WHEN ONE EYE IS SUFFICIENT: A STATIC CLASSIFICATION

R. Vilà
Facultat de Matemàtiques i Estadística, UPC, Barcelona.
In stage in Labo IA, Université Paris 8.
ritx@ai.univ-paris8.fr

T. Cazenave
Labo IA, Université Paris 8, 2 rue de la Liberté, 93526, St-Denis, France.
cazenave@ai.univ-paris8.fr

Abstract A new classification for eye shapes is proposed. It allows to decide statically the status of the eye in some restricted conditions. The life property enables to decide when one eye shape is alive regardless the number of opponent stones inside. The method is easy to program and can replace a possibly deep search tree with a fast, reliable and static evaluation.

Keywords: computer Go, eye, neighbour classification, life property

1. Introduction

It is well known for both, Go players and Go programmers, that when a string has two eyes it is alive. Though sufficient, it is not a necessary condition. Sometimes one big eye is sufficient to live, either it is possible to make two eyes at any moment, or it is alive in seki.

This paper deals with the classification of large eyes and when one big eye is sufficient to live. Here we propose an algorithm that gives statically an answer to that question. It is easy to program and very fast. We present the *neighbour classification*, a completely new concept that enables to group eye shapes with common interesting properties. We also introduce the concept of *life property* that permits to decide when one eye shape is alive regardless the number of opponent stones inside. This property relies only on the shape of the eye and, when applicable, is very powerful. It is a completely safe tool as no heuristics are involved. It can be applied to a wide variety of situations.

Section 2 describes the existing work related to handling eyes and life and death. Section 3 sets accurate definitions of the concepts used throughout this paper. New concepts like *end point* or *life property* are proposed. We also have enlarged Müller's (1999) concept of plain eye to cover statically more cases. Section 4 describes the main contribution of this paper, the *neighbour classification* and the theorem of the neighbour classification. Section 5 shows how to use the *neighbour classification* to identify the *vital* and *end points* for centre eyes. Section 6 discusses the limitations of the theorem for side and corner eyes and proposes possible ways to overcome them. Section 7 reports the application of this new theory to semeai problems. Finally, we suggest that the reader has a quick look at the first two paragraphs of Section 4 before reading Section 3 so that the captions in the figures of this section are clarifying instead of confusing.

2. Previous Work

Several approaches have been made to life and death and eye characterisation with great success.

Landman (1996) applies combinatorial game theory to determine a value for a given eye space. Fotland (2002) describes the way his program, THE MANY FACES OF GO, analyzes eyes. He represents eye shapes as its game tree with four different values; the upper and lower bounds on the number of eyes, and two intermediate values aiming to include the effects of ko and uncertainty. This work deals mainly with a big variety of general eyes. He combines static analysis with a small search.

Chen and Chen (1999) show a method to evaluate heuristically life for general classes of groups. Müller (1997) extends Benson's algorithm describing safety of blocks under alternating play.

Big eyes are of great importance in a wide variety of semeai problems. Though not being the key to the most common life-and-death problems, when they appear it is fundamental to handle them in a proper way. Most of the existing techniques treat them in an unsatisfactory way; either they treat them heuristically so unexpected situations may appear driving to a wrong answer, or they just let the search algorithm continue until they become a small eye with the subsequent inefficiency problems. Here we propose a theory and an algorithm to deal statically with this problem. It is very fast, easy to program, free of heuristic considerations and therefore completely reliable. It can replace completely a possibly deep search tree in a wide number of situations and it can be of great interest to enhance the existing techniques and to reduce the degree of inaccuracy.

3. Definitions

Eye. In this paper an eye[1] will be an area completely surrounded by one block. Opponent and own stones will be allowed in the eye region and also empty points not adjacent to the surrounding block. This is a generalization of Müller's (1999) definition of *plain eyes*. We will classify eyes according to its position on the board.

Corner Eye — The eye contains a corner point and its two neighbours (cf. Figure 1).

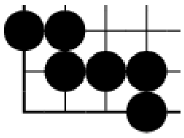

Figure 1. A corner [1122] eye, not plain.

Side Eye — The eye is not a corner eye and contains at least three side points (note: a corner point is a particular case of side point) (cf. Figure 2).

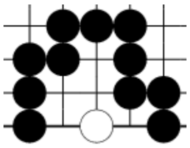

Figure 2. A side [112234] eye, plain.

Centre Eye — All the eyes that are not corner or side eyes (note: the most part of big eye shapes can only be centre eyes (Mathworld, 2003)).

