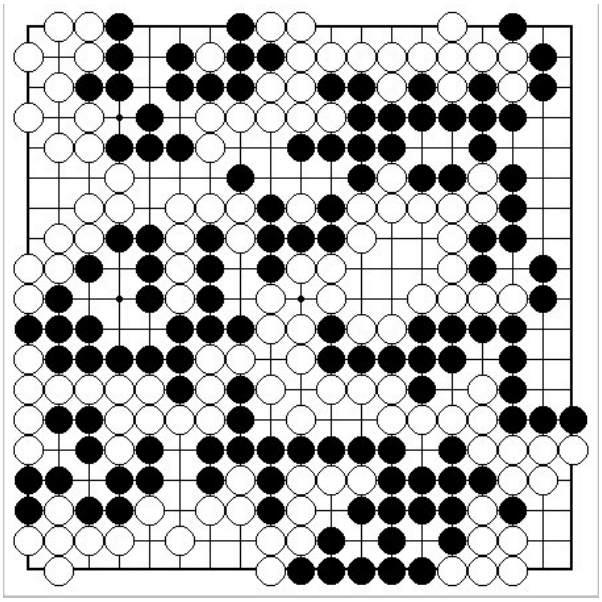
Towards Abstraction in Go Knowledge Representation



by Bjorn Goldis

Research Paper for Dr. K. Cooper's
Advanced Artificial Intelligence
CAP 6680
April 27, Tuesday, 1999

I Introduction:

Why Games

Games are to AI what car racing is to the automotive industry. [1]

Games describe a micro-world about which reasoning in computational terms can be the subject of experimentation, where, unlike in the real world, all the conditions pertaining to the outcome of specific results can be understood. However, even fully accessible, deterministic and discrete games like chess have branching factors that prevent complete searches of possible states. In this sense actions must be taken [on the part of a program] before certainty is available; as such, games are much more like the real world than the problems given by other domain specific areas for which AI has had more success.

AI is most successful in those areas where heuristics can map to possible event instances, and least successful where not rules, but general principles are required due either the inapplicability or unavailability of an appropriate rule. Games require general principles because their rules are only regulative, not prescriptive; they do not function like an algorithm, which produces a closed, limited, or a predictable output, but only an open ended, envelope defining set of limitation. To the extent that AI attempts to develop the capacity of generalization and become less bound to prescriptive heurism, it is emulating the human mind. If we can say that the operating base of computing resides in Sequence, Selection, Alternation and Case, and that of the mind in the Categories of Quantity, Quality, Relation and Modality[2], then the process of acquiring understanding of mental processes resides in the possibility of moving from rule based or analytical techniques to principle based synthesizing techniques. While epistemology explains the concept of an unknowable mechanism, AI attempts to construct a mechanism of an as yet undiscovered concept that will allow for principle based reasoning instead of mere search heuristics. So the goal of AI therefore becomes as much a process of coming to understand the mind in computational language with a program as a result and proof of concept, as only producing new applications as such.

In the short term, there are two possible outcomes to investigating games:

- If the program can defeat the best human players, analysis of the techniques which led to the improvement may uncover new AI techniques
- if the playing strength continues to fall short even after prolonged periods of research, at least a better understanding of the nature of the problems involved in performing at a high level may be understood. [3]

Why Go

"The research of Go programs is still in its infancy, but we shall see that to bring Go programs to a level comparable with current chess programs, investigations of a totally different kind than used in computer chess are needed." **--John McCarthy** [4]

Early AI researchers chose Western Chess --as opposed to other forms such as Chinese chess ChangKi, or ShoGi (Commander Chess), or Wei Chi (Surrounding Chess: Go)-- as the abstract nature of the game and the relative simplicity of the rules allowed for representation as a state space search. A chess-playing computer would serve as an existence proof of a machine performing intelligently.[5] However, the first performance jump not from better algorithms or evaluation functions, but hardware**.** [6] As a one-time insei (aspiring Go professional student), programmer, and doctoral student in Neural Science writes,

