A library of eyes in Go
I: A life & death definition consistent with 'bent-4'

Thomas Wolf
Department of Mathematics, Brock University
500 Glenridge Avenue, St.Catharines,
Ontario, Canada L2S 3A1
email: twolf@brocku.ca

February 14, 2009

Abstract

In the game of Go we develop a consistent procedural definition of the status of life and death problems. This computationally efficient procedure determines the number of external ko threats that are necessary and sufficient to win, and in the case of positions of the type of bent-4-in-the-corner it finds that they are unconditionally dead in agreement with common practice. A rigorous definition of the status of life and death problems became necessary for building a library of monolithic eyes (i.e. eyes surrounded by only one chain). It is also needed for comparisons of life and death programs when solving automatically thousands of problems to analyse whether different results obtained by different programs are due to different status definitions or due to bugs.

1 Introduction

1.1 Overview

In this contribution we describe a project whose aim was to built a data base of eyes together with their life and death status which at least reflects one aspect of ko accurately: the number of necessary external ko threats for the weaker side to win. The procedure how to determine this number is described in section 2. After that we seem to be ready for determining the status of a life and death problem if there would not be the bent-4-in-the-corner positions (in the following called *bent-4*) which are characterized in section 3 and force us in section 4 to refine the procedure that we take as the (procedural) definition of the status of a life and death problem. In the appendix we discuss current limitations of the program GoTools which is the implementation behind this article.

1.2 The key problem

The discussion in sections 2 - 4 is rather detailed and arguments are developed why procedures and rulings were designed as they are. In order not to lose sight and have an orientation when reading them we already now want to address the key problem.

In this contribution we consider the procedural definition of the status of life and death (l/d) problems, i.e. of positions which are isolated from the rest of the board through a single solid chain of stones that has enough external safe liberties to live statically. The procedure as outlined in section 2 is capable of classifying nearly all types of common and also strange l/d-positions correctly within some approximations (like treating life and seki alike as listed in the appendix) and up to mastering the computational complexity. The only exception encountered so far is a class of bent-4-type positions as defined in section 3 that includes, for example, the one in diagram 1. Positions of this class are characterized essentially a) by having a ko-status when evaluated according to the straight forward rules of section 2 and b) by the key property that at some stage of optimal play, the side for which this ko is unfavourable (White in diagram 1) has as single best move only the passing move.

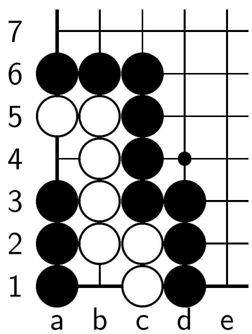


Diagram 1.

The combination of these 2 properties has severe consequences. If there are only removable ko's on the board (e.g. cuts, but no seki) then Black can wait until later in the game and protect all potential ko-threats (i.e. 'remove' potential ko threats) at no cost before starting the ko. In that case the status would be an unconditional loss for White which also is what Go players expect from a computer Go program to find[1]. A fundamental principle of the orthodox procedure in section 2 is that a position is alive or seki unless the attacker can prove how to kill it (unconditionally or through ko). But in bent-4-type problems the best move for the attacker is to pass, at least during the 'hot' phase of the game when playing elsewhere (*tenuki*) has some benefit. In other words, bent-4-type positions do not have a single solution for the best move, they have two: 1) if playing elsewhere is beneficial then the single best move for the attacker is to pass, 2) if playing elsewhere is not beneficial, and if the attacker is asked to prove that the position is not a seki then the best move for Black in diagram 1 is to play on b1. Situation 2) is mastered in a straight forward manner, as described in section 2. The challenge is situation 1). To satisfy for 1) these conflicting requests (i.e. to find that the position is not alive/seki *and* that passing is the best move for the attacker) in a consistent, local procedure, without having to check special cases separately, is the goal of section 4. Once both situations can be handled, all that is needed is an extra boolean input parameter specifying for which of the two situations the computation shall be valid.

