be alive by itself, the number of inner liberties is important for a capturing race as discussed in Section 5, where an *inner liberty* is a liberty in the enclosed region, and an outer liberty a liberty other than the inner liberties. In this case, the effective number of inner liberties, which is equivalent to the outside liberties, is different from the size n of the enclosed region with $n \geqslant 4$ as pointed out in [4,14]. The effective number of inner liberties of the region is given by $(n^2 - 3n)/2 + 3 - m$, where m is the number of the opponent stones in the region [14].

Following Berlekamp and Wolfe [2] and Chen and Chen [5], we represent the types of eyes in enclosed regions related to the life and death of groups by pairs $\{\alpha|\beta\}$ of symbols shown in Table 1, where α represents the state, if the group side moves next, and β, if the opponent moves next. The group of type T or S is alive. The combinations we consider in this section are $\{T|T\}$, $\{T|S\}$, $\{T|O\}$, $\{S|O\}$, $\{O|O\}$, $\{T|K_T\}$, and $\{K_T|O\}$, since we limit our discussion to the group enclosing one region. In some cases, the life and death depends on the outer liberties of the group as well as the features of the region. We remark that, in other publications [5,7], $\{T|T\}$ corresponds to 2.0 eyes, $\{T|O\}$ to 1.5 eyes, $\{O|O\}$ to one eye, $\{O|E\}$ to 0.5 eye, and $\{E|E\}$ to an empty eye.

3.3. Incremental computation of liberties of blocks

Two important properties of a block are its size and the number of its liberties. In this subsection, we assume that no capturing occurs by a move at a point A. We will discuss the changes of groups and blocks in the case of capturing in Section 5.2. Consider the case where p is adjacent to blocks $B_1, B_2, \ldots, B_n$, $1 \leqslant n \leqslant 4$ with liberties $L_1, L_2, \ldots, L_n$, respectively, and these n blocks can be unified by placing a stone at p. The incremental computation for the number L of total liberties of the unified block is given by

$$\#liberty(B_1 \cup B_2 \cup \cdots \cup B_n \cup \{p\}, W) = \sum_{j=1}^{n} L_j + L_p - n - c,$$

where $L_p = |ext(\{p\}) - \bigcup_{1 \leqslant i \leqslant n} B_i - W_t|$ is the number of liberties added by placing a stone at p, and c is the number of the liberties that are common to two blocks, and to any block and the point p. For example, consider the case where White connects two blocks by placing a stone at f in Fig. 2(a). The number of liberties of the connected blocks is $(5+4)+0-2-1 = 6$. In the case of placing a stone at point e, the result number equals to $(5+2+4)+1-3-0 = 9$.

When $n \geqslant 2$, the incremental computation is not as efficient as in the case $n = 1$, since the computation needs to check all the liberties of the block(s) to count the common liberties c. We can, however, determine the range of the number L of total liberties by

$$\max_{1 \leqslant j \leqslant n} L_j + L_p - 1 \leqslant L \leqslant \sum_{j=1}^{n} L_j + L_p - n.$$

The lower bound of liberties is determined by the case where one outer block includes all the liberties of the other blocks, and the upper bound the case where the blocks have no common liberties. We can restrict the precise computation of the number of liberties to the case where the number is small and essential to the static analysis.

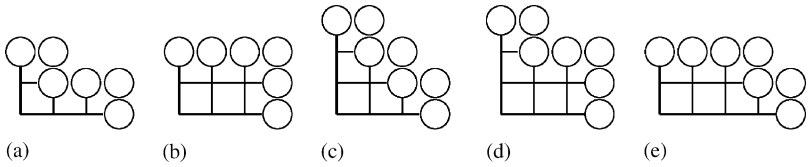


Fig. 3. Patterns of groups and their type when they have no outer liberties: (a) $\{T|K_T\}$; (b) $\{T|O\}$; (c) $\{T|K_T, S\}$; (d) $\{T|S\}$; (e) $\{T|K_T\}$.

3.4. Liberties of groups

Evaluating the liberties of groups is important for analysing capturing races. Let G be a group composed of blocks $B_1, B_2, \ldots, B_n$ $(n \geqslant 2)$ with liberties $L_1, L_2, \ldots, L_n$, respectively. We consider the case where at most one of the blocks has an eye. Let C be the set of connecting points in G, and $C' \subseteq C$ be one of the minimum sets of connecting points such that $\bigcup_{1 \leqslant i \leqslant n} B_i \cup C'$ is a block. Suppose that whenever the opponent (White) moves at a connecting point in $C - C'$, Black should connect two or more blocks by filling some related connecting point in C'. Each safe connecting point should be filled before when the related blocks have no liberties. Accordingly, the set of effective liberties of a group G is given by

$$liberty(G, W_t) = liberty(G \cup C', Wt) - (C - C').$$

Hence, we have the effective number of liberties as follows:

$$\#liberty(G, W_t) = \sum_{i=1}^{n} L_i - \sum_{p \in C} |ext(\{p\}) \cap G| + L_{C'} - F,$$

where $L_{C'} = |ext(C') - G - W_t|$ is the increment of liberties by filling all the connecting points and F is the number of safe connecting points in G. For example, consider the black group in Fig. 2(b). The subset C' of connecting points is either $\{e, f, g\}$ or $\{e, g, f\}$. In both cases, the number of effective liberties of white group is given by $(2 + 5 + 4 + 2 + 4) - (2 + 2 + 2 + 3) + 1 - 2 = 7$.

