Playing it Safe: Recognizing Secure Territories in Computer Go by Using Static Rules and Search

Martin Müller
Informatik, ETH Zürich
mueller@inf.ethz.ch

Abstract

Determining whether stones are safe from capture, and whether territories are protected against an invasion, are two fundamental tasks in Go evaluation.

In general, safety cannot be proven by static evaluation; it must be decided by searching. However, good static evaluation functions can greatly speed up a search by making early evaluation possible. In addition, static analysis can decompose a search problem into independent subtasks, which yields an exponential speedup.

The new algorithms for proving safety developed in this paper are closely related to standard Life&Death search methods. However, there are two important differences. First, searches are local: they are restricted to a single enclosed region. Second, the search space for proving the safety of large territories is potentially huge. Therefore powerful static evaluation and search decomposition becomes very important.

1 The Safety of Blocks and Territories

In Computer Go programs, the safety of blocks and territories is usually estimated by a heuristic evaluation function enhanced by tactical searches. Heuristics, often based on an influence function, decide whether it is necessary to make a defensive move. In dangerous looking cases, most programs will add a move, since losing one point is usually less problematic than risking the safety of stones or territory. Deciding whether it is really necessary to defend a territory can be very difficult. For example, in Figure 1, White must defend her territory on the right side at *a*, but the territory on the left side at *b* is safe.

The safety of stones is closely tied to the safety of adjacent territories. However, stones can be safe even if the adjacent territory is not: In Figure 2, the black stones are safe, but the area is not territory because White can still make *seki* by playing at *a*. Heuristics cannot be expected to reliably decide such problems. Search seems the only way to resolve them in general. This paper presents some first steps in that direction.

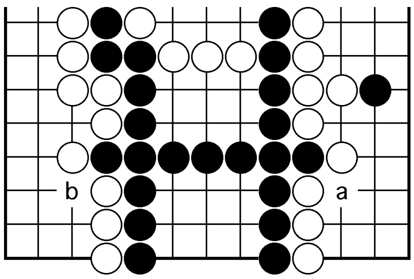

Figure 1: Safe (b) and unsafe (a) stones and territories

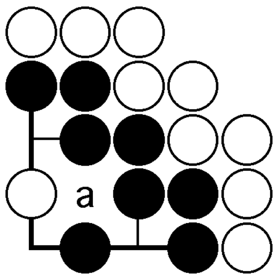

Figure 2: Safe stones, unsafe territory

1.1 Rules, Blocks, Regions

The algorithms in this paper are not very sensitive to the particular kind of rules used. However, suicide is forbidden, since otherwise not even Benson's criteria for unconditional safety are valid [1; 3].

The following terminology for blocks and regions is standard. A *block* is a maximal connected set of stones on the Go board. Each block has a number of adjacent empty points called *liberties*. A block that loses its last liberty is *captured*, i.e. removed from the board.

A *region* is a connected set of points on the Go board which is surrounded by blocks of the same color. A block is called *enclosing block* of a region if it has at least one adjacent point that belongs to the region, and at least one adjacent point that does not belong to the region. The color of enclosing blocks is called the *defender*, the other color is called the *attacker*.

The *interior* of a region is the subset of points not adjacent to an enclosing block. There may be both attacker and defender stones in the interior. Figure 3 shows a 15 point region surrounded by enclosing blocks a, b and e. The interior consists of four points marked $\diamond$. Some of the interior is occupied by stones of blocks c and d.

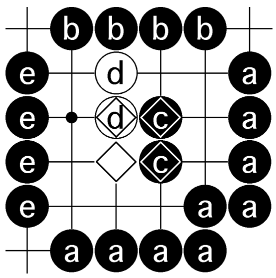

Figure 3: Blocks and regions

2 Procedures for Proving Safety

The following hierarchy of increasingly powerful procedures allows more and more general methods for proving the safety of blocks and territories:

- Unconditional safety (Benson)
- Safety by locally alternating play
- Nonlocal safety by two sure liberties
- Safety by other means

Benson's classic definition of blocks that are unconditionally alive [1] provides a starting point for finding safe blocks and territories. It finds sets of blocks that are safe even if the attacker can play an unlimited number of moves in a row. All blocks and regions on the 5×5 board in Figure 4 are unconditionally safe for Black.