1.3 Notation

Throughout this paper we will call the side that builds eyes and tries to live as White and the side trying to kill as Black. The side moving next in a position will be called 1^{st} and the other 2^{nd}. To have finite and effective searches the right to pass is strongly regulated. In this article we will derive, modify and collect *rulings* which include statements when passing is allowed. In these rulings we will use ● and ○ instead of 'Black' and 'White' to get a more compact formulation. We will follow common terminology and call a position which contains an empty point that is forbidden for one side due to the ko rule as a *ko-banned* position and all other positions *regular*. When the text refers to ko threats then these are always external ko threats, i.e. ko threats outside of the problem.

[1]At times when GoTools solving life and death problems online under [6] did not find bent-4 to be dead, many error reports were submitted by users.

2 The ko status of a life & death problem

The program GoTools (described in more detail in [1] and with restrictions listed in the appendix) is the implementation behind the theory in this article. It performs an α/β search with only two possible outcomes: life/seki or death, which is therefore called 'boolean search' below. If the status is ko then it repeats the search to find the number of external ko threats needed by the weaker side to win.

2.1 Re-runs with successively more ko-threats

In a first search no side is allowed to re-capture a ko. If for one side, say 1^{st}, all moves fail in a position in depth d then the result of this computation is not only the loss but attached is always a boolean variable *ko-chance* which, if true, means that the outcome could have been different if 1^{st} would have had a ko-threat. If that is the case, the previous move by 2^{nd} at depth $d-1$ is a winning move but attached to it is ko-chance=true. This is how this information moves upwards in the search tree.

In any position at depth d ko-chance is set to true

- if the position is ko-banned and re-capturing the ko would not have violated any cycle rules (see below), but was not possible due to a lack of ko threats, or
- if *any one* of the winning moves of 2^{nd} at depth $d+1$ returned ko-chance = true.

For example, 1^{st} has 5 possible moves in some position in depth d. The first tried move fails and has ko-chance=false. The second move of 1^{st} fails too but the counter-proving move of 2^{nd} in depth $d+1$ returns ko-chance=true. The third move of 1^{st} happens to win, thus search in this level stops. Thus the ko-chance of 1^{st} from the second try becomes irrelevant. What is relevant is whether 2^{nd} has a ko-chance from any one of its losing tries at depth $d+1$. If so, then not only will be reported to level $d-1$ which move of 1^{st} won but also that 2^{nd} has a ko-chance for this move.

At the end of the first computation the program knows whether in the verification tree of the search (the minimal tree to be searched where the final winner plays only winning moves and the loser all possible moves) the loser still has a *ko-chance*, i.e. a chance to win if it had one more external ko threat initially. In such a case a second run, and if necessary more runs, are performed with successively more external ko threats initially allocated to the loser. This is continued until either the loser wins or loses without having a ko-chance in the last run or until a maximum of k_m external ko threats are reached after which the status is regarded as an unconditional loss for the side that lost so far. In our calculations $k_m = 5$, which could easily be changed to an arbitrary high but fixed value.

2.2 The different ko-status

We now come to the different possible status of a position. If the maximal number of allowed ko threats is k_m then $2k_m + 2$ different status may result, each characterized by a numerical value. The possible outcomes sorted from most beneficial for 1^{st} to least beneficial for 1^{st} are:

Value		Status
$k_m + 1$	:	an unconditional win for 1^{st},
k_m	:	a win for 1^{st} unless 2^{nd} has k_m external ko threats more than 1^{st},

$\cdots$
1 : a win for 1^{st} unless 2^{nd} has 1 external ko threat more than 1^{st},
-1 : a loss for 1^{st} unless 1^{st} has 1 external ko threat more than 2^{nd},
$\cdots$
$-k_m$: a loss for 1^{st} unless 1^{st} has k_m external ko threats more than 2^{nd},
$-k_m - 1$: an unconditional loss for 1^{st}.

For example, in diagram 2 the status is $k_m + 1$ for any side moving first, as both can win unconditionally by playing on b1. In diagram 3 both sides would pass as the position is unconditionally dead, i.e. the status is $k_m + 1$ if Black moves first and $-k_m - 1$ if White moves first.