The story of computer chess had begun to sound like the children's book Fortunately, by Remy Charlip: Fortunately, the further computers could think ahead, the better their moves were. Unfortunately, deeper searches took longer. Fortunately, computer hardware doubled in speed every couple of years. Unfortunately, that speed encouraged investigators to abandon high-level processing and focus on search alone. [7]

From the qualitative side, the ability of hardware to overcome high-level processing vitiated the effort which originally was the focus of AI: research in and discovery of the representation and formalization of human like synthetic reasoning, and instead, the reliance on speed for mechanical state space search became predominant. However, speed alone is not adequate to the task in Go given its overwhelming state space:

..A Go computer as fast as Deep Blue (which analyzed some 200 million chess positions per second) would take a year and a half to mull over a single move. [8]

While a chess branching factor averages 35 move per ply; a Go board with its 361 points averages 250 moves per ply:

Even if a go algorithm could prune **that number** [250 moves/ ply] down **to fifteen or sixteen**, and consider 200 million positions per second, it would need **seventy years** to explore as deeply as Deep Blue does in **three minutes**.

Very well, one might say; computers will get faster soon enough. But the problem goes deeper than that. Go programs not only fail to evaluate 200 million go positions a second; they **cannot** accurately **evaluate** a single one.[9] [Bold added]

The source of the complexity of Go lies partly in the size of the playing board, but mostly in the nature of the game. Numerically, complexity is of two types;

- State Space Complexity: The number of legal game positions reachable from the initial position of the game.
- Game Tree Complexity: An estimate of the size of a minimax search tree which must be built to solve a game [10]

For Go, the state space can be calculated starting from 3^{361}, which equals 10^{761}. This is reduced to $10^{160 \text{ to } 500}$ depending on the author attempting to conservatively eliminate useless positions. Each point can be Black, White or empty. Its game tree complexity would be average game length 250 ply, and with an average of 200 branching states would yield 200^{250} or approx. 10^{575}. This last number is reduced by Allis to 10^{360}, but on the other hand longer professional games of almost 300 ply are not uncommon, with the current longest having 411 moves.

[Move number can exceed the total points due to the fact that playing pieces can be place in spaces vacated by former captures, and also due the recurring capture of a single stone (with intervening plays to prevent invariance) called "Ko" < "perpetual", Japanese >. See Appendix for rule set.]

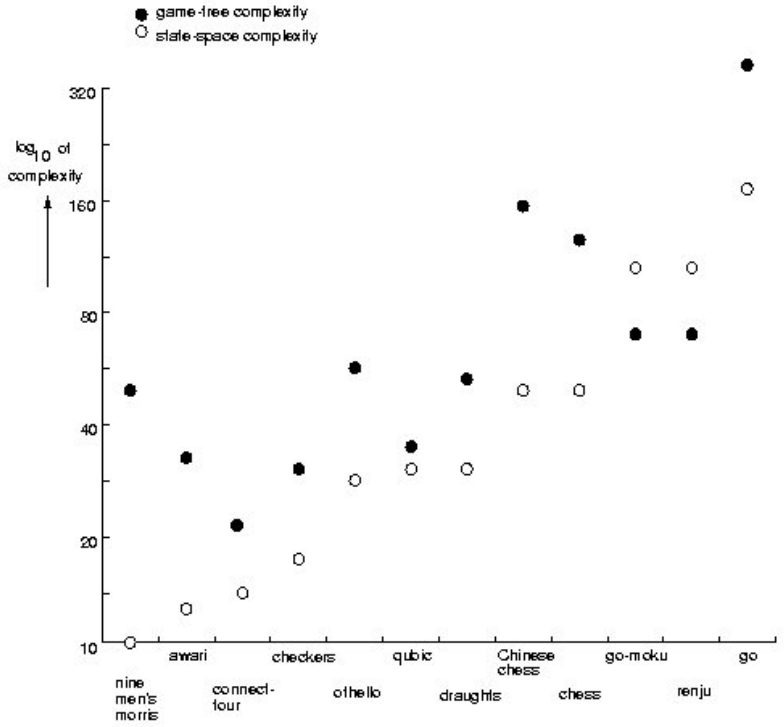


Figure 6.1: Estimated game complexities.

