Race to Capture: Analyzing Semeai in Go

Martin Müller
ETL Complex Games Lab
Tsukuba, Japan
mueller@etl.go.jp

September 10, 1999

Abstract

Skill in analyzing capturing races, or *semeai*, is an important component of Go playing strength for both humans and computer programs. Techniques for analyzing semeai have been developed centuries ago, and passed on among Go players. A number of informal descriptions of the basic methods are available in Go books. This paper starts developing a formal theory of semeai, leading to an algorithm for basic semeai classes that has been tested successfully against top Go programs.

Figure 1: Class 0 semeai: one essential block each, plain outside liberties, shared liberties

1 What is a Semeai?

Figure 1 shows two simple cases of a *semeai*. A textbook-like definition is "A race to capture between two adjacent groups that cannot both live". Such a definition may be good enough for humans, but it is not sufficiently precise for implementing semeai in a computer program. This paper gives both more narrow and more general definitions of semeai. The narrower definitions cover cases which can be detected and evaluated statically, without search. The more general definitions are designed to cover potential semeai, unclear situations which might end up as a race to capture, and can be resolved by search. However, during the search it might turn out that the situation is not really a semeai, because one side can make life, leaving the other side dead, or because both sides should defend themselves instead of attacking the other.

While this paper deals mainly with static analysis of semeai, it thereby also provides a foundation for efficient search-based algorithms.

Section 1 clarifies the term semeai, and its relation to other tasks such as proving the safety of territories and solving Life-and-Death (tsumego) problems. Section 2 describes the components of semeai such as essential blocks, liberties, and eyes, gives the outline of a general algorithm to solve semeai, and classifies semeai into nine categories. Section 3 deals with the static analysis of the two simplest semeai classes, describes how to recognize such semeai, and gives a complete solution in terms of combinatorial game values. The section also describes restrictions on which semeai can be statically recognized, and motivates why these restrictions are necessary by examples. Section 4 very briefly deals with semeai classes 2 and higher, and Section 5 shows some results on full board semeai problems.

1.1 Semeai and other Game Phases

Semeai often occur as subproblems of another task, such as proving the safety of stones and territory, or solving life and death problems. Examples include: keep a territory safe by winning a semeai against an intruder, rescue stones by cutting through a surrounding wall and winning the semeai against some part of the wall, connecting stones by setting up a

lack of liberties for the opponent: if he or she tried to cut, that would lead to a lost semeai.

Cases where blocks of both players have few liberties can be handled reasonably well with existing capture search techniques. The methods presented here are most useful in situations where blocks have many liberties and/or large eyes. In such cases they are much more efficient than a purely tactical search.

2 Describing Semeai Instances

2.1 Finding Semeai

Perhaps surprisingly, identification of semeai in a given Go position requires the same preliminary analysis as for the endgame [3, 5]: identification of safe blocks and territories [4], followed by a partition of the rest of the board into connected components called *local games*. Each local game, consisting of empty points, plus possibly unsafe stones of either player, can potentially become a semeai. This holds even if the area is currently completely empty or contains stones of only one player.

To be able to evaluate semeai statically, much more restrictive conditions must be satisfied. Such conditions will be discussed in section 3.

2.2 Classification of Blocks and Empty Points

Classification of points in a local game is a first step in identifying semeai. In each local game, we recognize the following types of blocks of stones and empty points:

Essential block A block of black or white stones which must be saved from capture. Capturing an essential block immediately decides a semeai.

Nonessential block Block which can be captured without losing the semeai. An example of a nonessential block is a small block contained in the opponent's eye, such as the single white stone in Figure 3.

Unknown block Contains all remaining blocks that cannot be classified as either essential or nonessential blocks. Saving or capturing such blocks has some priority as a heuristic, but it does not necessarily decide the semeai.

Outside liberty An empty point that is a liberty of an essential block of one player, but not a liberty of an essential opponent block. An outside liberty is called *plain* if it is also adjacent to a safe opponent block, so the opponent can fill the liberty without making additional approach moves.

Shared liberty Common liberty of essential blocks of both Black and White.

Eye An area completely surrounded by essential blocks of one player. The area can contain nonessential blocks of either player. A *plain eye* has only one surrounding block, and all empty points inside are adjacent to that block. This definition is broader than the usual one, and includes cases where the surrounded region will end up as more or less than one eye.