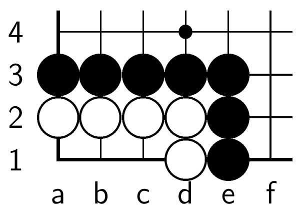


Diagram 2.

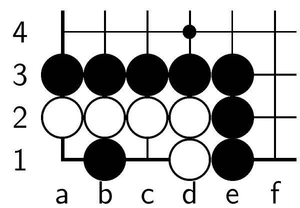


Diagram 3.

Status values are chosen so that if the outcome is the same for both sides moving first, then the status values just differ by a sign. For example, if the outcome is that White needs one ko threat in order to win, no matter who moves first, then the status value for White moving first is -1 and the status value for Black moving first is +1. In other words, if the outcome is independent of who moves first then the sum of the 2 status values is zero. But that is exactly the case when passing does not do any harm, i.e. passing is one of the best moves of both sides as in diagram 3. Conversely, if the sum of the status values is non-zero then it is beneficial for both sides to move first. In diagram 2 the sum has the maximum value $2k_m + 2$. In the context of this paper a move belongs to the best moves if no other move generates a higher numerical status value.

Lemma: *If for a regular position (such that both sides can be considered to move first) passing belongs to the best moves of one side then passing does also belong to the best moves of the other side.*

Proof (indirect): We assume that passing belongs to the best moves of, say, White. Then, if passing would not be one of the best moves of Black, then if Black would pass then White could make a move, such that the status for Black would be worse than if Black would not pass. In other words, if passing is not one of Black's best moves then passing is also not one of White's best moves which contradicts our assumption.

If passing belongs to the best moves of both sides then we call the position *settled* if it is unconditionally dead or alive/seki, otherwise it has a ko status and we call it *calm*.

A clarification:

To make clear that the terms 'best move' and 'calm' do depend on the type of computation performed, we should use *boolean-best move* and *boolean-calm* if they are determined in a boolean search but to keep the text better readable, we will continue to use simply 'best move(s)' and 'calm' although we exclusively refer to a boolean search. It is necessary to make this remark, because not all boolean-best moves are truly best moves[2] and thus not all boolean-calm positions are truly calm positions as seen in a collection of boolean-calm positions in [5].

[2]A move of White giving seki is boolean-best but not truly best if there is another move reaching life.

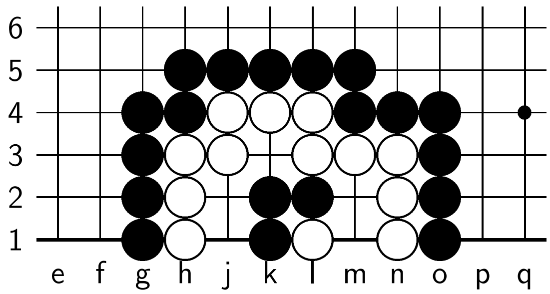


Diagram 4.

Diagram 4 is an example for a (boolean-) calm position: Here Black needs one exterior ko threat to kill, regardless of who moves first (White passing or playing on m2, Black passing or playing anywhere apart from m2 which would be followed by White on m1: seki). Therefore, the status for White moving first is 1 and for Black moving first −1, giving a zero sum.

Lemma: *For a regular position the sum of status values for both sides moving first is never negative.*

Proof (indirect): We assume, the sum of both status values is < 0. Then at least one of both status values must be negative. Let X be the side with the most negative of both status values, i.e. with value $s_X < 0$. Even if the opponent of X would have no better first move than to pass, then the achieved status value would still be $-s_X > 0$ from passing, giving a sum of at least zero in contradiction to the assumption.

The above lemma is typical for games that allow passing and have no ‘zugzwang’.

3 Characterization of bent-4-type positions

The collection of plausible rules from section 2 and the appendix describes a finite procedure with a definite result. This result is essentially identical to what one expects from real Go apart from limitations in the appendix and apart from the 4 bent-4 positions

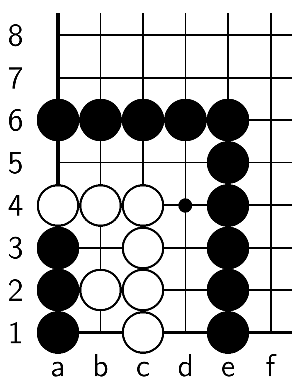


Diagram 5.