11

A Go game played on a 9 x 9 size grid, instead of the standard 19 x 19, would be comparable to the complexity of Western Chess, having 35 versus 50 in log10 search space, with a game tree complexity of about 50^60. ($8 * 10^{101}$) This estimate is borne out in the life and death analysis problems mentioned later in the paper.

How human players manage avoiding searching through the state space may not be known ultimately, but Go players use Strategic language to describe the process. The strategic concepts used in Go are more abstract than those in other games, including Chess games, for Chess relies on the individuality and discrete identity of the individual pieces to order the evaluation process. A strategic text in Chess will limit itself to the explanation of how to manipulate sets of pieces; n rooks and y bishops vs. m knights and z pawns. Various sets are described: to the extent that principles are enumerated, they are bound to particular instances of existing pieces. Other Chess ideas like time, space, position, tempo, apply to Go as well. However, Go has some others too.

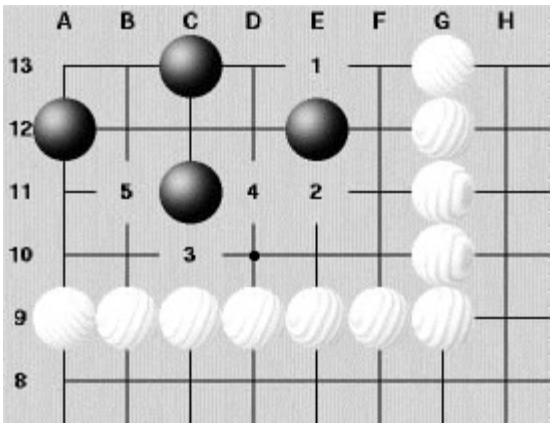
Go concepts, such as 'Thickness', 'Force', 'Potential', 'Probe', 'Lightness', 'Heaviness', 'Exchange' and 'Ambivalence' (for 'miai') relate not to pieces as such but to the context and relations of stones to other stones and groups. The stones are uniform in value and function; context determines a functional value only in relation to the state of the board as a whole, and the values of each stone change with each new stone placed on the board. The changing values brought about by changing contexts promote an ambiguity similar to the ambiguities found in the semantics of natural language processing, with the punctuation and spacing between the 'words' of the stones and groups shifting in the changing context of evolving formations as well.

The Go concepts mentioned are therefore used as protagonists in a story providing the elements to synthesize the manifold elements of actual and possible permutations, as the thematic linkage between the transfinite realm of evanescent probabilities and the requisite search space of actualized exigencies. For the most part, search techniques and pattern libraries have been used in Chess and Go; while Chess has benefited with this approach and fast processors alone, Go has not proved amenable to such stereotyped technique and technology, as far as developing performance abilities comparable to human experts.

Chess went from a beginner level of 1400 (MacHack 1965) to National Master strength of 2200 (Belle 1982) and most significantly, to a 'Professional' level of 2400 (Hitech 1987) and beyond to a Kasparov of 2800 (Deep Blue 1997). This means 25 years elapsed between beginner level and Professional @ 2400, in general terms. In the almost 30 years since Go game programs began with Zobrist in 1970, to Handtalk at 5 kyu in 1999, the state of the art is perhaps 1000 in Chess terms.[12]

Search may not be a sufficient technique for achieving significant ratings in Go, but it is necessary at some level of the game. It will be seen that the level at which search is useful compares to what is accomplished in Western Chess; namely, that minimax search has been and ought to continue to be useful on a dimensional scale comparable to the 8 x 8 chess board, and which in Go is concerned with the life and death situations of individual groups. See Appendix for examples of rules and other examples.