Unknown area Area that cannot be classified as outside liberties, shared liberties, or eye.

2.3 Eye Status and Liberty Count

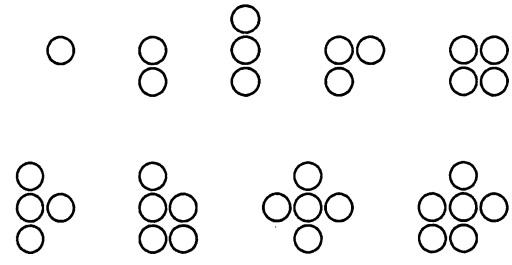


Figure 2: Basic nakade shapes

An eye area is called a *nakade* if the opponent can fill all but one of its points by one of the basic nakade shapes shown in Figure 2.

In semeai, small eyes with size from 1 to 3 behave in the same way, while larger eyes are stronger both in terms of providing more liberties than their size and in having an advantage in semeai against smaller eyes. We model this behavior by an *eye status*. For each eye size, Table 1 shows the status and the number of liberties. For $0 \leq m < n \leq 7$, a

Size	0	1	2	3	4	5	6	7
Status	0	1	1	1	4	5	6	7
Liberties	0	1	2	3	5	8	12	17

Table 1: Eye status and liberties

n point nakade shape filled by m opponent stones is equivalent to $(n^2 - 3n)/2 + 3 - m$ outside liberties. A nakade shape is *unsettled* if it has not yet been reduced to only one eye, and the defender can still make two eyes there. Figure 4 shows an example.

2.4 Steps of a Semeai Algorithm

The following steps are a general outline of a semeai solving algorithm. Some details are discussed in later sections.

1 Board Partition Find safe blocks of stones and territories. Partition the rest of the board into connected components, called *local games.*

2 Semeai Identification Investigate which local games are semeai candidates by the following substeps.

2.1 Eye Recognition Subdivide each local game into regions surrounded by stones of a *single* player. Test each such region whether it is a plain eye for that player.

2.2 Liberty Regions After finding blocks and eye regions, divide the rest of a local game into liberty regions surrounded by stones of *both* players. Classify liberty regions as outside liberties, shared liberties, or unknown.

2.3 Block Classification Classify blocks as essential blocks, nonessential blocks, and unknown blocks.

2.4 Semeai Safety Test For each color, determine if winning the semeai would ensure the safety of all essential blocks.

2.5 Semeai Classification Determine which semeai class the local game belongs to.

3 Static Evaluation For semeai of classes 0 to 2, statically evaluate the semeai to find its status and its combinatorial game evaluation.

4 Search For semeai of classes 3 or higher, use search to find the outcome.

5 Move Generation for Semeai Play Using the results of search or static evaluation, generate moves to play each semeai on the board. Use exact combinatorial game values when available, and a heuristic temperature estimate otherwise.

2.5 A Classification of Semeai

The following classification proceeds from simple semeai that can be analyzed statically to cases with less structure, which require more and more search to solve.

Class 0 Exactly one *essential block* of each color, only plain external and plain shared liberties, no eyes.

Class 1 One essential block of each color, may have one *plain eye* potentially containing one opponent nonessential block in a *nakade* shape.

Class 2 Like class 1, but eyes include *unsettled nakade.*

Class 3 Non-plain eyes and/or external liberties which can be proven by search to be equivalent to some plain eye/liberty region.

Class 4 General eyes and/or liberty regions. More complicated values of regions, for example regions that allow to gain or lose more than 1 liberty by a move. Regions with unsettled eye status: players can make or prevent eye(s) there.

Class 5 Additional unsafe blocks in liberty regions. Connect, cut these blocks to gain/reduce liberties. However, these blocks are not adjacent to opponent's essential block.

Class 6 More than one essential block per player, but they form a chain. Cutting the chain wins the semeai for the opponent.

Class 7 General semeai in completely surrounded local game.

Class 8 Local area not completely surrounded by safe stones.