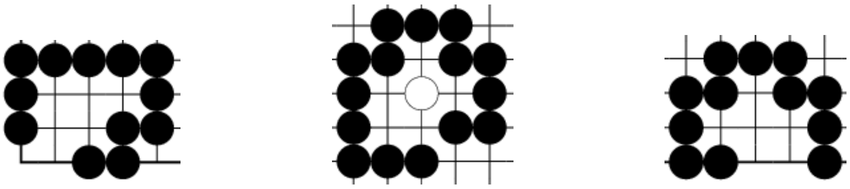

Figure 3. A centre [1122233] eye, plain (left); a centre [112224] eye, plain (middle), and a centre [112224] eye, not plain (right).

Eye Shape. This is the set of intersections of the eye. The intersections can be empty, or occupied by opponent or friendly stones. We will use the term *Nakade Shape* to refer to a set of intersections that, in case of being an Eye Shape, would have one or zero *vital points*.

Eye Status. We will define four possible status for a centre eye: *Nakade, Unsettled, Alive, and AliveInAtari.*

[1] In the existing literature *eye* is used to refer to a small one-point eye, while bigger eyes are referred to as *X-enclosed region* (Benson, 1976) or *Big eye* (Fotland, 2002). In this paper we mainly deal with big eyes, therefore as no confusion is possible we will keep the term *eye* as we define it.

Nakade — the eye will end up as only one eye and this will not be sufficient to live. A nakade eye can be the result of: (1) an eye with an empty set of *vital points* (cf. Figure 4b) or (2) an eye with all the set of *vital points* filled by the opponent's stones (cf. Figure 4a).

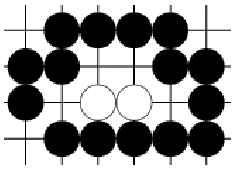

Figure 4a. A nakade status for a [112233]-α eye. The two *vital points* are filled by the opponent.

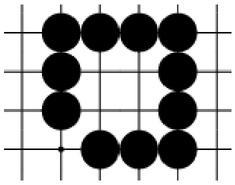

Figure 4b. A nakade status for a [2222] eye. It has an empty set of *vital points*.

Unsettled — the eye can end up as a nakade eye or an alive eye depending on the colour to play. An unsettled eye is the result of an eye with one and only one empty intersection in the set of *vital points* (cf. Figure 5). An unsettled status is what Landman (1996) defines as $1\frac{1}{2}\varepsilon$.

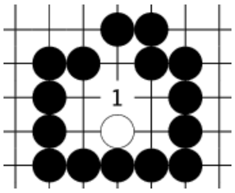

Figure 5. An unsettled status for a [1222234] eye. One of the two *vital points* is empty (1).

Alive — the string owning the eye is alive no matter who plays first and no matter what the surrounding conditions are. An alive eye can be the result of: (1) an eye with two or more empty intersections in the set of *vital points* (cf. Figure 6a) or (2) the eye is a n–shape that cannot be filled by the opponent with a $(n-1)$–*nakade shape* (Figure 6b). We will make no distinction between being alive or being alive in seki like in Figure 6b, as in many cases being alive in seki may be almost as good as living with two eyes (Landman, 1996).

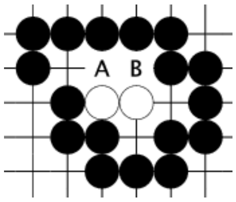

Figure 6a. Alive status for a [1112234]-β eye. Even though these shape can be filled with a rabitty six, A and B belong to the set of *vital points* so we have a miai of life.

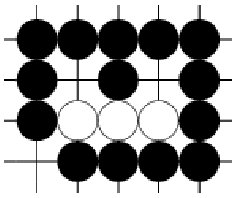

Figure 6b. Alive status for a [11222] eye. No matter how many stones plays White inside, Black is unconditionally alive. The opponent cannot fill the eye space with a *nakade shape* of size four.

AliveInAtari — this is a particular case in which the surrounding conditions determine the status of the eye. We say that an eye has an AliveInAtari status if there are only one or zero empty intersections adjacent to the surrounding block but capturing the opponent stones inside the eye grants an alive status. Only when the external liberties of the string owning the eye are played it is necessary to capture the stones inside the eye (cf. Figure 7a and 7b).

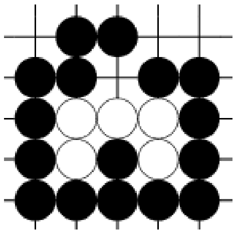

Figure 7a. AliveInAtari status for a [111223] eye. Capture grants life.