A qualitative aspect of the complexity in Go can be seen in the example of a Life and Death problem which took a Pentium class processor 6 hours to solve. [13]
Nevertheless, this problem is 'only' in the amateur Dan level degree of difficulty. [Dan = Chess 1600 - 2000 approx.] A program that is to gain proficiency on a Professional level would have to be able to solve such problems in terms of seconds to minutes, not hours. Professional level L&D problems can take this L&D program *days to more than a week to solve*. The program was given a rating of between amateur 1 and 5 Dan.[14]



White To Kill Unconditionally

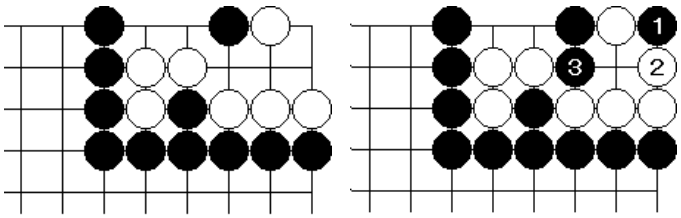
Finally, notice that in terms of size, this problem is comparable to a chess board; i.e., it is 7 x 5 in its grid size. In other words, solving a small scale, isolated L&D problem alone can take more time than a world class Chess Match game with present Pentium hardware..

Now we will look at some techniques used to prune the search trees in Life and Death situations.

II Searching Limitations

An example of pruning searches in Go positions concerned with Life & Death illustrates the use and benefits of search in this positionally limited analytical context, along with some caveats. The times given are using a 10 MHz processor and Lisp language, and the times can therefore be reduced by a factor of 10 as an approximation when compared to Pentium processors.

First, a sample problem is given. [15] [See Appendix L&D Search for human solutions to this one.] This is a problem on a beginner level, perhaps in the 15 Kyu range. Nevertheless, it has a few interesting characteristics. As mentioned in the Appendix, the three outcomes are Life, Death, and Seki. These can be combined with Ko but will not be considered here.



BLACK to move WHITE to move

Time is the time required to solve the problem
Nodes is the number of nonterminal nodes with children
Evals is the number of positions statically evaluated
Seen is total positions seen, including cached positions which didn't require evaluation.

#1: Full tree, no cutoffs at all. All nodes @ depth limit 15 ply.
Time: 1 hour 49 minutes 56 seconds, 22605 Nodes, 51421 Evals

To start with, one should notice the ply depth is at 15, though there are only 6 points on which to make a move and a response. This is actually part of the problem of searching in Go positions: If Black were to start with the cut of 3, White could play a sequence that easily lasted 11 moves; true, a person would 'see' them as incorrect, but the search doesn't 'see', it has to find. Also, as stones are captured, there are resultant spaces left into which more moves can be played. A Ko involvement can double the moves played as well.

Therefore, the search here involved 50 K positional evaluations. As the programmer observes;

Some "simple" heuristics could prune this tree by a large factor; for example, considering the problem to be solved when any large group is captured. The **problem with such simple heuristics is that they work in some situations, but cause the correct line of play to be ignored in others.** The challenge is to limit the search without losing the correct result. [15] [Bold added.]

#2: Alpha-Beta Only (reverse ordering)
Time: 27 minutes 50 seconds, 4941 Nodes, 11835 Evals

Transpositions occur in sequences where the same stones end up in the same positions, having taken a different order to get there.; i.e. a b c d, c b a d.
So sometimes the result can be reused. However, it is quite common that a change in order can change the result, so it is not absolutely true all the time. Still, this has been used in examples 3 and 5: [Underline added.]