We note that the effective number of total liberties of a group is fewer than that of a group without safe connecting points (or with false eyes) by the number of the connecting points. The *oi-otoshi* mentioned in Section 3.2 occurs, when the effective number of liberties of some subset of blocks in the group connected by false eyes is equal to one.

3.5. Life and death of groups enclosing one region

This subsection deals with life and death of a group having one empty region. The number of effective eyes of one region enclosed by a group is determined by the size and the shape of the regions. Chen and Chen [5] have shown that the number of eyes of an empty region R can be determined by the following features: the perimeter of R, $|ext(R)|$, and the existence of square $\begin{smallmatrix}+&+\\+&+\end{smallmatrix}$, which is given by $\#adjacent(R) - |R| + 1 \geqslant 1$ in our terminology. Another possible set of features for this recognition is the size of R, $|R|$, the number of adjacent to relations in R, $\#adjacent(R)$, and the maximum number of the neighbours in R, $\max_{p \in R} |R \cap ext(\{p\})|$.

Fig. 3 shows typical empty regions in the corner and their types when the regions have no outer liberties. For example, the *bent four in the corner* (a) is in type $\{T|T\}$, if the white group has two or more outer liberties, and in $\{T|K_T\}$ otherwise. These patterns can be identified by the following features: (1) the number of intersections on the Edge, (2) the perimeters of R, and (3) the number of squares except patterns (d) and (e), both of which have the values 5, 5, and 2, for the three features. Therefore, we need another feature to classify these two patterns. A feature for this is the number of inside intersections in R, $|ext(ext(ext(R))) \cap R|$, whose value is 2 for (d) and 3 for (e).

We tested several sets of features described by the set operations for various patterns of enclosed regions, and found that these features are useful to identify the types of the life and death of the regions with the size of five to eight including those in the corners and those containing opponent stones [17]. A characteristic of this method is that the recognition is based on numerical features of the regions and groups, to which incremental computation can be applied. The method does not use direct pattern matching as used in many Go programs. However, we remark that this method

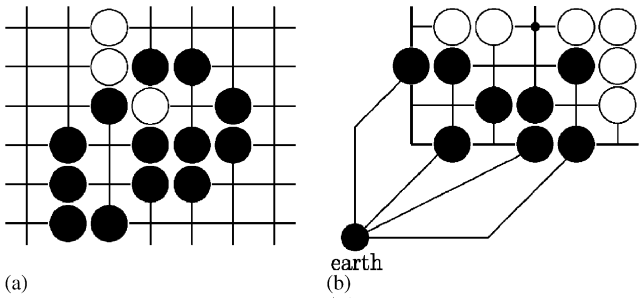


Fig. 4. Black groups enclosing two determinate regions.

is only applicable to the groups enclosing "closed" regions. To analyse patterns with incompletely closed regions, or patterns with half eyes or open eyes, several methods have been proposed such as those by eye values and eye regions in Chen and Chen [5] and Fotland [7] and by position evaluation in Chen [6]. Another method of determining the life and death that can be applied to incompletely closed regions or loosely static groups is described in Section 4.3.

4. Finding the number of enclosed regions based on Euler's formula

The regions enclosed by groups are important for deciding the life and death of the group, since the eyes are enclosed regions and a group needs to enclose a region to be alive. Nakamura [15] first proposed a method of applying Euler's formula to find the number of regions enclosed by groups. In this section, we show an improved method of finding the number of enclosed regions based on incremental computation.

For any connected planar graph, the number N of regions enclosed by edges, or minimal loops, is given by Euler's formula $N = n - k + 1$, where n and k are the numbers of edges and vertices, respectively. This formula has been applied in computer graphics to find the number of enclosed "open regions" in digital figures, which are represented by bit arrays [8]. Euler's formula is also applied to finding "holes" in the game Lines of Action (LoA) [20].

4.1. Finding determinate numbers of enclosed regions

First, we discuss how to apply Euler's formula to finding the determinate number of enclosed regions in Go. We consider each stone in a group as a vertex, and each "link" between the stones as an edge. The *link* is either the "adjacent to" relation or the diagonal relation of two stones in the group.

$$\#link(G) \triangleq \#adjacent(G) + |G \cap \overrightarrow{G} \downarrow| + |G \cap \overleftarrow{G} \downarrow|.$$

Here, we assume that every block in the group is jointed to other block(s) not only by *takefu* but also by diagonal links. This condition is necessary, because Euler's formula is applicable to connected graphs. The restriction will be relaxed in Section 4.3.

The group can contain *closed loops* of stones composed of three stones and three links, i.e., either  or . To find the number of enclosed regions (or the number of open loops), the number of the closed loops $\#closed\_loop(G)$ should be subtracted from the number of minimal loops. The number of regions enclosed by the group G is given by

$$\#empty\_region(G) \triangleq \#link(G) - |G| - \#closed\_loop(G) + 1.$$

A group can contain a closed loop of the form , which contains two diagonal links. In this case, only one of the diagonal links is valid, since Euler's formula applies solely to planar graphs. For example, the black group in Fig. 4(a) has 19 links including 8 diagonal links, 13 stones and 5 closed loops. The number of enclosed regions is given by $19 - 13 - 5 + 1 = 2$.