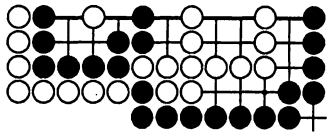


Figure 3: Class 1 semeai: plain eyes

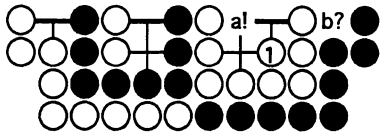
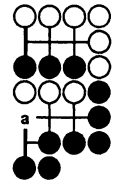


Figure 4: Class 2 semeai: unsettled nakade



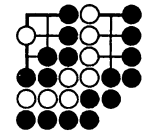
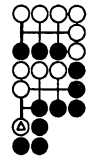


Figure 5: Class 3 and 4 semeai: non-plain liberty at 'a', non-plain eye space



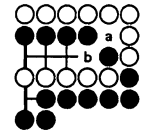


Figure 6: Class 5 and 6 semeai: unknown block, more than one essential block

Figures 1 to 7 contain examples of each semeai class. Figure 1 shows two class 0 semeai. On the left, Black has one plain liberty region containing three plain liberties, and White has two plain liberty regions containing one liberty each. On the right, White has one plain liberty, and there is one shared liberty region containing 3 liberties.

In Figure 3 both Black and White have an eye. Black's eye includes a white stone. This stone is nonessential: Black should not try to capture it, and White can afford to lose it without losing the semeai. During the course of the semeai the eye area will be occupied by nonessential stones and emptied by a capture several times.

Figure 4 shows an unsettled nakade shape. Black must play in at 'a' to prevent White from making two eyes. In Figure 5 on the left, the white liberty region is not plain because the point 'a' is not adjacent to a surrounding opponent block. However, in semeai the area behaves like a plain three liberty area, a fact which can be proven by search. The right side picture shows an eye area that is not plain, since the corner point is not adjacent to the black block. In this case the number of liberties is dramatically reduced from 7 to 3.

Figure 6 shows a class 5 semeai on the left, which contains an additional white block. White can win the semeai among the essential blocks going first, but cannot save the marked block. In the class 6 semeai on the right side, Black has two essential blocks which form a chain and can connect at 'a' or 'b'. Black cannot give up either of the essential blocks, in contrast to the situation in the previous figure, where White could afford to give up one block yet win the main fight. Figure 7 shows a general semeai of class 7 on the left, which lacks any of the special properties of classes 0-6. The semeai area is still completely surrounded by safe blocks. Finally, the right side shows a class 8 semeai which does not have a complete surrounding wall. In such a situation players must consider additional options such as trying to break out to the outside, or counterattacking against the wall. Fixing the boundary of the semeai becomes somewhat arbitrary.

3 Static Semeai Analysis

3.1 Semeai Evaluation

The traditional liberty-counting methods for semeai evaluation are known to many Go players. [1] provides a detailed introduction, while [2] contains a more concise summary. In our classification, the evaluation directly applies to semeai of class 0 and 1.

3.1.1 Who Wins, and by How Much?

To decide who wins a semeai, compute each player's liberty count and eye status, as well as both players' shared liberties.

A player's liberty count, L_B or L_W, is the sum of the number of outside liberties, plus the liberty

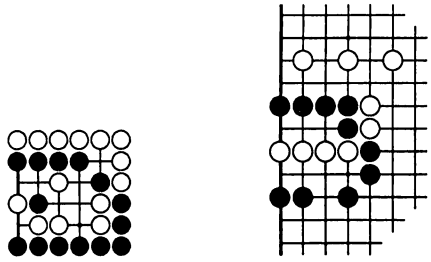


Figure 7: Class 7 and class 8 semeai: completely surrounded and open-ended area

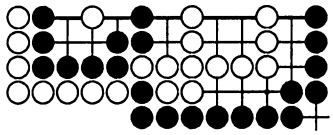


Figure 8: Counting liberties

count of the players eye (if there is one). For example, in Figure 8 the black block has 0 outside liberties plus $8 - 1 = 7$ liberties from a five point eye containing one opponent stone, so $L_B = 0 + 7 = 7$. The white block has 6 outside liberties plus 5 liberties from a four point eye containing 0 opponent stones, for a total of $L_W = 6 + 5 = 11$ own liberties. The number of shared liberties $S = 2$.