This paper mainly deals with proving safety by *locally alternating play*. In this case, safety can be ensured by playing each region independently, letting the attacker move first in each region and also yielding all ko fights to her. Figure 5 shows an example where Black is not unconditionally alive, but safe by locally alternating play.

Figure 6 shows an example of *nonlocal safety by two sure liberties*. Black cannot survive by locally alternating play. He must reply to White's attack at a by playing b in a different region.

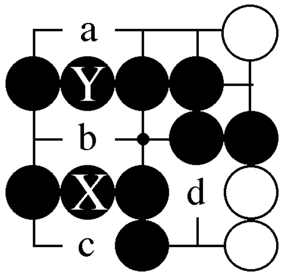

Figure 4: Unconditional safety

Finally, there are several possibilities to survive without two sure liberties. In double ko, as in Figure 7, both players dynamically retain two liberties by capturing single stones. However, neither player can safely keep a liberty at a or b.

It can also happen that stones are in atari, but the attacker does not gain anything by capturing them. The most frequent case is snapback, shown in Figure 8. More complex situations where stones could be captured, but it is not profitable to do so, are given in texts on rules such as Ikeda [2]. If the safety of stones depends on winning a ko fight, they are not considered safe here.

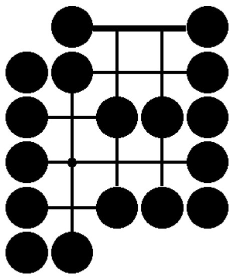

Figure 5: Safety by locally alternating play

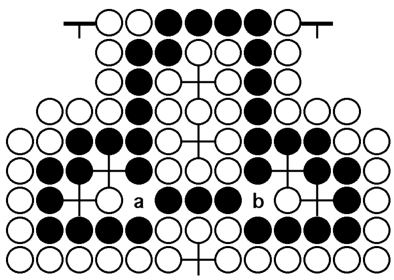

Figure 6: Nonlocal safety

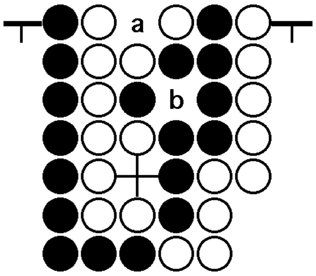

Figure 7: Safety by double ko

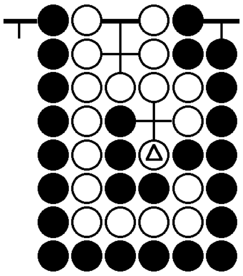

Figure 8: Safe single stone in atari

2.1 Benson's Unconditional Life

The following definition of unconditional life in terms of liberty counts is equivalent to the one in Benson's classical paper [1]. The reformulation is useful for the extensions of the method in subsequent sections.

A *small color-enclosed region* either has no interior, or its interior is filled with attacker stones. In Figure 9, region a is *small* in the diagram on the right but not *small* in the diagram on the left. In both cases, the interior of region a consists of the single point in the top right corner.

A small color-enclosed region is called *healthy* for a block if the block is adjacent to all empty points of the region. In Figure 9 in the left diagram, regions b, c and d are healthy for block X, but only region b is healthy for block Y. In the right diagram, a is also healthy for Y.

Given a set B of blocks, a region r is vital to a block $b \in B$ if

- r is healthy for b
- All enclosing blocks of r are contained in B

Benson's *Sure Liberty Count* of a block b in a region r with respect to a set of blocks B is defined as

- $SLC_{Benson}(b, r, B) = 1$ if r is vital to b in B
- $SLC_{Benson}(b, r, B) = 0$ otherwise

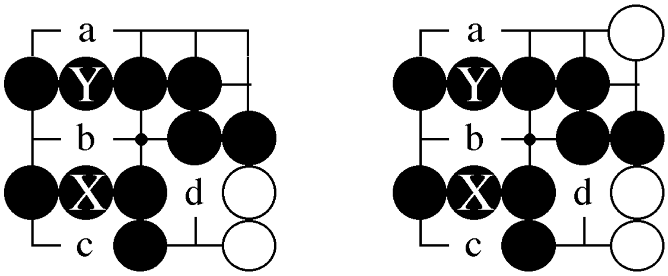

Figure 9: Small and healthy regions

Definition 1:

A set B of blocks is called *unconditionally alive* in a set of regions R if

$$\forall b \in B \sum_{r \in R} SLC_{Benson}(b, r, B) \geq 2$$

3 Safety under Alternating Play

In contrast to the previous discussion, the topic of the following sections are blocks and territories that are safe under alternating play. The algorithms are *local*, they analyze one region at a time. Both static and search-based characterizations of safe blocks and territories are presented.