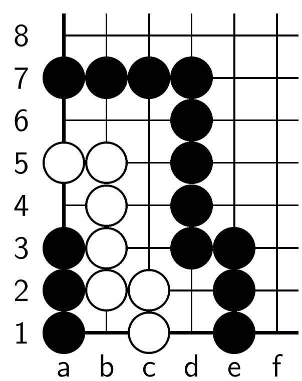


Diagram 6.

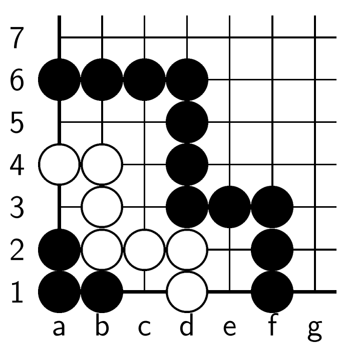


Diagram 7.

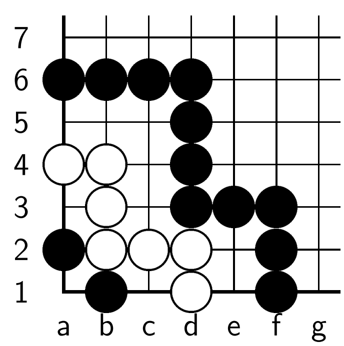


Diagram 8.

including those created by filling of external liberties of White, rotation, reflection and swapping colours.

When computed in accordance with the procedure definition given in the above sections, for example in diagram 5, the first 7 moves of Black would fill White's liberties , each followed by a pass of White and further ● b1 (giving 4 bent black stones in the corner, hence the name 'bent 4 in the corner'), ○ b3, ● a2, ○ a1, ● b1 resulting in a ko where White needs an external ko threat to live, independent of who moves first.

An essential difference between bent-4 and the position in diagram 4 is that in bent-4 passing is the *only* best (internal) move for White. Hence Black has the option to wait long enough until later in the game and then protect all potential white ko threats[3], fill outside liberties of White, produce an L-shaped throw-in chain which is caught by White and then play on a2 and start a ko in the corner, which Black captures first, i.e. White needs an external ko threat. As Black had enough time to remove at least all removable ko-threats, the position on its own is commonly regarded as dead, although its value in a real game, depends on the situation on the board and the rules used.[4]

When evaluated according to section 2 these four positions have the following more general properties.

Definition 1

Any position that has the following three properties is said to be of 'bent-4-type':

1. The initial position does not have a ko-forbidden point.
2. The status is a ko in which side X needs at least one external ko threat (more than the other side) to win.
3. Passing is the single best first move for X (apart from playing on dame points).

Comments

- From 3. it follows that passing is also one of the best moves of the opponent of X.
- In bent-4 (diagrams 5-8) we have X = White and dame points would be external liberties of White, like a5 in diagram 5.
- There are many positions which satisfy criteria 2 and 3 for X = Black (the attacker) but not 1-3, i.e. they have a ko-forbidden point. In a search of all positions that involve a single white eye with up to 11 internal points no position showed up satisfying criteria 1-3 for X = Black (i.e. X = attacker) and only diagrams 5-8 satisfy criteria 1-3 for X = White.
- Larger bent-4-type positions are possible ([4]). In Diagram 9 the best move for Black is to pass whereas White can always provoke a favourable ko by playing ○ e17, ● f17, ○ e19, ● f18, ○ c19 (i.e. here X = Black).

  In Diagram 10 the best move for White is to pass whereas Black can always provoke a favourable ko by playing ● h18, ○ g19, ● c19, ○ h19 (i.e. here X = White).

[3] This may not be possible, for example, in the case of an infinite source of ko threats, like a double-ko-seki somewhere else on the board, or a seki which for White is *less* costly to lose than bent-4 and for Black *more* costly to lose than bent-4.