If the eye status of players is different, then the player with poorer eye status must be the attacker. If both have equal eye status, and one player has a surplus of own liberties, that player is the attacker. If both eye status and liberty count are equal then either player can be the attacker. Set $\Delta = L_B - L_W$ if Black is the attacker and $\Delta = L_W - L_B$ otherwise. In the example, White is the attacker because of poorer eye status (4 vs. 5), so $\Delta = L_W - L_B = 4$.

The number of *forced liberties* F is the number of shared liberties that attacker has to fill to put defender into atari. If defender has an eye, or there are no shared liberties ($S = 0$), then $F = S$. If defender has no eye and $S > 0$, $F = S - 1$. In the example Black has an eye, so $F = S = 2$.

The semeai formula compares the attacker's liberty surplus to the number of forced liberties:

The semeai formula: Attacker succeeds if

$$\Delta \geq F.$$

Furthermore, attacker needs to play only if $\Delta = F$. If $\Delta > F$, attacker is $\Delta - F$ moves ahead in the semeai and need not play immediately. If attacker fails, there are two cases: if the eye status of players is different, or the number of shared liberties is too small, then attacker loses. Otherwise it is a seki, in which the stronger side must play $F - \Delta$ moves before it becomes unsettled. In the example, $\Delta > F$, so White wins and is $\Delta - F = 2$ moves ahead in the semeai.

The outcome of the semeai can be summed up by two numbers: *Semeai status* measures how many moves the winner is ahead in the fight. Positive values are good for Black, negative values are good for White. If the semeai status is 0, the outcome is either unsettled, or seki. *SekiLevel* is defined only if semeai status is 0, and measures how many consecutive moves the stronger side can make before the outcome changes from seki to unsettled. If *sekiLevel* $= 0$, the outcome is unsettled, if *sekiLevel* > 0 it is a seki. The following code fragment computes semeai status and sekiLevel, determines whether the semeai is unsettled and computes the winner.

```
// input: betterEye, Δ, ΔBW = LB - LW, F
// output: semeaiStatus, sekiLevel, isUnsettled, winner
  if (betterEye == none) // same eye status: may be seki
  { if (Δ ≤ F)
    { semeaiStatus = 0; sekiLevel = F - Δ;
    }
    else
      semeaiStatus = ΔBW + ((ΔBW > 0)? -F : F);
  }
  else // different eye status: never seki
  { semeaiStatus = ΔBW;
    semeaiStatus += (betterEye == Black) ? F : -F;
  }
  isUnsettled = (sekiLevel == 0) && semeaiStatus == 0;
  winner = (semeaiStatus == 0) ? none :
    (semeaiStatus > 0) ? Black : White;
```

3.2 Semeai and the Safety of Stones and Territories: When Winning the Semeai Loses the Fight

3.2.1 Capture-Recapture Tactics

In some cases, being the first to capture the opponent does not prevent the player's stones from being captured later. Examples of *snapback*, *oi-otoshi* and *ishinoshita* are shown in Figures 9 and 10. In each of these examples, semeai analysis shows that Black can capture White going first. However, this capture fails to secure the black stones, and they

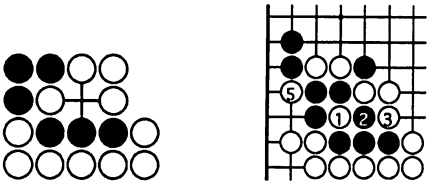


Figure 9: Snapback and oi-otoshi

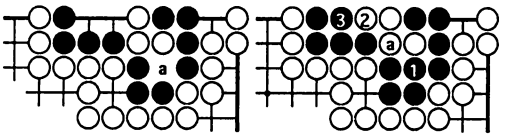


Figure 10: Ishinoshita

will be captured when White again plays into the empty region of the original white stones.

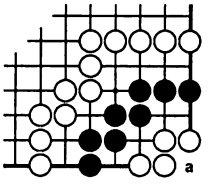


Figure 11: Hidden nakade shape

3.2.2 Capture Does not Create Two Eyes

For determining whether a capture will guarantee two eyes for a player, it is not enough to analyze the current shape of the opponent's stones. In Figure 11, the current shape of the white stones in the corner is not a nakade, so Black might have some hope of living. However, White can create nakade at 'a' any time, and Black cannot prevent it, so Black is dead with only one eye. Figure 12 shows another example: Black connects at 1, expecting White to make two eyes in the corner at 'b' so Black can connect at 'a'. However, White can play at 'a', and it is useless for Black to 'kill' the corner. If Black captures two stones, White recaptures and connects, leaving a 6 point nakade shape. Black has only one eye.