To apply this method to the groups in the peripherals and corners, we consider that there are links between stones on the edge of the board and a special virtual stone called the *earth* as shown in Fig. 4(b). To find the number of the virtual

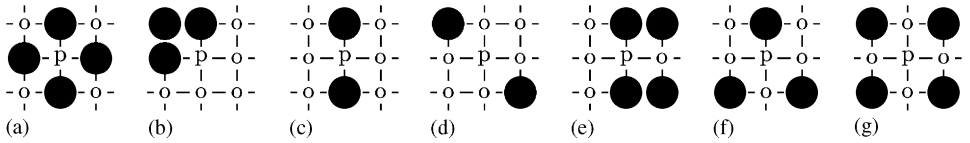


Fig. 5. Typical patterns of neighbours and Euler numbers: (a) −1; (b) 0; (c) +1; (d) +1; (e) +1; (f) +2; (g) +3.

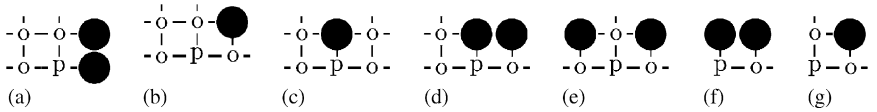


Fig. 6. Typical patterns of neighbours on the edge (a–e) and the corner (f, g) and Euler numbers for earthed groups: (a) 0; (b) +1; (c) +1; (d) +1; (e) +2; (f) 0; (g) +1.

links, we first assign $G \cap \mathcal{E}$, the set of stones on the edge, to a variable X. The number of virtual links is $|X|$, and the number of closed loops with the earth is $|X \cap X\downarrow| + |X \cap \vec{X}|$. We say that the group G is *earthed*, if $G \cap \mathcal{E} \neq \emptyset$. Since the group in Fig. 4(b) has 13 links including 4 virtual links, 9 stones including earth and 3 closed loops, the number of regions is $N = 13 - 9 - 3 + 1 = 2$.

4.2. Incremental computation of the numbers of enclosed regions

For incremental computation of the number of enclosed regions, we consider the change in the number of links and closed loops caused by placing a stone at an empty intersection p in a group G of the same colour. The changes are used for finding effective numbers of enclosed regions in the next subsection. For an empty point $p = (i, j)$ not on the edge, let $C(p)$ be the circular sequence of eight neighbour states,

$$\begin{array}{ccc} S_{i-1,j+1} & S_{i,j+1} & S_{i+1,j+i} \\ S_{i-1,j} & \text{-p-} & S_{i+1,j} \\ S_{i-1,j-1} & S_{i,j-1} & S_{i+1,j-1}, \end{array}$$

where each state $S_{x,y}$ is either empty, a black stone, or a white stone. Let e_p be the number of the consecutive subsequences of either empty states or opponent (white) stones in $C(p)$ such that each subsequence contains one or more elements in $ext(\{p\})$. We call the value $E(p) = e_p - 1$ the *Euler number* of p, which represents the change in the number of empty regions caused by placing a black stone at p. The fact that the change equals $E(p)$ is derived from $E(p) = n' - L'$, where n' is the change in the number of links and L' is the change in the number of the closed loops caused by adding the stone. Fig. 5 shows typical state patterns of neighbours and the increments of the number of regions enclosed by a black group. The symbol o denotes either an empty point or a white stone.

For any empty point p on the edge or the corners, the value e_p is similarly defined as the number of consecutive subsequences in the sequence of the neighbour states from $S_{i-1,j}$ to $S_{i+1,j}$, or from $S_{i,j+1}$ to $S_{i+1,j}$, respectively

$$\begin{array}{cccc} S_{i-1,j+1} & S_{i,j+1} & S_{i+1,j+i} & \qquad\qquad S_{i,j+1} \quad S_{i+1,j+i} \\ S_{i-1,j} & \text{-p-} & S_{i+1,j}, & \qquad\qquad \text{p-} \quad\quad S_{i+1,j}. \end{array}$$

The Euler number $E(p)$ is $e_p - 1$, if the group is earthed and left and right neighbour intersections are empty. Otherwise, the Euler number is $e_p - 2$. In this case, since a stone is placed on the edge, the group becomes earthed. Fig. 6 shows typical patterns of neighbours and Euler numbers when the group is earthed. The pattern (c) represents that the change is one, if the group is earthed, and zero otherwise. Patterns (f) and (g) represent two cases in the corner intersection.

4.3. Finding effective numbers of enclosed regions

By computing the Euler numbers for all the liberties of a static group, we can obtain the effective numbers of enclosed regions. In this method, all blocks in the group need not to be linked together as described in Section 4.1. We call any

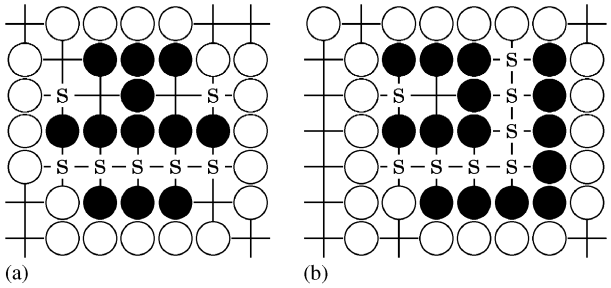


Fig. 7. Black groups enclosing two effective eyes.