The algorithms find blocks that will eventually achieve two sure liberties. During play, the liberty count of these blocks might go to 1, but it will ultimately be 2 or more.

Regions can provide either one or two liberties for an enclosing block. The *Sure Liberty Count* of a block b in a region r is computed under the assumption of alternating play, with the attacker moving first and winning all ko fights. This work generalizes the notions of 1-vital and 2-vital regions introduced in [3].

Definition 2:

$SLC(b, r, B) = k,\ k = 1 \ldots 2,$ if b is an enclosing block of r and there is a strategy for defender that guarantees the following invariant:

- Either all blocks enclosing r have at least k liberties in r,
- or
  - some blocks have $k - 1$ liberties
  - and it is the defender's turn

This definition implies that defender can regain k liberties for all his blocks enclosing r with his next move in r. With this definition, there is a simple definition of life under alternating play which is analogous to Benson's:

Definition 3:
A set B of blocks is *alive under alternating play* in a set of regions R if

$$\forall b \in B \sum_{r \in R} SLC(b, r, B) \geq 2$$

Note that this ensures that blocks will never be captured: by construction, each block in B has at least one liberty overall after any attacker move and two liberties after the defender's local reply. An example is shown in Figure 5.

3.1 Extension to Chains

Unfortunately, definition 3 does not work very well in practice, since blocks are often only adjacent to a region that provides one sure liberty. These blocks obtain their second liberty by connecting to another block that has more sure liberties elsewhere. Figure 10 shows a typical example. Definitions 4 and 5 cover chains of safely connected blocks instead of single blocks.

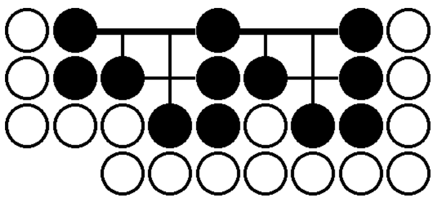

Figure 10: Definition 3 does not suffice to prove safety

Definition 4: Chains and the Chain Condition
A *chain* is a set of blocks and a set of chain conditions for connecting the blocks together.

A strategy for defender in region r fulfills the *chain condition* for a set of defender blocks C if it guarantees the following invariant:

Either all elements of C have been merged into a single block,

or it is defender's turn

or all blocks in C have at least one liberty in r.

Note that the chain condition is independent of the problem of finding sure liberties. Blocks can be connected by a chain condition in one region, but have their sure liberties in other regions. However, all chain conditions and all sure liberties must be maintained simultaneously.

Definition 5:
Let a set B of blocks be partitioned into a set of chains

$$C_B = \{c_1, \ldots, c_n\}, c_i \subseteq B$$

Let the sure liberty count of a chain c in a region r be defined as

$$SLC(c, r, B) = \max_{b \in c} SLC(b, r, B)$$

where the computations of $SLC(b, r, B)$ are restricted to strategies that maintain all chain conditions of c in r. B is called *alive under alternating play* in a set of regions R if there exists a partition C_B of B such that

$$\forall c \in C_B \sum_{r \in R} SLC(c, r, B) \geq 2$$

4 Static Recognition of Sure Liberties

This section develops some static criteria for recognizing one or two sure liberties for enclosing blocks in a region. The definitions are extensions of those given in [3], pp. 62-64.

4.1 Auxiliary Definitions

The following terms will be used for the static recognition of sure liberties. The *accessible liberties of a region* $AL(r)$ is the set of liberties of all enclosing blocks in the region. Given a region r, the *attacker's surroundable area* $ASA(r)$ is the set of interior points that are not occupied by the attacker. A point p in a region is called a *potential attacker eye point* if the attacker could make an eye there, provided the defender passes locally. The set of potential attacker eye points is called the *attacker's eye area*. It is a subset of the attacker's surroundable area.