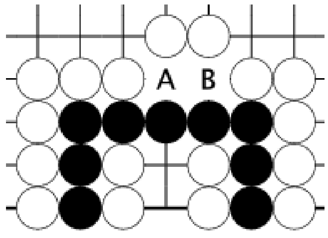

Figure 7b. AliveInAtari status for a [222233] eye. When A is played the status changes to Unsettled, and if both A and B are played the status is Nakade. However capturing the stones inside the eye grants an alive status.

Vital Points. A minimal set (one or more) of intersections inside the eye that should be filled by the opponent to grant a nakade status for the eye (cf. Figure 8a and 8b).

End Points. A minimal set (one or more) of intersections inside the eye that should not be filled by the opponent until the end to grant a nakade status for the eye in the process of killing the string (cf. Figure 8a and 8b).

This should not be confused with Fotland's (2002) *number of ends*. While Fotland's concept deals with the shape, our concept deals with the order in which the intersections of the eye should be filled by the opponent. In Figure 8a there is one end point but three Fotland's ends. However, in most cases we see that an end point from this paper's point of view is also a Fotland's end.

Life Property. We will say that an eye shape has the *life property* if the only possible status for this shape are Alive or AliveInAtari. Thus when an eye shape has the *life property* we only need to check whether the stones inside the eye should be captured due to an AliveInAtari status.

For example, a 3–shape in a line can have a Nakade, Unsettled, or AliveInAtari status depending on the opponent stones played inside. This 3–shape does not have the *life property* since a Nakade and Unsettled status are possible. In contrast, the [11222] shape showed in Figure 6b can **only** have an Alive status

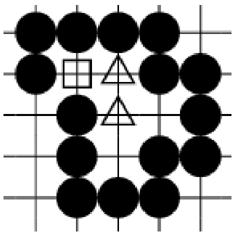

Figure 8a. *Vital* (△) and *end* (□) *points* for a [11123] eye.

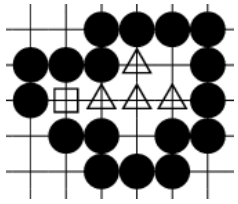

Figure 8b. *Vital* (△) and *end* (□) *points* for a [1122224] eye.

or an AliveInAtari status (when four out of the five intersections are played by the opponent), this shape has the *life property*. Therefore, while all the shapes having the *life property* are alive, not all the shapes having an alive status have the life property.

The *life property* should be regarded as a property slightly below Benson's (1976) unconditional life, because if we have an AliveInAtari status it might be necessary to play inside the eye, but with the great advantage that detecting it is just a matter of counting neighbours as it will be shown in Section 4.

4. Neighbour Classification

Let $\mathcal{E}_i$ be the set of all possible eye shapes of size i. Note that for $i = 1..6$ there is an isomorphism between $\mathcal{E}_i$ and $\mathcal{P}_i$ being $\mathcal{P}_i$ the set of free i–polyominoes (Mathworld, 2003). An n–polyomino (or "n–omino") is defined as a collection of n squares of equal size arranged with coincident sides. Free polyominoes can be picked up and flipped, so mirror image pieces are considered identical. For size seven we should discard the holed-polyomino to keep the isomorphism.

Let $e \in \mathcal{E}_i$, we define the *Neighbour Classification* of e, $NC(e)$, as a number of i digits sorted from low to high; every intersection in the eye space is associated to a digit that indicates the number of neighbours (adjacent intersections) to that intersection that belong to the eye space (cf. Figure 9).

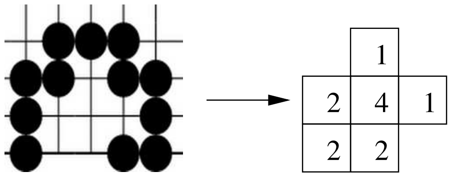

Figure 9. For the rabbity six $NC(e) = 112224$.

Let $\sim$ be the following equivalence relation: let $e_1, e_2 \in \mathcal{E}_i$ then

$$e_1 \sim e_2 \iff NC(e_1) = NC(e_2)$$

Thus $\sim$ gives a partition of $\mathcal{E}_i$ defined by the equivalence classes in $\mathcal{E}_i \,/\!\sim$ (cf. Appendix B).