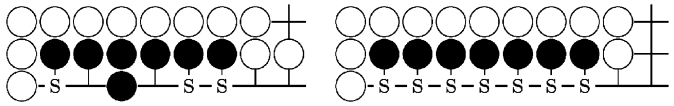


Fig. 8. Black groups enclosing 1.5 effective eyes in the edge.

empty point p in the liberties of a group a *semi-joint*, if it satisfies the following two conditions:
(1) If p is not on the edge, then $E(p) \geqslant 1$.
(2) If p is on the edge, then either $E(p) \geqslant 2$, or $E(p) = 1$ and $|ext(p)| = 1$.
The patterns (c), (d) and (e) in Fig. 5 and (c), (d) and (e) in Fig. 6 represent semi-joints. The semi-joint can be considered as an additional link or as an eye depending on the number of the neighbour semi-joints. Based on the size of the connected set of semi-joints, we classify the semi-joints into the following three cases:
(1) *Single semi-joint*. The single isolated semi-joint p is generally considered as a half additional link and as the point with $E(p) = 0.5$. The single semi-joint on the edge can be equivalent to a half link to the earth. Note that there are cases where even if the opponent moves at the point of the semi-joint, this stone can be captured. In this case, the semi-joint is equivalent to one link.
(2) *Two or three consecutive semi-joints.* The semi-joints are considered as one link and a connecting point.
(3) *A sequence of four or more semi-joints.* The connected set of the semi-joints can be considered as forming eyes. The following table shows the relation between the sizes of connected sets and the numbers of eyes.

Size of the set of semi-joints	4	5–6	7	8–
The number of eyes	0.5	1	1.5	2

Fig. 7 shows examples of groups composed of two blocks. Since the black group in configuration (a) has two single semi-joints, which is equivalent to one effective eye, and a sequence of five semi-joints, which is equivalent to one effective eye, the state of the group is type $\{T|T\}$. The black group in (b) has $0.5 + 1.5 = 2$ eyes. Both of the two groups in Fig. 8 have 1.5 effective eyes, or equivalently type $\{T|O\}$. The black group in Fig. 9(a) has two sets of semi-joints, each of which is equivalent to a link to the earth. If Black moves at point 1 in the configuration (b), then the group has two effective eyes and becomes alive. In contrast, if White moves as a sequence in the configuration (c), then Black can form only one region. This causes a capturing race, which White can win and so can capture the black block.

The semi-joints inside an enclosed region are considered as *vital points* and can be used for determining the life and death of the group. Any group enclosing a region generally has type $\{T|T\}$, if the region has two or more vital points. The group is dead, if only one vital point in the region is occupied by the opponent stone. Hence, the type of the group is $\{T|O\}$, if the region has only one vital point. This method of determining the life and death is useful for many simple cases. There are, however, cases where the group of type $\{T|T\}$ has no semi-joint. For example, a group enclosing the empty region of the form  is alive, but has no semi-joint inside the region.

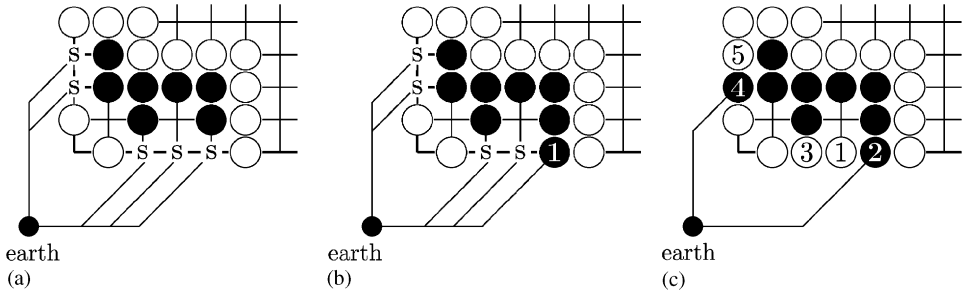


Fig. 9. A black group in the edge: a complex case. (b) Black to move; (c) White to move.

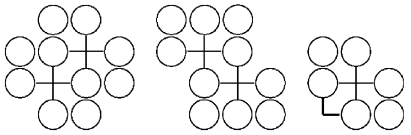


Fig. 10. Groups consisting of two blocks connected by diagonal links.

4.4. Estimating the number of eyes

The number of enclosed regions discussed so far can be used to estimate the number of eyes. Each enclosed region other than the false eye can form one or two eyes, unless the region is so large that the enclosed region can contain an opponent group of a certain size. We can solve this problem by analysing capturing races as described in Section 5.

Although we can recognize a false eye as a local pattern, we have another straightforward method of excluding the false eyes from the number of regions. A false eye occurs, when two blocks are connected by exactly two diagonal links (Fig. 10). Hence, we have the following rules.

Rule 1: The group composed of two blocks has two or more enclosed regions that are not false eyes, if the blocks are connected by three or more diagonal links, respectively.

Rule 2: The group composed of two earthed blocks has two or more enclosed regions that are not false eyes, if the blocks are connected by two or more diagonal links, respectively.

Note that Rule 1 is also useful for recognizing the *dragon with two heads*, i.e., a circular group with two blocks connected by two false eyes. Although the condition for unconditional life by Benson [1] covers these groups, our rule is simpler and more appropriate for incremental computation.

5. Analysing capturing races

In this section, we discuss a method of analysing capturing races and sekis. We intend to apply the analysis to the configurations such that the territories among the groups have been almost determined, and the numbers of liberties of the groups and the sizes of their internal eyes, or at least their ranges, have been determined.

5.1. Confronting groups and semeai graphs

The groups B_X and W_X are in (single) *capturing race*, if:
(1) neither B_X nor W_X can form two eyes;
(2) neither Black nor White can connect B_X and W_X to other black and white groups, respectively; and
(3) if either Black or White captures W_X or B_X, respectively, then a group including B_X or W_X can form two eyes.
When the game progresses preserving the above condition (2), it reaches to a configuration such that B_X and W_X have a determinate property defined as follows. We say that any black group B_X and a white group W_X *confront* each other

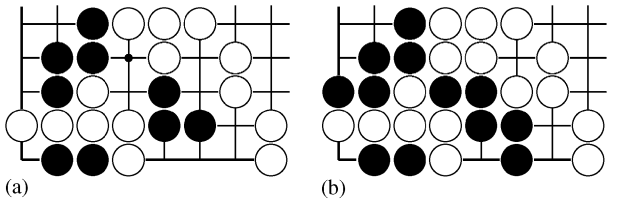


Fig. 11. Effectively confronting blocks (a) and a determinate configuration of confronting blocks (b).