[4] According to the Japanese 1989 rules, bent-4 positions are even unconditionally dead. More precisely, in the confirmation phase of the game the so-called 'pass-for-ko rule' implies an unconditional loss for White (see [2]). In the Japanese 2003 rules of Robert Jasiek, during the hypothetical-analysis (stage 2 of the game, following stage one, which is the alternating-sequence of moves) White would also have to ko-pass, leading to the capture of the white stones before White would be able to re-capture the ko (see [3]).

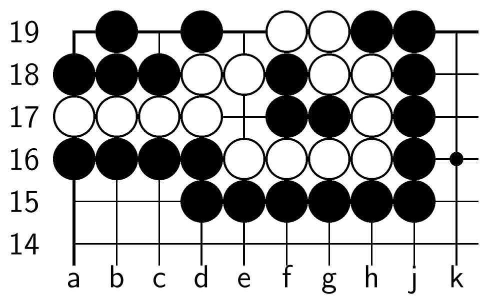


Diagram 9.

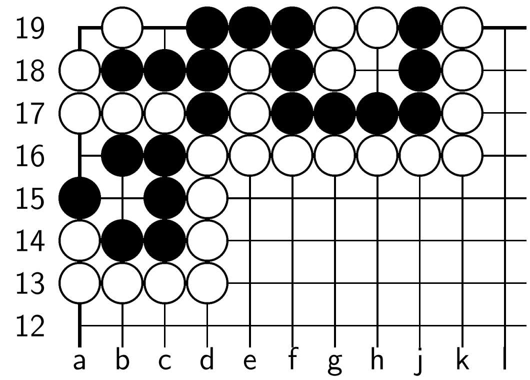


Diagram 10.

The question to be answered in the following section is:
Can one have a consistent boolean search which on one hand evaluates anything to be alive/seki which can not be killed by a non-passing move and on the other hand evaluates bent-4-type positions to be dead but the killing move is a pass?

4 Modification of the status defining procedure

4.1 The passing rules so far

To define the boolean search completely one must clarify under which circumstances passing is allowed. The standard passing rules in GoTools that do not yet take care of bent-4 are:

Ruling 1:

1. In a regular position,
   (a) if ○ moves next then
      i. if ○ re-captured a ko two moves earlier using one of its ko threats, and ● passed afterwards then ○ is not allowed to pass
      ii. else ○ is allowed to pass.
   (b) ● is not allowed to pass
2. In a ko-banned position,
   (a) if 1^{st} (the side moving next) has no external ko threat it may pass,
   (b) if 1^{st} has an external ko threat it may not pass.

If the position contains a point that is ko-forbidden then both sides may pass (rule 2). A refinement (rule 2b) is necessary in order not to lose unnecessarily despite of having ko threats and thus unnecessarily require re-runs with more and more external ko threats.

To treat *seki* as a situation in which White cannot be killed, it is necessary to allow White to pass also in regular positions (rule 1(a)ii). This rule needs an exception in form of 1(a)i. If White would be allowed to pass under the circumstances of 1(a)i then Black could re-capture the ko and a so-called *negative-value* 4 move cycle would be created in which White wasted an external ko-threat (see 'Handling cycles' in the appendix).

Black, on the other hand, should not be allowed to pass in a regular position (rule 1b) because afterwards White could pass too and White could not be killed. This rule creates a problem with bent-4 positions in diagrams 5 - 8 for which the best move for Black is to pass. The key to get bent-4-type positions right must therefore involve a change of the passing rules.

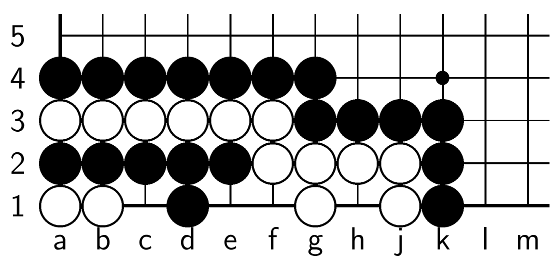
4.2 Passing for bent-4-type positions

One way to solve the problem with bent-4-type positions would be to perform an extra computation if in the first run the status turns out to be a favourable ko for Black and if there is initially no ko-forbidden point (i.e. if it is regular). In this additional search one would test whether Black can win if it passes as first move. This may work for diagrams 5 - 8 but a bent-4-type position could only result within one branch of a larger treesearch and the status of the larger problem may depend on the correct solution of the bent-4-type sub branch. We therefore need a 'local' rule (local in the sense of the whole search tree) about the right to pass and not a separate computation.