Example 3 & 5 shows the effect of **caching results**.

```
#3: Alpha-Beta  (reverse ordering), Result Caching
Time: 4 minutes 55 seconds,  680 Nodes, 1477 Evals

#4: Alpha-Beta Only (good ordering)
Time: 2 minutes 25 seconds,  198 Nodes, 580 Evals

#5: Alpha-Beta (good ordering), Result Caching
Time: 57 seconds,  141 Nodes, 399 Evals, 506 Seen
```

Making adjustments to the ordering of the search can quickly reduce the search amount, however as the programmer notes:

With a reasonably good ordering in **example 4**, *alpha-beta* reduces the search by another factor of 20, to about 600 positions. Unfortunately, ordering the alternatives is a black art; any adjustments which solve one problem faster tend to make other problems solve more slowly**.** In the example, the correct first move is to play in the corner; any ordering that made this alternative the most likely will probably perform poorly for most other problems. [15]
[Underline added]

Not only is the first move in the corner unusual from a point of view of tactical sense, it also involves immediate capture. This is a sacrifice move though not 'seen' as such by the program. Humans might avoid looking at such a move as it appears self-defeating, though with experience it becomes a routine tool.

The "good" ordering used by the example is based on a balance of all the factors you might expect, including liberties, eye shapes, connectivity, cutting points and so on. [15] [Bold added]

This comment indicates the type and scope of conditions required in the evaluation of Life & Death positions; as the stones remain in place in Go, other conditions must be evaluated in the search for capture. In the position changing, or kinetic, games like checkers, backgammon and chess, a capture in itself concerns itself with the elimination of the piece from the board, thereby simplifying the remainder of the search.

#6: Alpha-Beta + Result Caching, Preset i-can-get
Time: 53 seconds, 118 Nodes, 330 Evals, 411 Seen

Presetting involves assuming a value before the search; an example is to set the initial value to a values just slightly less than the estimated value. Normally, the initial value in NegMax form of mimimax is minus infinity, and values increase through the search. [NegMax takes max of children, then returns the negative value.] However,

The hazard of presetting the search parameters is that if the actual value of the search is not inside the initial bounds, **you will complete the search with no useful information** except that the value is outside the initial bounds. In most Go problems, it isn't useful to make *any* move in a lost position, so initializing *i-can-get* to something between a *seki* and a loss will not lose any information you really need; Example 6 shows that presetting reduces the cost of search by almost 20%. [15]

Although it is true that in most positions it isn't useful to make a move in a lost position, it can be in the case of Ko fight, where a threat is required, or in Chinese stone counting rules. But this is not a major issue in the amateur Kyu level programs; most beginners are averse to Ko anyway.

#7: Alpha-Beta + Result Caching, Preset i-can-get, killer heuristic
Time: 54 seconds, 121 Nodes, 327 Evals, 364 Seen

The 'killer heuristic' attempts to improve ordering with when moves found have previously been found to be 'good' in similar situations. But ,

Deciding what is ***similar***, and remembering which moves were useful in those circumstances, is an open problem.
[Bold & ULine added]

#8: Alpha-Beta + Result Caching, Preset *i-can-get*, killer heuristic, sibling promotion
Time: 46 seconds, 124 Nodes, 341 Evals, 371 Seen

Finally, move branches are promoted when 'killer heuristics' are found occurring in sibling ordering. Again, not a perfect solution as ..

.. in example 8, adding sibling promotion actually makes the search examine slightly more nodes. This is unfortunate but not surprising **- in general any of these "optimizations" can have the opposite effect in any particular instance.** [15] [Bold Added]

One final observation: These search techniques omit indicating that in actual play, the state of such a sample position is determined at a professional level much earlier than what is shown as initial state for the problem. A single stone at the 4 - 4 point is known to allow invasions at the 3 -3 which can live; we can understand such a position to appear in a professional game only as the result of a Ko fight in which White was using the possibility of living in the corner, worth

about 12 points (+10 for Black + 2 for White.) Under ordinary conditions, such positions don't appear in professional games, rather, they serve as 'reading', or 'look-ahead' exercises for beginner players.

Several pruning techniques by one Go programmer have been seen; although all were described as within the category of 'admissibility' in that the correct result was guaranteed to be found, we see that this criteria has been stretched to perform more quickly in a certain rating level. It may be the case that no pruning is allowable, and as in Chess, at this level of analysis, only faster processing speeds will overcome the searching quantities. If the searching could be constrained to individual groups within an 8 x 8 matrix on average, this limitation might not be overwhelming.
Next we will see how Go analogues to pattern matching in chess, are used, and find some of their advantages and constraints.