Example. Given $\mathcal{E}_5$ we can find four different *neighbour classifications* for its elements (Note that $\|\mathcal{E}_5\| = \|\mathcal{P}_5\| = 12$) (Mathworld, 2003).

$$NC(e) \in \{11222, 11123, 11114, 12223\}, \quad \forall e \in \mathcal{E}_5$$

THEOREM 1 (OF THE NEIGHBOUR CLASSIFICATION) *Let e be a centre eye, $e \in \mathcal{E}_i$ and $[e] \in \mathcal{E}_i \,/\!\sim$ the equivalence class of e, for $i = 1..7$ if e has the* life property *then $\forall f \in [e]$, f has the* life property *inversely if e has not the* life property *then $\forall f \in [e]$, f has not the* life property.

Proof: For $i \in \{1, 2, 3, 4\}$ there is no eye shape that has the life property so the theorem is correct. The 1–shapes and 2–shapes are always nakade, the two existing 3–shapes have one vital point so their status can be nakade or unsettled (depending on the fact whether the opponent has or has not played the vital point). There are five 4–shapes with zero, one or two vital points. All of them can have a nakade status if the opponent plays all the vital points. The interesting point comes with higher size shapes.

Under the conditions of the theorem, having the life property is just a matter of shape. If and only if an i–shape cannot be filled by the opponent with an $(i-1)$–shape that has one or zero vital points (*Nakade Shape*), then this i–shape has the life property.

Ko cannot arrive in the centre for eye shapes of size below seven. For size seven there are only two shapes that can have a ko status in the centre (cf. Figure 10). These shapes do not have the life property as they can be filled by a rabitty six so the ko does not interfere with our theorem.

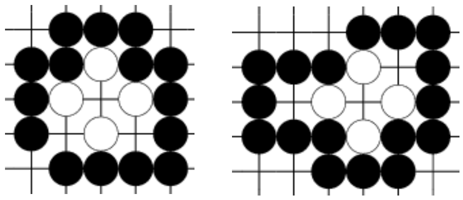

Figure 10. Shapes in classes [1222234] and [1122224] can have a ko status in the centre.

5–shapes. There are two *nakade shapes* of size 4 (the square and the pyramid) so all the 5–shapes that do not contain a square or a pyramid will have the *life* *property*. All the shapes belonging to [11222] have the life property while the others do not. The only *nakade shapes* of size five are the bulky five and the star.

6–shapes. As before, all the 6–shapes that do not contain a bulky five or a star have the *life property*. These are the 26 shapes belonging to classes $\{[112222], [111223], [111133]\}$. The only *nakade shape* of size six is the rabbity six.

7–shapes. Now we only have to care about those shapes containing a rabbity six, there are five shapes distributed in four equivalence classes. All the other 7–shapes have the life property. There is no *nakade shape* of size seven. This concludes the proof of Theorem 1.$\square$

The exhaustive classification for all the eye shapes under size eight is summarized in Table 1.

$\mathcal{E}_i$	$\mathcal{E}_i/\sim$	$\lvert\cdot\rvert$	*Life Property*
$\mathcal{E}_1$		1	
	[0]	1	No
$\mathcal{E}_2$		1	
	[11]	1	No
$\mathcal{E}_3$		2	
	[121]	2	No
$\mathcal{E}_4$		5	
	[1122]	3	No
	[1113]	1	No
	[2222]	1	No
$\mathcal{E}_5$		12	
	[11222]	7	Yes
	[11123]	3	No
	[11114]	1	No
	[12223]	1	No

$\mathcal{E}_i$	$\mathcal{E}_i/\sim$	$\lvert\cdot\rvert$	*Life Property*
$\mathcal{E}_6$		35	
	[112222]	13	Yes
	[111223]	12	Yes
	[111133]	1	Yes
	[112233]	4	No
	[122223]	2	No
	[112224]	1	No
	[111124]	1	No
	[222233]	1	No
$\mathcal{E}_7$		107	
	[1122222]	30	Yes
	[1112223]	40	Yes
	[1122233]	11	Yes
	[1111233]	8	Yes
	[1222223]	5	Yes
	[1111224]	4	Yes
	[1112333]	2	Yes
	[1222333]	2	Yes
	[1112234]	2	No
	[1222234]	1	No
	[1122224]	1	No
	[2222224]	1	No

Table 1. Neighbour Classification for $\mathcal{E}_i, \quad i = 1..7$.