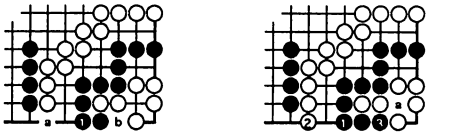


Figure 12: Another hidden nakade shape

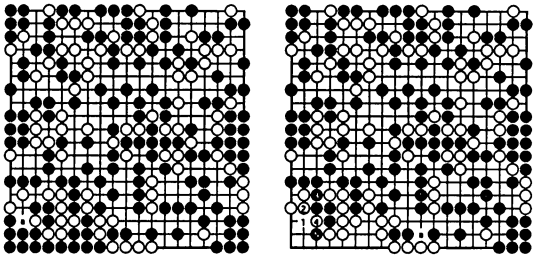


Figure 13: 88 stone capture problem

Figure 13 shows a classic problem composed by Intetsu Akaboshi in the 18th century, in which White captures a total of 88 black stones but still dies. The problem starts off with a 16 stone capture which leads to only one eye.

A related question is whether a nakade-like situation guarantees at least seki by two shared liberties for the player on the outside. In Figure 14, blocks 'A' are alive in seki, but blocks 'B' are dead. A general classification procedure for eye shapes is beyond the scope of this paper.

3.2.3 Two Simple Rules for Ensuring Safety

One of two simple static rules can often be applied to ensure that winning the semeai will lead to safety. Rule 1 ensures safety by connection, Rule 2 by making a second eye for a group that already has one eye. As the examples of Figures 11, 12 and 13 demonstrate, it may not be easy to come up with good static rules for the third case: ensuring two eyes by capturing.

Rule 1: A player's essential block in semeai will be safe after winning the semeai if there exist two opponent stones which are adjacent to both the

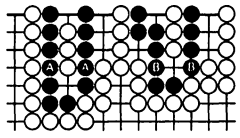


Figure 14: Seki(A) and seki-like nakade shape(B)

block and a safe block of the same player.

Rule 2: A player's essential block in semeai will be safe after winning the semeai if it has one eye and there exists another region containing an adjacent opponent block which is *1-vital* [4] for the player.

3.3 Determining the Game Value

We will use Japanese rules for counting. Chinese rules differs mainly in case of seki. The game value for semeai of classes 0 and 1 is simple: if the winner is decided, or if it's a seki, the score is an integer. If the semeai is unsettled, the score is a switch. The game value can be computed using two auxiliary functions returning an integer: *WinnerScore(player)* and *SekiScore()*.

WinnerScore(player) is determined as follows: every shared liberty and every opponent external liberty counts as one point. Every opponent stone counts as two points. From the sum of these values, subtract the number of necessary defensive moves. Finally, the sign of the total score is +1 for Black and -1 for White.

$$|WinnerScore(player)| = shared + ext(opp)$$
$$+2 \times stones(opp) - DefensiveMoves(player)$$

The number of necessary defensive moves, *DefensiveMoves(player)*, is a nonnegative value, the minimum of two values: the number of moves required to capture the opponent, and the number of moves the player has to make to remain ahead in the semeai if the opponent fills all outside liberties.

$$DefensiveMoves(player) = max(0,$$
$$min(ext(player) + 1 - |semeaiStatus|,$$
$$libs(opp) + shared))$$

SekiScore() is close to 0, but takes into account the number of stones in an eye that can be captured. An eye of size 2 is worth one point, an eye of size 3 three points. For example, a seki between a black 2 point eye and a white 3 point eye has score $1 - 3 = -2$.

The game value can now be computed as follows:

```
ComputeGameValue()
{
  if (isUnsettled)
  { int bValue = CanWin(Black) ?
       WinnerScore(Black) : SekiScore();
    int wValue = CanWin(White) ?
       WinnerScore(White) : SekiScore();
    game = Switch(bValue, wValue);
  }
  else if (winner == none)
    game = IntGame(SekiScore());
  else
    game = IntGame(WinnerScore(winner));
} // ComputeGameValue
```

To get the exact combinatorial games, all scores must be adjusted by * if an odd number of dame points remains after resolving the semeai.