III Pattern Interpretation

To avoid search and move to more rapid evaluation requires identification of meaningful information; patterns are one form of higher level information. Libraries of joseki, and fuseki are available and have been used for the opening phases of Go. [samples in Appendix.]
In general, as soon as the opponent plays moves diverging from the library move, the program must be able to adapt; like other AI applications, neither rules nor cases can provide deep knowledge, or principles of generalization.

We will look at a few pattern approaches that are applied to later phases, particularly in the midgame.

L & D Library

First, let's look at the Life and Death problems again but from the side of pattern libraries.
One way to reduce search time is to have available a library of shapes that have pre-determined solutions.

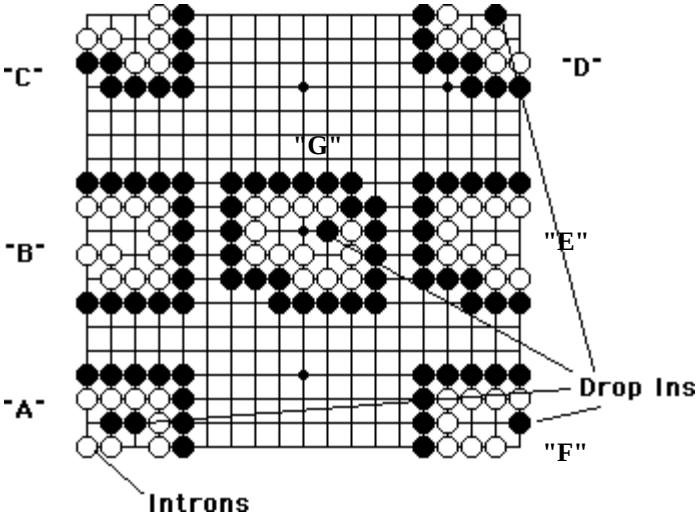
Here is a sample of patterns based around a 'bent four' shape. One thing to notice is that the L&D status varies with the pattern environment.

Absolutely alive is: B, and E.

Alive if white first: C, F, and G.

Alive in Ko: D. 15 point Ko fight.

Dead: A



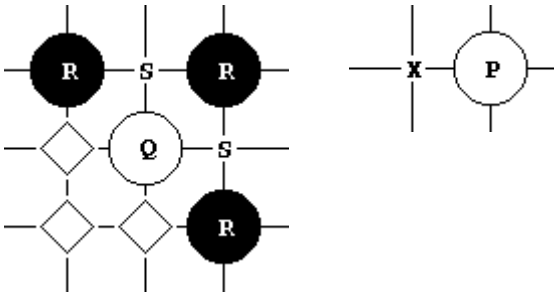
Having a pattern does not mean it is directly applicable:

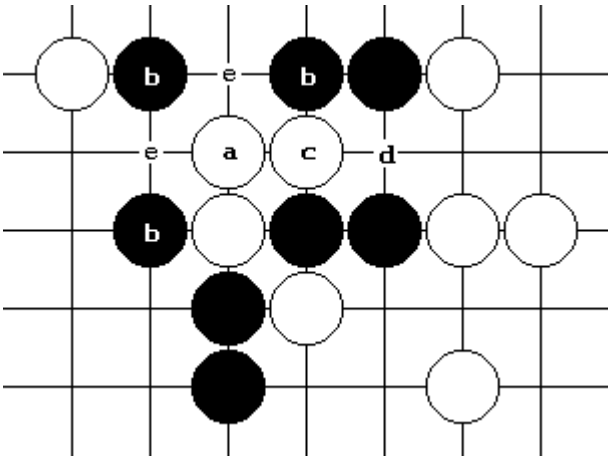
Sometimes, the stones surrounding the shape are not simply connected, as occurs at **"C"**. In these cases, the information in the shape library is not directly usable to determine life or death of the surrounding group, but can still be used under some circumstances [16] [U-line added]

Pattern Mapping Knowledge Representation

Representing knowledge is always difficult, but one approach attempts to combine predicate rules with use of pattern maps for tactical solutions to re-occurring configurations.