The strength of the theorem lies in the fact that the *life property* only depends on the shape. So given an eye shape we have just to find its neighbour classification. If the class has the *life property*, whatever number of opponent stones inside, we know that the group owning the eye is alive, we only need to check if it is necessary to capture the stones inside the eye due to an AliveInAtari status. If the class does not have the *life property* a further study is required to decide the status (cf. Section 5).

We cannot extend the theorem for higher sizes in the centre because ko's and opponent eyes may appear. But we will see that usually it is not much of a problem as the *life property* is an excessively strong condition for such that eyes.

5. Vital Points and End Points Identification

Another interesting property of the *neighbour classification* is that it allows, for centre eyes, to find the *vital* and *end points* for a given eye shape just looking at its signature. Below we will show the identification for the five classes of size six without the *life property*. The identification for eye shapes with sizes from one to five is easy to find out and size seven requires an analogue procedure as size six.

For size six we have five different classes without the life property and thus, the status should be checked.

[112224] — The rabbity six is the only nakade shape of size six. The vital point is the 4-neighbour point. The 2-neighbour point not neighbouring the vital point may be considered an end point (cf. Figure 11). Though not necessary to be filled at the end, only if filled we should test for a non nakade shape inside.

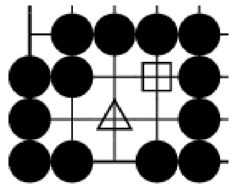

Figure 11. *Vital* and *end points* for [112224].

[[111124] — Vital points are $\{2{,}4\}$-neighbour points and the end point is the 1-neighbour point neighbouring the 2-neighbour point (cf. Figure 12).

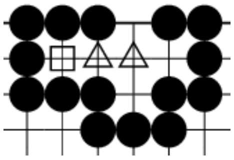

Figure 12. *Vital* and *end points* for [111124].

[[222233] — Vital points are the two 3-neighbour points (cf. Figure 13). There is no efficient way to define end points. So we should always test for a nakade four zigzag inside.

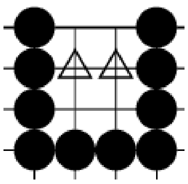

Figure 13. *Vital* and *end points* for [222233].

[[112233] — We need to create two subclasses in this class to perform the identification. We define class [112233]-α as the subset of two elements in class [112233] in which $\{3{,}3\}$ are neighbours and the class

[112233]-β as the subset in which $\{3,3\}$ are not neighbours. So for α elements are two vital points corresponding to the $\{3,3\}$-neighbour points and two end points corresponding to the $\{1,1\}$-neighbour points. For β elements the end points are the same, but all the other intersections are vital points (cf. Figure 14 and 15).

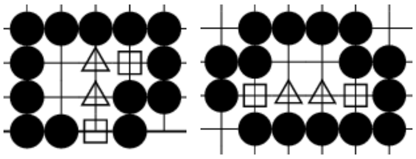

Figure 14. *Vital* and *end points* for [112233]-α.

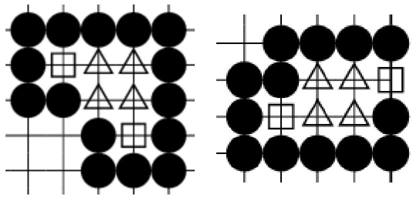

Figure 15. *Vital* and *end points* for [112233]-β.

[[122223] — The end point is the only 1-neighbour point. For vital points we need to consider the 3-neighbour point and its three neighbours (cf. Figure 16).

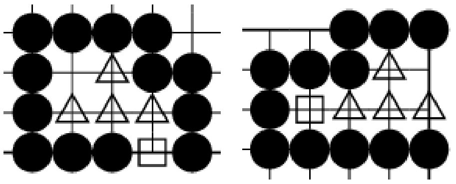

Figure 16. *Vital* and *end points* for [122223].

We do not know a unique way to find the vital and end points for no matter what kind of shape. So far a case by case implementation is needed, but in the process, the neighbour classification efficiently helps to determine them for each given class.

Once the identification is done it is possible to give the status and the hot point to play inside the eye, if necessary, depending on the opponent and friendly stones played in the eye shape (cf. Appendix A).

6. Corner and Side Eyes

To approach corner and side we should first remark the following implication (NoLP = No *Life Property*):

$$NoLP\ Centre \Rightarrow NoLP\ Side \Rightarrow NoLP\ Corner$$

Thus, once the study for centre eyes is done only classes with the *life property* in the centre need to be checked in the border and the corner.