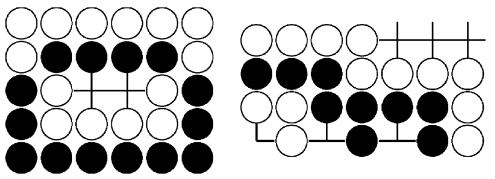


Fig. 12. Two typical determinate seki situations.

(by a buffer or by adjacency):
(1) if there is an empty region R, called the *buffer*, that is adjacent to both B_X and W_X, then $ext(R) \subseteq B_X \cup W_X$;
(2) otherwise, B_X and W_X are adjacent to each other.

For example, two black groups in Fig. 11(a) are in a capturing race and effectively confronting each other. The figure (b) shows a determinate configuration of confronting groups. Note that the crosscut (*kiri-chigae*) of the forms  and  plays an important role in determinate confronting relations.

Two groups B_X and W_X confronting each other are in a *seki*, if both B_X and W_X are alive in the sense that neither side can capture B_X or W_X.

A *semeai graph* is a labelled graph satisfying the following conditions:
(1) Each vertex corresponds to a group. The label of the vertex is a 3-tuple (C, S, L), where C is the colour of the group, S is the size of the group, and L is the number of the internal liberties. Every live group is assumed to have infinite internal liberties.
(2) Each edge represents that the corresponding black and white groups confront each other. The label of the edge is a pair $L_B : L_W$, where L_B and L_W are the numbers of liberties of the black and white groups, respectively, in the buffer(s). If the two groups have no buffer, then $L_B = L_W = 0$.

The labels (C, S, L) of vertices are represented by the colours of the nodes and the marks of the form $S(L)$ in the diagrams. From the semeai graphs, we can determine the number of liberties of each group by the sum of the size of its eye and the liberties in the buffer(s).

The simple seki situations as shown in Fig. 12 have the following two semeai graphs, where $D_1 \geqslant 2$ and $D_2 \geqslant 1$:

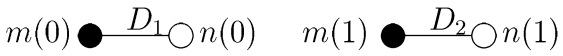


Fig. 13 shows a configuration having a capturing race and its semeai graph. The black group (block) confronts both the inner white group and the outer white group. The label ∞ of the right white group means that the outer white group is alive. The label 8(0) represents that the black group has 8 stones and no internal liberties. We represent the label of the lower edge simply by 2 instead of 2 : 2. The state of the black group is $\{T|S\}$: If Black moves next, it can capture the inner white four-stone block and make the black block alive, whereas if White moves next, it can make the inner white and black blocks a seki.

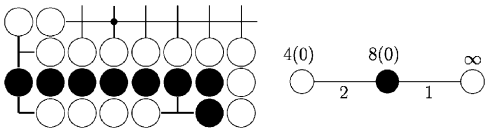


Fig. 13. An example of a simple capturing race and its semeai graph.

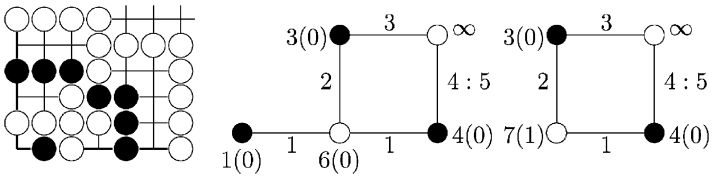


Fig. 14. A double capturing race and its semeai graphs.

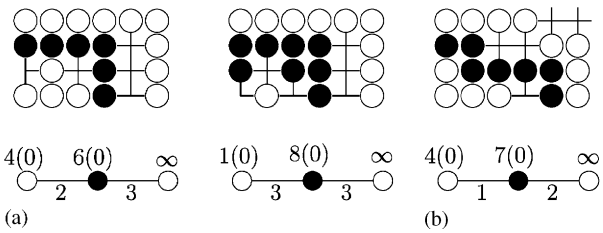


Fig. 15. Two cases of determining the life and death: (a) The black block is dead; (b) a difficult case.

Fig. 14 shows another example of a capturing race and its semeai graphs. The left semeai graph is the direct representation of the board. This configuration contains double capturing races. We can reduce this semeai graph to the right graph, in which the inner white block is considered to absorb the dead black block and to have internal liberties. The multiple capturing races are discussed in Section 5.4. [6]

5.2. Life and death and capturing races

In general, when a group captures a confronting group of a sufficient size, it becomes alive. [7] However, if the captured group is small then the capturing groups might not be alive. The life and death of the capturing block depends on the size and the shape of the captured block. Any separate group enclosing an empty region with L intersections is dead, if $L \leqslant 3$, and the group is alive if $L \geqslant 8$, when the opponent moves next as described in Sections 3.2 and 3.5. For example, although Black in Fig. 15(a) can capture the inner white block, White can make the black block dead after capturing the white block as shown in the right figure.

Fig. 15(b) shows a more difficult case. After Black has captured the white block in the corner, the resultant empty region is of the form "*bent four in the corner*." If Black is to play, Black can form two eyes by *oshi-tsubusi*. Otherwise, if White reduces the outer liberty of the black block, Black can only make the group a ko.