The key idea is to allow Black to pass if White has an external ko threat, i.e. to replace rule 1b in ruling 1 by rules

1(b)i: if ○ has no external ko threats then ● is not allowed to pass,
1(b)ii: if ○ has at least one external ko threat then ● is allowed to pass.

But after a type 1(b)ii passing of Black, White could pass too and would not be found to be dead. It seems to be necessary to forbid White to pass after a type 1(b)ii passing of Black, but this does not work either as can be seen from the following problem:



Black to move first
Diagram 11.

The solution sequence is ❶ f1, ② e1, ❸ c1, ④ a1, ❺ f1 such that White needs one external ko threat to live. Therefore, during the solution of this problem it comes to a second run in which White has one external ko threat. With one extra ko threat for White in the second run, the move ❶ f1 fails and all other alternatives have to be investigated, like ❶ pass. Therefore, in the second run eventually rule 1(b)ii is applied yielding ❶ pass, ② f1, ❸ pass. The resulting position is a seki, but to recognize it White must be allowed to pass with ④ (and it must be forbidden for Black to pass afterwards to have a finite algorithm). Like with bent-4 the crucial situation takes place in the second run when White has an external ko threat. The question is, what are natural rules which in a run where White has a ko threat,

- for bent-4 after ❶ pass *forbids* ② pass but
- for diagram 11 after ❶ pass, ② f1, ❸ pass, *allows* ④ pass (and forbids ❺ pass) ?

All Black's passes are of type 1(b)ii. The difference between both situations is: if there would be *no* extra White ko threat

- in bent-4 after ❶ pass, ② pass then Black still *wins*,
- in diagram 11 after ❶ pass, ② f1, ❸ pass, 4. ④ pass Black can *not win*

which in both cases is what we want. But that cannot be found out in a second run in which White *has* an extra external ko threat, *unless* one lets White pay the price of losing all external ko threats if White wants to pass *after* a Black type 1(b)ii pass.

If White cannot win after losing all ko threats, then this is a situation where passing is one of Black's best moves which improves a favourable ko to an unconditional kill as Black can wait until the end of the game before starting the ko. We therefore modify rule 1(a)ii and get

Ruling 2:

1. In a regular position,
   (a) if ○ moves next then
      i. if ○ re-captured a ko two moves earlier using one of its ko threats, and ● passed afterwards then ○ is not allowed to pass, else
      ii. if ○ has at least one external ko threat and if ● *has* done a type 1(b)ii pass in the sequence of moves up to now then ○ is allowed to pass but has to *give up* all external ko threats for any following moves, else
      iii. if ○ has no external ko threats or if ○ has at least one external ko threat and ● has *not* done a type 1(b)ii pass in the sequence of moves up to now then ○ is allowed to pass *without* giving up ko threats.
   (b) if ● moves next then
      i. if ○ has no external ko threats then ● is not allowed to pass,
      ii. if ○ has at least one external ko threat then ● is allowed to pass.
2. In a ko-banned position ,
   (a) if 1^{st} (the side moving next) has no external ko threat it may pass,
   (b) if 1^{st} has an external ko threat it may not pass (but should instead play the ko threat and then re-capture).