An *intersection point* of a region is an empty point p such that $region - \{p\}$ is not connected and p is adjacent to *all* enclosing blocks. Figure 11 shows some examples.

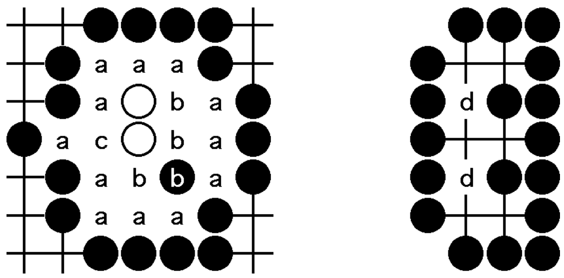

Figure 11: Accessible liberties (a), potential attacker eye points (b), attacker's surroundable area (b+c), and intersection points (d)

4.2 Static Recognition of One Sure Liberty

A region r with exactly one enclosing block has $SLC \geq 1$ under following conditions:

1. each defender block in $ASA(r)$ can be uniquely assigned two adjacent accessible liberties.
2. After step 1, all liberties of interior defender blocks are added to the set of accessible liberties.
3. All empty points in $ASA(r)$ which are not adjacent to a defender block can be uniquely assigned two adjacent accessible liberties.

Uniquely assigned means that in the whole process, each liberty can only be used once. The result of the assignment is a set of triples (p, l_1, l_2), where p is a defender block or an empty interior point and l_1 and l_2 are liberties.

The following simple *miai strategy* ensures one liberty for the defender: If the attacker occupies an empty point p in a rule (p, l_1, l_2), the rule is deleted and no action is taken. If the attacker plays l_1 or l_2 in a rule (p, l_1, l_2), the defender responds at the other liberty (l_2 or l_1), and the rule is deleted. Following this strategy ensures that the attacker cannot surround any of the interior points, therefore the defender keeps a liberty in the region.

Figure 12 illustrates the construction of a miai strategy. The diagram on the left shows a region and its interior. The diagram on the right shows a miai strategy that provides 1 sure liberty. In step 1, moves a_1 and a_2 provide a connection for block a. In step 2, the two liberties e_1 and e_2 of block a are added to the accessible liberty set. In step 3, two accessible liberties are found for the remaining interior points b, c, d and e. The complete strategy is given by the set of triples

$$(a, a_1, a_2), (b, b_1, b_2), (c, c_1, c_2), (d, d_1, d_2), (e, e_1, e_2)$$

Benson's healthy regions correspond to the special case where $ASA(r)$ is the empty set and the miai strategy consists of an empty set of triples.

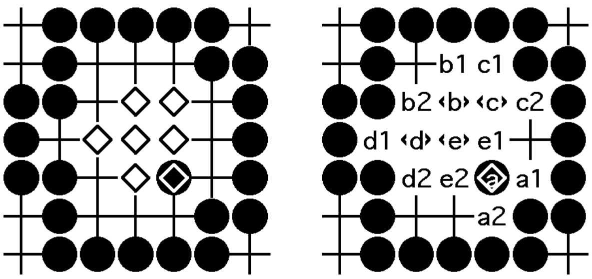

Figure 12: One sure liberty

More Than One Surrounding Block

In the case of a region with more than one surrounding block, blocks are joined by *miai connection strategies* before testing condition 1 above. Such a strategy is just a pair of liberties (l_1, l_2), where defender responds to an attacker move on one liberty by taking the other. In Figure 13, three miai strategies (a_1, a_2), (b_1, b_2) and (c_1, c_2) connect the four enclosing blocks.

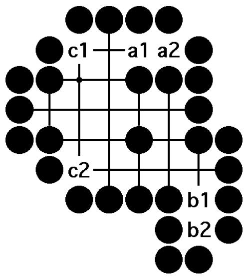

Figure 13: Connecting enclosing blocks

Longer Connection Chains for Interior Blocks

Condition 1 above can be generalized in the following way: blocks for which a miai connection strategy has been found are merged into a chain, and the whole process is repeated, respecting the liberties used so far. The condition evaluates to true if all enclosing and interior defender blocks are eventually connected by miai strategies. Figure 14 shows some examples.