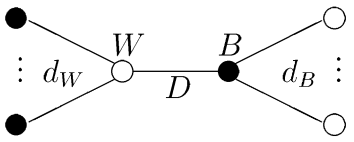
5.3. Single capturing races between two confronting groups

A *single* capturing race is a situation of black and white groups B and W such that we can determine whether B or W captures the other or both are in a seki and alive. Two groups confronting each other can be generally represented by

[6] Black cannot win this capturing race, which is an example of Case (8) in Section 5.4.

[7] There is an exceptional case, *ishi-no-shita*, where the captured side can recapture one of the blocks that enclosed the captured block.

the following semeai graph.



where, W and B represent the inner liberties, and d_W and d_B are the sums of all liberties for all edges from W and B, respectively. We assume that neither the black group B nor the white group W is alive and the external black (left) and white (right) groups are alive. There are three cases that are characterized as follows:

(1) $W = B = 0$.
White can capture the black group, if White is to play and $d_W \geqslant D + d_B - 1$. Similarly, Black can capture the white group, if Black is to play and $d_B \geqslant D + d_W - 1$. From the complement of these relations, we obtain the following condition for the seki:

$$d_W \leqslant D + d_B - 2 \quad \text{and} \quad d_B \leqslant D + d_W - 2.$$

This implies a necessary condition for a seki, $D \geqslant 2$.

(2) $W \geqslant 1$ and $B \geqslant 1$.
White can capture the black group, if White is to play and $d_W + W \geqslant D + d_B + B$. Whereas, Black can capture the white group, if Black is to play and $d_B + B \geqslant D + d_W + W$. Neither Black nor White can capture the confronting group, if

$$d_W + W \leqslant D + d_B + B - 1 \quad \text{and} \quad d_B + B \leqslant D + d_W + W - 1.$$

These two relations and their consequence, $D \geqslant 1$, are necessary conditions for the seki.

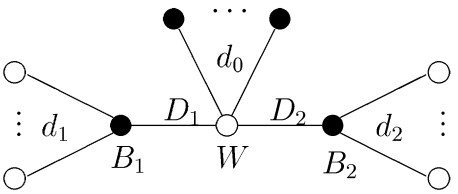
(3) $W = 0$ and $B \geqslant 1$.
In this case, White can capture the black group, if White is to play and $d_W \geqslant D + d_B + B$. Black can capture the white group, if Black is to play and $d_W \leqslant D + d_B + B$. In this case, no seki is formed. This situation is called *me-ari me-nashi*, meaning that the group that has an eye has the advantage over the group that does not.

5.4. Double capturing races

A *multiple* capturing race of three or more groups is a situation such that the life and death of each of the groups depends on all the groups. For the analysis of a multiple capturing race, we first assume that all the capturing races are single and then combine the local results to determine the life and death of the groups. We use the following notations for the relations in the single capturing race between two groups X and Y with different colours, which confront each other, and each of which is not alive by itself. All the external groups of X and Y are assumed to be alive

$X \Rightarrow Y$: X can capture Y but Y cannot capture X.
$X \Leftrightarrow Y$: X and Y are in a seki.
$X \leftrightarrow Y$: Either X or Y which moves next can capture the opponent Y or X, respectively.

We consider the double capturing races of three groups, which can be generally represented by the following semeai graph



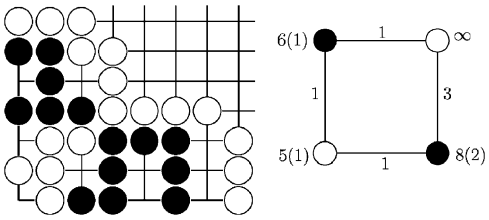
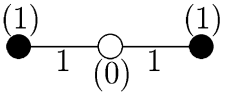


Fig. 16. A double capturing race with the seki collapse.

The labels B_1, W and B_2 of the three groups also represent the numbers of inner liberties. We assume that none of B_1, W, and B_2 is alive, and that all the external groups are alive. Considering the symmetry of the colours, there are the following nine cases of the combinations. [8]

(1) $B_1 \Leftarrow W \Rightarrow B_2$: Obviously, White can capture two black groups B_1 and B_2, since it has the largest liberties. The results of the two single capturing races are preserved in the double capturing races in this case.

(2) $B_1 \Rightarrow W \Leftarrow B_2$: Although Black can capture the white group W in most cases, there is an exception with the graph 

, where the three blocks are in a seki, which is a linear form of seki described below.

(3) $B_1 \Rightarrow W \Rightarrow B_2$: From the two relations, $d_1 + B_1 \geqslant W + d_0 + D_2 + D_1 + 1$ and $W + d_0 + D_1 \geqslant B_2 + d_2 + D_2 + 1$, we have

$$d_1 + B_1 + D_1 > W + d_0 + D_1 + D_2 > B_2 + d_2 + D_2.$$

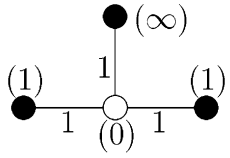
This means that the total number of liberties of B_2 is the smallest, and W captures B_2. If the white group is not alive after absorbing B_2, there remains the single capturing race between B_1 and the white group. Otherwise, White also captures B_1.

(4) $B_1 \Leftrightarrow W \Leftrightarrow B_2$: The assumption of the single race that either B_1 or B_2 is alive does not hold. White wins the race and captures both B_1 and B_2.

(5) $B_1 \Leftrightarrow W \Rightarrow B_2$ and $B_1 \Leftrightarrow W \leftrightarrow B_2$: White can capture B_2 and then capture B_1. This situation is called a seki collapse (*seki-kuzure*).