Comments

- The above rules about passing are part of a boolean search, they make no statement whether passing is a good or bad move in a particular situation. It may very well be that one of the above rules forbids passing in a situation where passing is the only correct move. Nevertheless, the above ruling is correct, because in such a situation, non-optimal moves have been made earlier by that side (which is forbidden to pass now) in the sequence of moves leading to this situation. The following diagrams 12 - 15, also known as *mannen ko*, give an example. 'K' marks a ko-forbidden point.[5]

[5] In all 4 positions both sides have the option to pass which leads to seki. In diagram 12 Black can

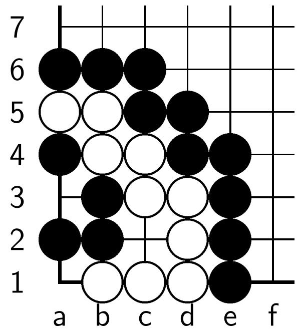


Diagram 12.

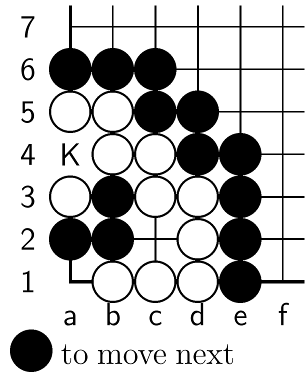


Diagram 13.

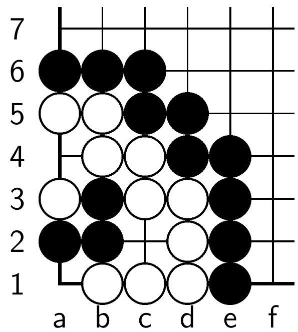


Diagram 14.

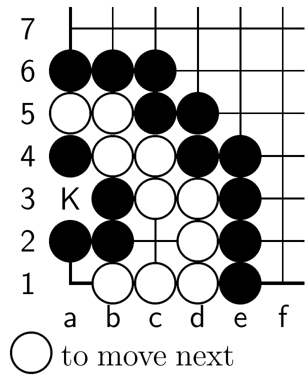


Diagram 15.

We consider diagram 15 and a run in which White has one external ko threat available. We assume further, White (non-optimally) uses the ko threat to recapture the ko (leading to diagram 13) and Black passes (correctly) leading to diagram 14. According to rule 1(a)i in this situation White is not allowed to pass although the passing move is the correct move for White. If White loses in this position because it may not pass then this has only the consequence that White has to try a different move two moves earlier and do the correct move there and pass. But the fact that the proper move is forbidden has a consequence for programming. GoTools runs a hash data base in which intermediate results are stored. If White loses in a position where passing was forbidden due to rules 1(a)i, 1(a)ii or 2b then the status of these positions and of positions in sub trees may not enter the database.

- Rules 1(a)i and 1(b)ii are applicable as well after swapping attacker (Black) and defender (White) but we do not have to change our ruling because this case is already included. The symmetry between attacker and defender is broken by treating seki=life and not seki=death. Rule 1(a)i after swapping Black $\leftrightarrow$ White is included in rule 1(b)i because in rule 1(a)i the side which is forbidden to pass had at least initially ko threats whereas in rule 1(b)ii it is the opposite side of the side that is allowed to pass which has a ko threat. In other words, rule 1(a)i after swapping colours does not forbid any passing which is not already forbidden by rule 1(b)i.

enforce a ko by playing on a1 or c2 which is about unconditional life or death and which is unfavourable for Black. In diagram 14 White can enforce a ko, unfavourable for White, by playing on a1 or c2. In a boolean search with seki=life the single best move for White in all 4 positions is to pass and the status is an unfavourable ko for Black.

For the same reason, rule 1(b)ii after swapping colours does not allow any passing for White which is not already allowed by rule 1(a)iii.

5 Summary

In this article we develop a consistent procedural definition of the status of life and death problems. The procedure performs a boolean search which has as a consequence minor limitations described in the appendix but which, on the positive side,

- determines how many external ko threats are needed for the weaker side to win,
- evaluates bent-4-type positions as dead and as the best move for the attacker to pass, and
- it is efficient because it does a boolean search and minimizes the number of passes done during the search.

The implementation in the program GoTools proved to be consistent when evaluating the status of more than $2 \cdot 10^8$ eyes with up to 11 inner points being surrounded by only one chain as reported in [5].