For side eyes, theorem 1 continues to be true for sizes from one to four. For sizes five and six, ko only appears in classes that do not have the *life property* in the centre so the theorem continues to be true. For size seven there are two classes ([1222333] and [1112333]) that have the *life property* in the centre but fail to have it in the side due to ko situations. Unfortunately class [1122233] has seven out of eleven members that do not have the *life property* due to ko while the other four continues to have it in the side (cf. Figure 17a and 17b). So the theorem is no longer applicable for side eyes with size seven.

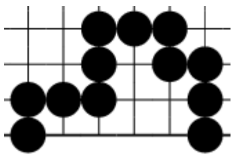

Figure 17a. This element of class [1122233] has the *life property* in the side.

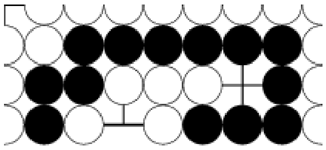

Figure 17b. This element of class [1122233] has not the *life property* in the side due to ko status.

Even though the theorem fails for side eyes, there is still a lot of knowledge that can be used for an implementation to solve side eyes. We will only have to consider more special cases. Shapes that do not have the life property will need to be treated more carefully in the side. For example, we have seen that a [112233]-α shape has two vital points. Therefore, if no vital point was played by the opponent, in the centre we had an alive status. This is no longer true in the side as Figure 18 shows. What might be called an *Unsettled-Ko* status appears for side eyes.

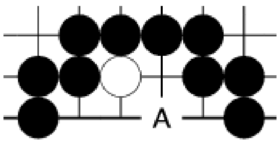

Figure 18. If White plays A a ko will appear. If black wins the ko the status will be alive, if loses will be nakade.

In the corner the situation is worse. Bent four in the corner, ko's and the possibility for the opponent to make easily an eye inside the big eye makes the corner a difficult battleground to apply the theorem.

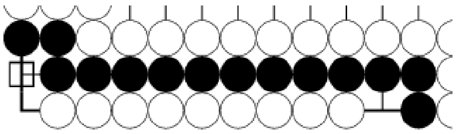

Figure 19. A 12–size corner eye without the *Life property*. If White plays □ a ko status appear.

It is the moment to remark now how strong the condition of having the *life property* is. Strange examples of eye shapes in the corner can be found. They do not have the life property (ko's can arrive) but they are almost impossible to kill in a real game (cf. Figure 19).

However, the fact that the theorem is not applicable in the corner does not mean that the *neighbour classification* is useless in those cases. It can be used to classify shapes in a straightforward way. For example, for size six there are only 12 out of 35 shapes that can be corner eyes. Six of these 12 are shapes of class [112222] and [111223]. These shapes can have their status easily decided depending on the opponent stones played inside. For the other six shapes we can just return an unknown status and let the search continue until they become a size five corner shape which are not so hard to decide by means of a case by case implementation.

7. Application to Semeai Problems

The *neighbour classification* has been successfully used and tested in semeai problems. Following Müller's (1999) classification of semeais and using the *neighbour classification* we have been able to solve statically classes from 0 to 2, but also all the semeais with centre and side eyes which are over class 2, either because the eye is not plain or because there are more than one non essential block inside the eye. This signifies an improvement over the results achieved statically and reported by Müller (1999).

A representative subset of semeai problems solved using the *neighbour classification* can be found at `www.ai.univ-paris8.fr/~ritx/semeai.zip`.

8. Conclusions

Three new ideas about eyes are presented in this paper: the concept of *end point*, the definition of *life property*, and the *neighbour classification*.

The *neighbour classification* and the *life property* perform a completely safe tool for deciding eye status statically under some restricted conditions. The method is easy to program and can, in many situations, replace a possibly deep search tree with a fast, reliable and static evaluation.

For eye shapes that do not have the initial conditions, like side and corner eyes, we have shown that still a great deal of useful knowledge coming from the *neighbour classification* can be used.

It has been tested for semeai problems and proved to be a powerful tool.

References

Benson, D. (1976). Life in the game of Go. *Information Sciences*, Vol. 10, pp. 17–29.

Chen, K. and Chen, Z. (1999). Static analysis of life and death in the game of Go. *Information Sciences*, Vol. 121, pp. 113–134.

Fotland, D. (2002). Static Eye Analysis in "The Many Faces of Go". *ICGA*, Vol. 25, No. 4, pp. 203–210.

Landman, H. (1996). Eyespaces Values in Go. *Games of No Chance*, Vol. 29, pp. 227–257.

Mathworld (2003). http://mathworld.wolfram.com/Polyomino.html.