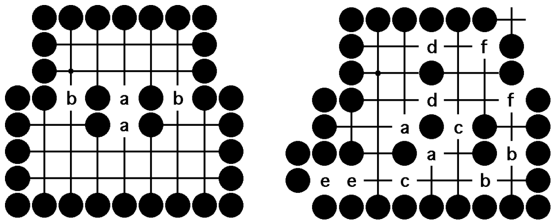

Figure 14: Connecting many interior blocks

4.3 Static Recognition: Two Sure Liberties

Definition 6:

A region r has $SLC(b, r) \geq 2$ for all its enclosing blocks b if all empty points in the region are accessible liberties and if the region has two *intersection points*. Each such region can be split into two regions r_1, r_2 with $SLC(b, r_1) \geq 1$ and $SLC(b, r_2) \geq 1$ by playing a miai strategy on the two intersection points. Figure 15 shows two examples.

5 Search for Sure Liberties

Search for both one and two sure liberties makes use of the static recognition methods developed in the previous section. Search may be constrained by chain conditions on some sets of blocks. Players move alternately in the same region. The attacker moves first and wins all ko fights. Play ends as soon as the static evaluation can decide whether one or two liberties can be reached locally by defender.

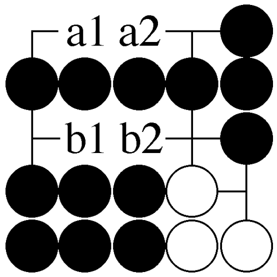

Figure 15: Two sure liberties

The search procedures use standard game tree search techniques: alpha-beta minmax search with iterative deepening, a transposition table and standard enhancements. The only task-specific procedures are move generation and evaluation.

5.1 Move Generation

Move generation is currently very simple: all legal local moves, plus a pass move, are generated. In addition, attacker is allowed to immediately recapture a ko. Move generation could easily be improved by recognizing forced moves during the search, such as answering when a block's local liberty count decreases critically.

5.2 Evaluation

Evaluation returns three possible values: *success*, *failure* and *unknown*. Success for the defender is decided by the static evaluation. Defender fails if a boundary block loses its last local liberty, or if a chain condition is violated. If the last two moves were both passes, the real local liberty count is used as the sure liberty count. In all other cases, the value *unknown* is returned by evaluation. Success and failure of the attacker are computed by negating the defender's evaluation.

6 Proving the Safety of Regions

The static methods of sections 2.1 and 4 simultaneously prove the safety of blocks and regions. When sure liberties are proven by search, as in section 5, the safety of regions does not follow automatically: a region can provide sure liberties for an enclosing block because the attacker does not have a good move. This can happen even if the liberties are shared with the attacker in *seki*. In Figure 16, both black blocks have two sure liberties locally. However, it does not follow that the area is black territory. If Black does not have further sure liberties on the outside, it is a seki and the two liberties are also sure for the white block inside.

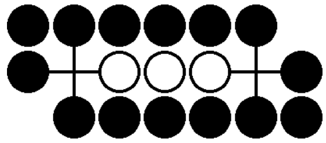

Figure 16: Two sure liberties in possible seki

6.1 Regions Surrounded by Safe Blocks

Once the safety of some blocks has been established, it becomes easier to prove the safety of further adjacent regions since these regions need not provide sure liberties for these blocks. It is sufficient to prove that the attacker cannot live inside the region. In Figure 17, the safety of the black blocks has already been proven by using regions *a* and *b*. If White cannot make two eyes in the top left, it proves that the area is black territory. Both static and search-based methods are considered for proving the safety of regions surrounded by safe blocks.

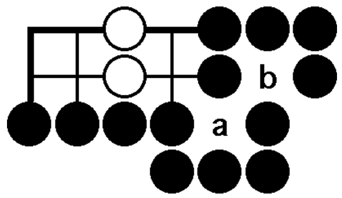

Figure 17: Proving the safety of regions

Static Recognition of Safe Regions

The attacker *cannot live inside* a region surrounded by safe blocks if there are no two nonadjacent potential attacker eye points, or if the attacker eye area forms a *nakade* shape.

6.2 Areas Adjacent to Some Safe Stones

Consider the case where the safety of some but not all enclosing blocks has been established already. These blocks have two safe external liberties. Some of the other stones may have one safe external liberty, or external connection possibilities. Such cases can be handled by a combination of the search algorithms discussed above.