Appendix: Limitations

Boolean search: Because the outcome of life & death fights is polarized (life/seki or death) the value of such all-or-nothing fights is naturally high and it is critical to determine the status of the position and the best moves of both sides as early as possible. Thus, even with near unlimited computing power one would rather use it to solve problems earlier in the game even if one is limited to do a boolean search than optimizing the territorial value of life but being slow. The risk of losing few points by applying boolean search can be lowered by checking different first moves each in a boolean search and selecting that one of the optimal moves which in addition seems to give the most outside influence.

Seki equal life: Apart from the given position on the board we need for the definition of a life and death problem also one or more chains of one colour identified which are to fight for life (in this paper White tries to live) and possibly a side moving first, otherwise both sides moving first are considered. Having only two possible outcomes it is more appropriate to classify seki as life than as death. For example, if the White chain in diagram 16 would be regarded as unconditionally alive then White would have 10 points more compared to treating it as a seki (what it is).

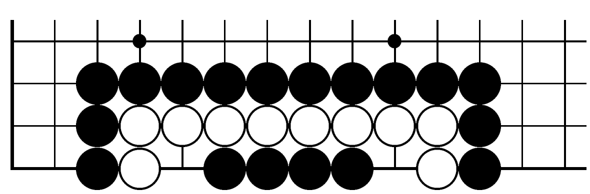


Diagram 16.

The error in regarding White as dead would be larger: 22 points. Nevertheless, this is a serious limitation as a difference of, for example, 10 points is not negligible. If one has already at the starting position intruding black stones then one can *confirm* seki with a boolean search by checking whether they can be caught. This should work for the classification of monolithic eyes, although it is not done in this paper. On the other hand, an extra run will not be able to detect seki if the intruding black chains do not already exist in the initial position.

Handling cycles: For the boolean search we need a rule how to handle cycles. The side moving next, in this paper called 1^{st}, is not allowed to restore a position encountered earlier in the sequence of moves that are already done

- if in the resulting cycle the opposite side 2^{nd} has caught more stones than 1^{st} (otherwise repeating this cycle sufficiently often would result in a loss of 1^{st} exceeding the value of the life and death problem), or
- if 1^{st} had spent external ko threats in this cycle and 2^{nd} not (in the computation described in section 2, only one side is allowed to use external ko threats to recapture ko's), or
- if 1^{st} = Black because repeating this loop would mean that White at least reaches seki, i.e. wins.

With these rules for handling cycles and rules about passing discussed in section 4.1, every search must be finite as problems of finite size can have only finitely many moves before the position must repeat.

Value of tenuki, i.e. passing in the life&death fight: A more serious restriction in our life and death computations comes from trying the passing move only as a last resort. Therefore, strictly speaking, the determined status and winning move are only correct under the assumption that playing elsewhere (i.e. passing in the l/d fight) has negligible value as it is typically the case towards the end of the game. Especially when comparing different ko's, the result may depend on the value of a passing move. Since the number of available external ko threats and the value of a passing move are in general incomparable, one ideally would have to determine the status for any combination of both, i.e. for the number of external ko threats and the number of passes. Our computations cover the special case where passing has negligible value.

External ko threats: As explained in section 2, the number of external ko threats is limited in practical computations to 5 but could easily be changed to any large value.% if that would be of interest.

Acknowledgements

The author thanks Volker Wehner, Robert Jasiek and Bill Spight for comments to the manuscript.

References

[1] T. Wolf: Forward pruning and other heuristic search techniques in tsume go, Special issue of *Information Sciences* **122** (2000), no 1, pp. 59-76.

[2] J. Davies, J. Cano, F. Hansen: The Japanese Rules of Go, `http://www.cs.cmu.edu/~wjh/go/rules/Japanese.html`

[3] R. Jasiek, Japanese 2003 rules, version 35a and commentary, `http://home.snafu.de/jasiek/j2003.html`

[4] Chi-Hyung Nam (`http://www.mju.ac.kr/chihyung`): private conversation.

[5] T. Wolf, M. Pratola: A library of eyes in Go, II: Monolithic Eyes, in this volume.

[6] T. Wolf: GoTools Online, `http://lie.math.brocku.ca/GoTools/applet.html`