Müller, M. (1997). Playing it safe: Recognizing secure territories in computer go by using static rules and search. *Game Programming Workshop in Japan '97* (ed. H. Matsubara), Computer Shogi Association, pp. 80–86, Tokyo, Japan.

Müller, M. (1999). Race to capture: Analyzing semeai in Go. *Game Programming Workshop in Japan*, Vol. 99(14) of *IPSJ Symposium Series*, pp. 61–68.

Appendix A: Implementation for 6–shapes centre eyes

Below we present the general guidelines for an implementation of an algorithm to decide the status for size six centre eyes. We suppressed irrelevant details. Eye and Rzone should be regarded as classes that allow to store a set of intersections on the board. The names of the variables have been chosen to allow reading the implementation as if it were pseudo-code.

`FindShapeVitalEnd` takes `e` as input, decides the `shape` using the *neighbour classification* and initializes `vital` and `end` variables using the explanations already given in Section 5.

`InitLocals` takes `e`, `vital` and `end` as input and initializes `EyeFilledSpace`, `vitalFilled` and `endFilled`. The "Filled" variables contain the intersections in the eye, the vital zone and the end zone that are filled with opponent stones.

```
typedef enum eye6_t { t112224,  t111124,  t222233,  t122223,   t112233a,
                      t112233b, other6 };

void Size6_Centre( Eye &e ) {
  eye6_t shape = other6;
  Rzone vital, vitalFilled, end, endFilled, EyeFilledSpace;

  FindShapeVitalEnd( &shape, &vital, &end, e );
  InitLocals( &EyeFilledSpace, &vitalFilled, &endFilled, vital, end, e);

  switch( shape ){
  case other6:                        //the shape has the life property
    if( EyeFilledSpace.size() == 5 )
      e.setEyeStatus( AliveInAtari );
    else
      e.setEyeStatus( Alive );
    break;
  case t111124:
    if( endFilled.size() == 0 ){
      switch( vitalFilled.size() ){
      case 0:
        e.setEyeStatus( Alive );
        break;
      case 1:
        e.setEyeStatus( Unsettled );
        //set Hot Spot: the intersection in vital not present in
        //              vitalFilled
        break;
      case 2:
        e.setEyeStatus( Nakade );
        break;
      }
    }
```

```
    else{
      if( EyeFilledSpace.size() == 5 )
        e.setEyeStatus( AliveInAtari );
      else
        e.setEyeStatus( Alive );
    }
    break;
  case t122223:
    if( endFilled.size() == 1 ){
      if( EyeFilledSpace.size() == 5 )
        e.setEyeStatus( AliveInAtari );
      else
        e.setEyeStatus( Alive );
    }
    else{
      switch( vitalFilled.size() ){
      case 0:
      case 1:
      case 2:
        e.setEyeStatus( Alive );
        break;
      case 3:
        e.setEyeStatus( Unsettled );
        //set Hot Spot
        break;
      case 4:
        e.setEyeStatus( Nakade );
        break;
      }
      break;
    }
  case t222233: [...]
    break;
  case t112224: [...]
    break;
  case t112233a: [...]
    break;
  case t112233b: [...]
    break;
  }
}
```

Appendix B: Equivalence classes for {5,6,7}–shapes

Below we present the complete set of eye shapes of size five, six and seven grouped by equivalence classes.

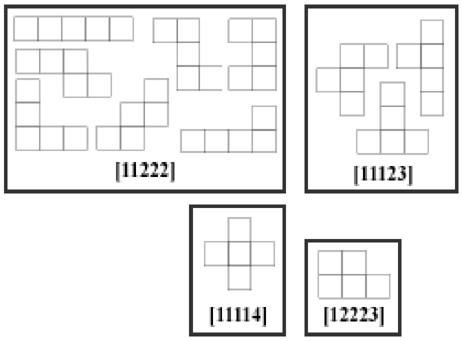

Figure 20. The complete set of pentominoes grouped by classes.