(6) $B_1 \Leftrightarrow W \Leftarrow B_2$: Since Black cannot fill the liberties between B_1 and W, Black cannot capture W. In contrast, White can capture B_2 by filling the remaining liberties of B_2, and then capture B_1 by the seki collapse.

(7) $B_1 \leftrightarrow W \leftrightarrow B_2$: If the outer liberties of White d_W is not zero, Black or White to move next captures the opponent. Fig. 14 shows an example of this capture. There is, however, an exception with the following semeai graph.



In this case, Black can only make the three groups a seki by reducing the outer liberty of White to zero, whereas White can capture either B_1 or B_2, and hence both of the Black groups. This is a case of the linear form of a seki described in the next subsection.

(8) $B_1 \leftrightarrow W \Rightarrow B_2$: White can capture B_2, and thus also B_1. Fig. 16 shows an example of this.

(9) $B_1 \leftrightarrow W \Leftarrow B_2$: Either Black or White which moves next captures the opponent.

5.5. Capturing races among three or more groups

The analysis for the three groups can be extended to include four or more groups. In the analysis, it is desirable that the life and death of a group is determined by local relations between the group and its neighbour groups. There are,

[8] Note that in many cases the double capturing races are not a simple combination of two single capturing races.

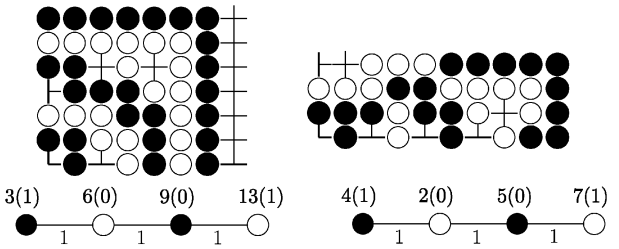


Fig. 17. Linear forms of sekis.

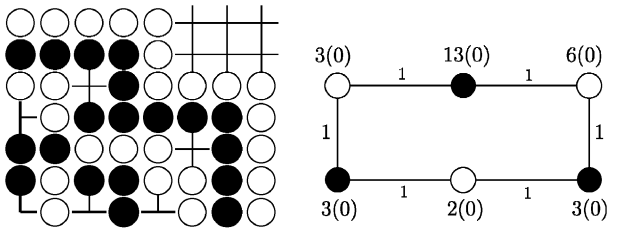


Fig. 18. A circular form of a seki.

however, cases where this condition is not satisfied because of the large seki that contains an unrestricted number of blocks. The relations for four groups are classified into the linear form, the circular form and the "T" form. The linear form of relations such as $B_1 \Rightarrow W_1 \Rightarrow B_2 \Rightarrow W_2$ and $B_1 \leftrightarrow W_1 \leftrightarrow B_2 \leftrightarrow W_2$ can be analysed similarly as above with an exception: the case where $B_1 \Rightarrow W_1 \leftrightarrow B_2 \Leftarrow W_2$ includes the following graph representing a linear form of a seki:

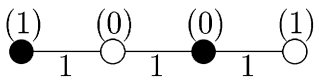
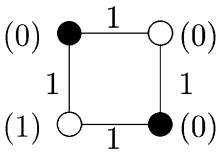


Fig. 17 shows two examples of sekis with the linear form. There are variations of this seki such that one or more of the inner blocks is a chain of blocks connected by false eyes.

The large sekis include the circular form of sekis. The case where $B_1 \leftrightarrow W_1 \leftrightarrow B_2 \leftrightarrow W_2$ and $W_2 \leftrightarrow B_1$ includes a special circular form of the seki of the following form:



This form also includes a large seki with an unrestricted number of blocks. An example of the seki of the circular form is shown in Fig. 18.

For the capturing race of four groups in the "T" form as shown below, we can show that White can capture all of the black groups B_1, B_2 and B_3, since White has the largest liberties and hence is predominant as in the double capturing races.

$$\begin{array}{ccc} B_2 & B_2 & B_2 \\ \updownarrow & \Uparrow & \Downarrow \\ B_1 \leftrightarrow W \leftrightarrow B_3 & B_1 \Rightarrow W \Rightarrow B_3 & B_1 \Rightarrow W \Rightarrow B_3 \end{array}$$

The other capturing races of four or more groups can be similarly analysed by extending the method for three groups.

5.6. A procedure for analysing semeai situations

For constructing the semeai graph, we need to find (1) the pairs of groups that cannot be connected, (2) the buffers between black and white groups, and (3) the pairs of groups that confront each other. The buffers can be recognized as special empty regions and as incompletely closed regions by the methods described in Sections 3.2 and 4.3. The liberties of the groups are found by the method shown in Section 3.4.

The considerations in the previous subsections suggest a procedure for analysing capturing races and for determining the life and death of the groups based on the semeai graphs as follows.

Step 1: For a given semeai graph, if all the groups are determined to be alive or dead then terminate. Note that the live group can be a long chain of groups in a seki. For any dead group, delete the corresponding vertex from the semeai graph, change the semeai graph so that the neighbour vertex (or vertices) absorbs the dead group.

Step 2: Assume that every capturing race is single, and determine which of the relations $\Rightarrow$, $\Leftrightarrow$ and $\leftrightarrow$ holds between black and white groups for each edge in the semeai graph.

Step 3: For the single capturing races, the result is obtained. Determine whether the capturing group is alive or dead by examining the size and the shape of the captured block. Go to Step 1.