One special case must be handled differently: a block with a big eye might be in atari. In this case, the player should often move immediately even if the semeai status is not unsettled. However, if the player is losing the fight anyway, the incentive for moving might be only a small *semedori* value, the difference between the opponent winning in one move and winning by filling a number of liberties. The game value also changes accordingly. We have implemented the necessary modifications in our program, but lack space to show the details here.

4 Solving Other Semeai

We have implemented static evaluation for class 2 semeai, and a search method for classes 3 and higher. Again, because of a lack of space we must postpone a detailed description of these methods to a future technical report. A few examples of semeai search and some preliminary results are contained in a previous paper [6].

4.1 Class 2 Semeai: Unsettled Eye

Class 1 semeai are not 'closed' under play: during play, if nakade stones are captured, an unsettled eye shape such as the one in Figure 4 can result,

so the semeai classification changes to class 2 for one move. A single player's unsettled eye shape can be handled similarly to the case of a group in atari mentioned above. Further rare special cases can occur: One player can be doubly unsettled: the player's group is in atari, and the opponent can live in an unsettled eye shape. In this case the player has lost without further moves (unless the opponent is also in atari). If both players are unsettled because of an unsettled eye and/or atari, players must decide between attack and defense, depending on the scores and outcomes that would result.

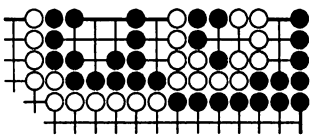


Figure 15: Double sente play in class 2 semeai

In Figure 15 Black can make seki, while White can live with 9 points against Black's 5. The game value is $38|*||-4|-45*$, a 4 point double sente play in classical endgame terminology.

5 Test Examples: Full Board Semeai Problems

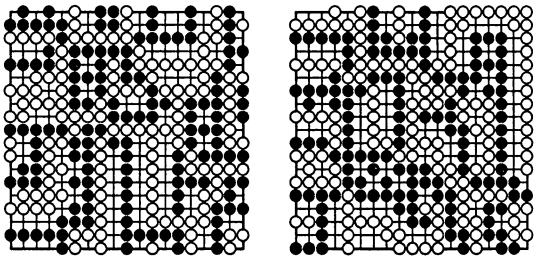


Figure 16: Full board semeai problems 1 and 2

The semeai evaluation method described here has been integrated into our Go program *Explorer* [3]. For testing, we constructed two full board Go positions that contain a large variety of semeai positions of classes 0-2, as shown in Figure 16. Starting from these positions, we played Explorer against two of the recent world champion programs, *Many Faces* and *Go 4++*. Explorer won most test games by large margins, often gaining more than 100 points over the game-theoretic outcome. Detailed results are given in [6]. The test positions and game records in SGF format are also accessible through the internet at http://www.etl.go.jp/etl/suiron/~mueller/cgo/semeai.html. From this experiment, it is clear that our method is able to evaluate semeai much earlier and much more precisely than the other tested programs.

References

[1] R. Hunter. Counting liberties, How to win capturing races. In R. Bozulich, editor, *The Second Book of Go*, chapter 7,8. Kiseido, Tokyo, 1998.

[2] K.F. Lenz. Die Semeai-Formel. *Deutsche Go-Zeitung*, 57(4), 1982.

[3] M. Müller. *Computer Go as a Sum of Local Games: An Application of Combinatorial Game Theory.* PhD thesis, ETH Zürich, 1995. Diss.Nr.11.006.

[4] M. Müller. Playing it safe: Recognizing secure territories in Computer Go by using static rules and search. In H. Matsubara, editor, *Proceedings of the Game Programming Workshop in Japan '97*, pages 80–86, Computer Shogi Association, Tokyo, Japan, 1997.

[5] M. Müller. Decomposition search: A combinatorial games approach to game tree search, with applications to solving go endgames. In *IJCAI-99*, volume 1, pages 578–583, 1999.

[6] M. Müller. Partial order bounding: Using partial order evaluation in game tree search. Technical Report TR-99-12, Electrotechnical Laboratory, Tsukuba, Japan, 1999.