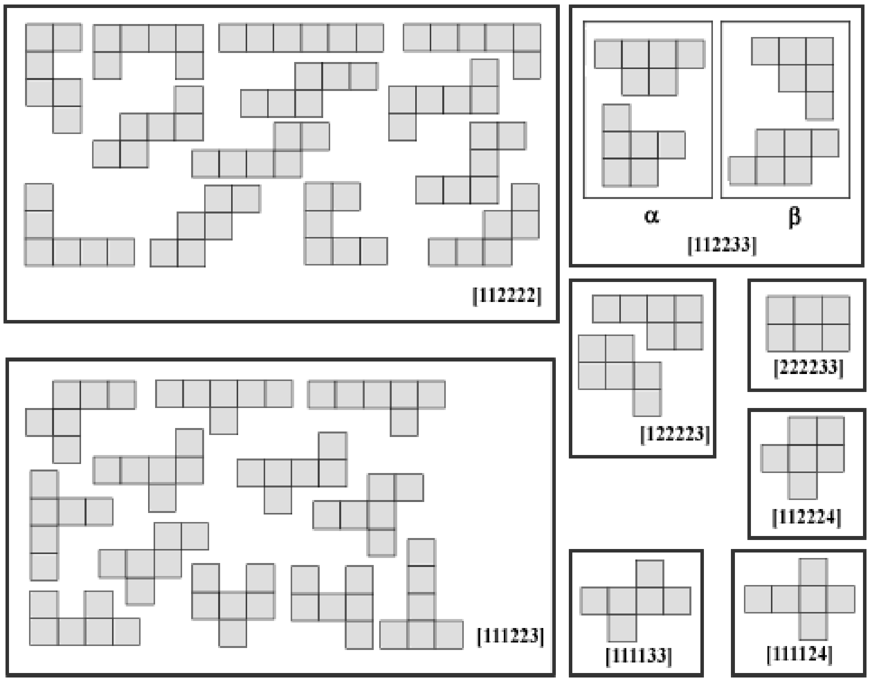

Figure 21. The complete set of hexominoes grouped by classes.

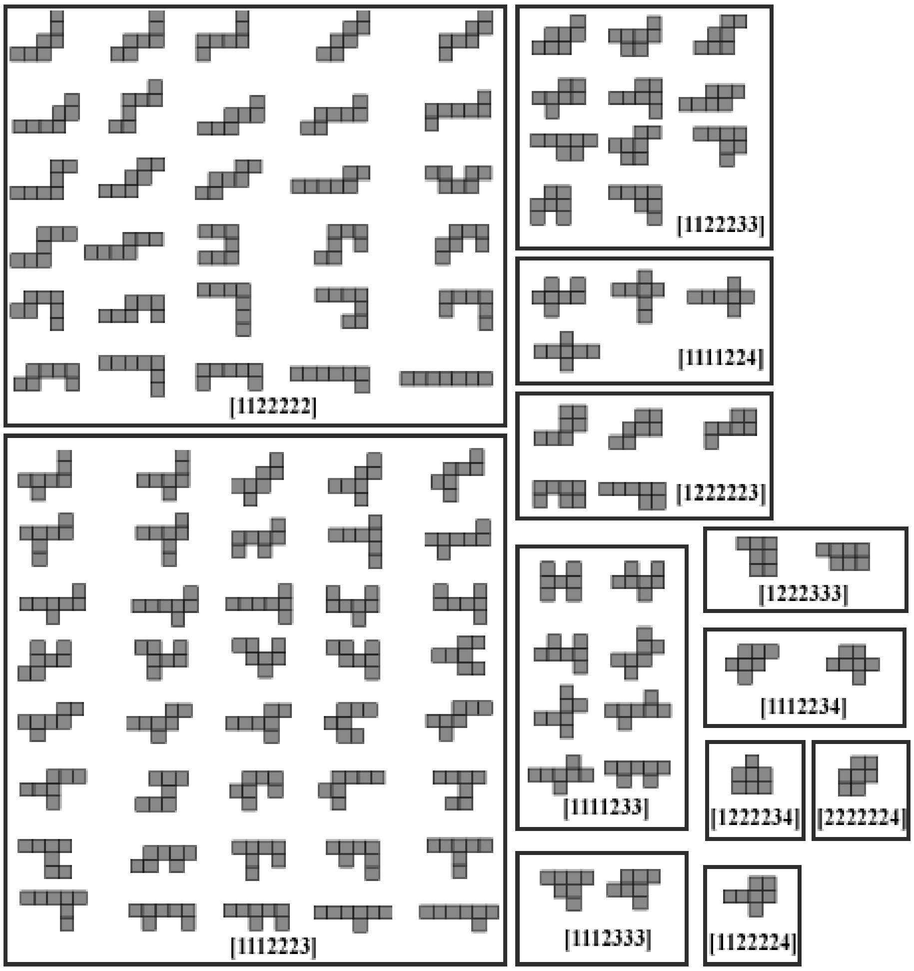

Figure 22. The complete set of size seven eye shapes grouped by classes.