Step 4: For capturing races with three or more groups, combine the two or more successive relations obtained in Step 2 and determine which group(s) wins the race as described in Sections 5.4 and 5.5. Determine whether the capturing group is alive or not by examining the size and the shape of the captured block. Go to Step 1.

Incremental computing of this procedure is not difficult, since we can determine the change of the semeai graph and its influence.

6. Concluding remarks

In this paper, we discussed some formal models and incremental computation to be used in the static analysis in Go programming. We proposed several methods of analysing the position on the Go board including:

- Determining the life and death of a group enclosing a region based on the numerical features and not on direct pattern matching.
- Finding the number of regions enclosed by a group based on Euler's Formula and incremental computation.
- Analysing capturing races and sekis by semeai graphs.

Most notions and the methods for the static analysis are mathematically defined by the operations on sets of intersections. In Sections 3 and 4, the incremental computation is used to combine the formal models and the practical computation methods.

We have not sufficiently discussed the ko in this paper. In the case where the life and death of a group or a connection between the groups is related to a *ko*, we can generally divide the analysis into two cases of winning and losing the ko. With better ko descriptions we can be more precise and general so that the analysis covers the ko and partial searches as described in [7].

We discussed several methods of incremental computation for static analysis. It is still an important subject to investigate further the efficiency and the restrictions of incremental computation. Other future problems and works include the following.

- Investigating the relations between incremental computation and object-oriented approach for maintaining objects such as groups and the semeai graphs, and an application of incremental computation to the pattern matching, which is a common mechanism in Go programming.
- Developing a method of combining the static analysis methods in this paper with the methods based on estimating influences of stones such as those by potential functions.
- Developing a method of combining the static analysis with the search in dynamic analysis, which is not based on the direct game tree of board configurations but on higher-order descriptions of the phases to classify these two patterns.

Acknowledgements

The author would like to thank Professor H. Jaap van den Herik and Professor Hiroyuki Iida for their encouragement and help for his research in Go programming, and Dr. Erik van der Werf and the anonymous referees for their helpful comments. He also would like to thank Hiroyuki Ohtsuka, Ayumi Konno, and Masashi Agio for their help in implementation.

References

[1] D.B. Benson, Life in the game of Go, Inform. Sci. 10 (1976) 17–29.
[2] E. Berlekamp, D. Wolfe, Mathematical Go—Chilling Gets the Last Point, A.K. Peters, Ltd., 1994.
[3] B. Bouzy, Incremental updating of objects in Indigo, in: Fourth Game Programming Workshop, Hakone, Japan, 1997, pp. 179–188.
[4] K. Chen, Computer Go: knowledge, search, and move decision, ICGA J. 24 (2001) 203–215.
[5] K. Chen, Z. Chen, Static analysis of life and death in the game of Go, Inform. Sci. 121 (1999) 113–134.
[6] Z. Chen, Semi-empirical quantitative theory of Go—Part 1: estimation of the influence of a wall, ICGA J. 25 (2003) 211–218.
[7] D. Fotland, Static eye analysis in "The Many Faces of Go", ICGA J. 25 (2003) 203–210.
[8] S.B. Gray, Local properties of binary images in two dimensions, IEEE Trans. Comput. C-20 (1971) 551–561.
[9] R. Hunter, Counting Liberties and Winning Capturing Races, Slote & Shell, 2003.
[10] H. Iida, Konpyu-ta ha meijin wo koerarreruka (Can a Computer Defeat the Shogi Champion?) Iwanami Shoten, 1992 (in Japanese).
[11] K. Klinger, D. Mechner, An architecture for computer Go, < http://www.cns.nyu.edu/mechner/compgo/acg/>, 1996.
[12] H. Landman, Eyespace values in Go, in: Games of No Chance, Cambridge University Press, Cambridge, 1994, pp. 227–257.
[13] M. Müller, Races to capture: analyzing semeai in Go, Game Programming Workshop in Japan '99, IPSJ Symp. Series, Vol. 99 (14), 1999, pp. 61–68.
[14] M. Müller, Conditional combinational games and their application to analyzing capturing races in Go, Inform. Sci. 154 (2003) 189–202.
[15] K. Nakamura, Graph theoretic analyses of Go board phases, in: H.J. van den Herik, H. Iida (Eds.), Games in AI Research, University Maastricht/University of Shizuoka, 2000, pp. 239–249.
[16] K. Nakamura, Analyzing capturing races and seki situations, in: H.J. van den Herik, B. Monien (Eds.), Advances in Computer Games 9, 2001, pp. 295–311.
[17] K. Nakamura, S. Kitoma, Static analysis by incremental computation in Go programming, in: H.J. van den Herik, H. Iida, E. Heinz (Eds.), Advances in Computer Games 10, Many Games, Many Challenges, 2003, pp. 175–192.
[18] E.C.D. van der Werf, AI Techniques for the game of Go, Ph.D. Thesis, Universiteit Maastricht, The Netherlands, 2005.
[19] E.C.D. van der Werf, H.J. van den Herik, J.W.H.M. Uiterwijk, Learning to score final positions in the game of Go, in: H.J. van den Herik, H. Iid, E.A. Heins (Eds.), Advances in Computer Games 9, Many Games, Many Challenges, 2003, pp. 143–158.
[20] M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, The quad heuristic in Lines of Action, ICGA J. 24 (2001) 3–15.
[21] S.-J. Yan, S.-C. Hu, A positional-judgment system for computer GO, in: H.J. van den Herik, B. Monien (Eds.), Advances in Computer Games 9, 2001, pp. 313–326.