7 Results

The performance of both static and search-based algorithms has been tested on the problem set *IGS_31_counted* from the Computer Go Test Collection [4]. This set contains 31 games played by amateur dan players. The games have been played to completion and counted.

7.1 Test on Final Positions from Games

This test was performed on the final positions of the 31 games. The goal was to prove the safety of stones and territories in each position. An ideal algorithm should classify 100% of the points in this test set. The table shows the number of points and blocks that could be proven safe using Benson's, the static and the search-based algorithms. It also shows the number of points that were proven to be *dame*, using the algorithm from [3], p. 64.

Test Results

The test set contains $31 \times 361 = 11191$ points and 1123 blocks. Our preliminary implementation used a subset of the methods described in section 4. The depth limit for iterative deepening was set to 6 ply. Three different kinds of searches were performed:

- recognize 1 sure liberty
- recognize 2 sure liberties
- prove the safety of regions surrounded by safe stones

Method	Points	Blocks	Dame
Benson	1886 (16%)	103 (9%)	16
Static	2481 (22%)	168 (14%)	22
Search	2954 (26%)	198 (17%)	24

Figure 18 shows the best and the worst results. The shaded regions have been proven to be safe territory. In the best case (92% of points classified), there are many small regions, and almost all of them can be resolved. In the worst case (0%), the board contains only a few huge territories. Current algorithms have no chance of proving them safe. This situation occurred in 5 of the 31 test positions.

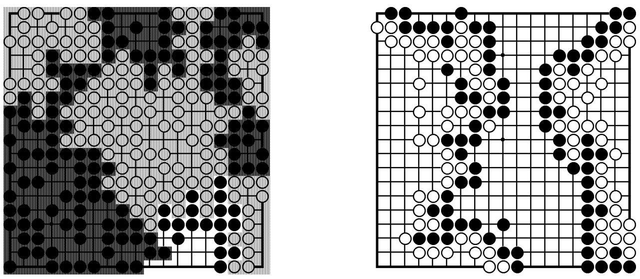

Figure 18: Examples of good (Nr.4) and bad (Nr.9) classification

Heuristic Classification

In an informal experiment, the evaluation of the Go program Explorer [3] was tried on the same test set. Almost all blocks and territories were classified correctly by the heuristic evaluation. In a few cases, Explorer thought that a dead group was still unsettled, or that a huge territory was not yet safe. For computer-computer games, these assessments may be appropriate.

8 Applications of Safety Proving Techniques

Safety proving techniques have many important applications in Computer Go, and will probably become standard ingredients of programs as they approach dan level. Some cases where safety proofs are useful are:

exact counting of the score To count the score it is neccessary to determine which territories are already safe and which defensive moves have to be played.

finding threats Ko threats can be found by trying out moves against safe territories and re-checking whether they are still safe.

board partition For the exact solution of endgames by combinatorial game methods [3], board partition by provably safe stones is a required first step.

seki To create or prevent seki as in Figure 2, safety of territory must be investigated even if the tactical safety of stones has been proven.

9 Summary and Future Work

Increasingly powerful computers make the application of exact methods in Computer Go feasible for a growing number of subproblems. In this paper, an exact method for proving the safety of stones and territories was presented.

The algorithms work well if the board decomposes into a large number of regions. To handle larger areas, the algorithm can be improved in a number of ways: more elaborate partition plans for big areas, heuristic evaluation for move ordering, and better static tests for sure liberties. The algorithms could also be generalized to work in non-closed regions, make use of outside liberties, and incorporate knowledge on *semeai*.

References

[1] David B. Benson. Life in the Game of Go. Information Sciences, vol. 10 (1976), pp.17-29. Reprinted in Computer Games, Levy, D.N.L. (Editor), Vol. II, pp. 203-213, Springer Verlag, New York 1988.

[2] Toshio Ikeda. On the Rules of Go. Fujitsu Ltd., Tokyo, Japan 1992.

[3] Martin Müller. Computer Go as a Sum of Local Games: An Application of Combinatorial Game Theory. Ph.D. thesis, ETH Zürich, 1995.

[4] Martin Müller. CGTC, the Computer Go Test Collection. http://wwwjn.inf.ethz.ch/martin/cgtc.html. ETH Zürich, 1995.