Pattern maps are made of size 1 x 1 to 19 x 19. Here a net pattern similar to what is shown in the appendix in Dia. 4 # 6. Such patterns are what Go players start with in order to organize the myriad patterns that do re-occur in actual games.





The maps serve as preconditions for rules which are applied to discovered patterns on the board.

They consist of factors composed out of attributes determined by the spatial relations of the stones to each other such as but not limited to:

- Occupancy (of particular points)
- territorial ownership of points (in terms of areas controlled by stones)
- liberties (empty points orthogonal adjacent to key stones)

So sets of occupancy constraints are represented in a single pattern map. Multiple maps can be related by inter-map relations such as:

- Relations: X <> S AND camelback(P,Q)
- Constraints: lib(Q) = 3 AND lib(R) > 2

In the pattern maps above, the points marked with diamonds allow either color or empty. The InterMap relation specifies that the points P and Q are in the same white block, whatever shape it has. Block Q must have 3 liberties each of the block labeled R have more than 2. After rotations, Q maps to a in the third diagram, an P maps to c. The other constraints are satisfied. So Black receives the point d as a means to capture the white stones.[17]

No search has been made to arrive at this conclusion. Every pattern in the database has however passed every point on the board as the board accumulates stones and configurations. Although the application being developed includes other components to in effect allow for planning, no mention is made of planning much less of a planning involving the type that human players use. Planning contrasts with search in its ability to avoid search, relying not on memory of patterns as such but on the principles those patterns present to the player in their totality over time. Most of all the principles represented in the abstract strategic concepts are context independent. Even if and or when a pattern is discovered that leads to a recommendation that a capture can be made as in the example net pattern above, one still has to determine whether that move is necessary, if there are more urgent points elsewhere on the board in unrelated areas, or if that capture though profitable in the local area, might nevertheless violate a more general strategic principle that would continuously lead to defeat versus beginning human players, which the present state of Go programming lacks the ability to do.

Searching or Pattern matching and then applying rules are two sides of a brittle coin.

The assumption in this paper is that rather than developing techniques that augment search methods, other methods that reach into more abstract, cognitive knowledge and provide a more general representation that can then be converted to a symbolic form for use in programmatic processing which would not only alleviate the dependence on search and pattern matching and instead provide for a more principle based and therefore encompassing mode of knowledge representation, but also provide prospective pattern generation according to strategic principles, instead of relying on retrospective oriented pattern matching post hoc.

We will next look at such an endeavor which, while not part of a functioning program, at least has demonstrated the plausibility of approaches which attempt to access a more generalizing realm.

IV Cognitive Knowledge and Perceptual Representations in Go

Context

In AI approaches to Go programming, pattern information modeled by pattern recognition processes are translated into symbolic form which are modeled by rule-based systems which allow access by symbolic reasoning processes. [18] Research in the field of cognitive psychology has provided a means to reverse the process in an attempt to embed cognitive knowledge into a numeric form that symbolic reasoning processes can access.

Traditional AI approaches view the information flow from perceptual processes to cognitive processes as a one way flow in a perceptual bottom to symbolic top manner. AI approaches to cognitive modeling usually omit the perceptual process from the model and assume the existence of a 'representational module' which will form the systems initial representation. [19]

Three problems derive from the approach:

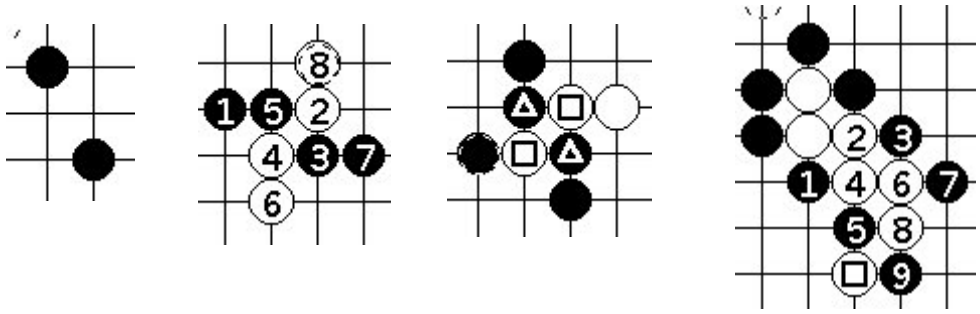
1. Deciding what information is relevant and how it should be organized.
2. Assumption of a single correct representation for objects, relations and events.
3. Prevention of a representation module for context-sensitive representation.

Go stones are identical in value, and the game goal resides in points equally distributed around the board; there is no 'focus' as in Chess-the King, or Backgammon-to remove all one's pieces from the board first. Of the three problems above, the issue of context is most easily seen as applicable. Context determines what is relevant, important, urgent, for individual stones and groups with each move of the game. One big hurdle for beginning Go players is to understand that the phases and formations provided to explain the game stages and shapes, such as joseki, fuseki, are not to be regarded as absolute, or permanent. It is only in the later middle game that final shapes need to be decided upon. Therefore, there are no absolute rules that can be formulated to direct a program or a person in the progression of moves at any point in the game.

An illustration of context dependency was provided by another research group[20] using the domain of analogical mapping between letter strings in what was called the Copycat project. If ppqrss were to be mapped to aamnxx, most natural would be to view them as pp.qr.ss / aa.mn.xx.. However, with ppqrss : aijklx, then more likely the second would be viewed as a.ijkl.x and the first p.pqrs.s. The view changes with the string to which mapping takes place.
In the Copycat project, perception was categorized as low and high. Low-level perception filtered and formed representations from environmentally derived sensory information. High-level perception extracted meaning from the low-level representations which is then used in performing high-level cognitive tasks. In this way high-level perception may represent an interface between low-level perception and cognition. Integrating perceptual and cognitive knowledge into a theory would require further research into the nature and utility of cognitive knowledge.[21]

Ladders

Nevertheless, the Copycat model provided the impetus to understand and provide the model for an aspect of Go tactical processing. In Go, stones are considered linked when each stone is orthogonal contiguous to another. However, there are many virtual links. One example is the kogeima (knight's move):



- The first shows two black stones which are not contiguous but are considered connected.
- In the second image, 1 & 3 form the knight relationship, which is 'cut' by white. Now 1 & 5 and 3 & 7 are two separate groups. Each group must live independently, so four eyes must be formed, whereas if black had a stone at 2 or 4, all would be connected and only two eyes would be required for the single group of five stones.
- In the third image, where the Knight's virtual link exists before White moves in the area, the two marked White stones are cut, and Black has put the single stone in 'atari', and also a 'ladder' has taken shape. [Appendix shows ladder capture with drive to side.] The single White stone has been caught in a ladder, but, assuming there is a ladder breaker existing in the ladder path, the White stone eludes capture, and the two marked Black stones remain cut.
- The fourth image shows a White ladder breaker; it is marked with a square. After Black 9, White can capture 5 and escape. If Black connects with 5, White now has time to 'atari' above 7, then below 7 and extend, thereby escaping. [or atari below1, or above 3 ..]

This explanation illustrates the physical aspect of the virtual link; an abstract example would be the opponents goals, plans, in the short term and longer. Due various conditions, the link may not be considered worth attacking; for the time being it can be considered as if it were three stones in a row, surrounding or displacing an area, the total of which may be used in determining overall strategies.

Thus, as with the Copycat analogical string mapping example, the way in which stones, groups and links are perceived or evaluated, depend upon contextual factors which can be determined in a top-down, cognitive to perceptual manner.[22]

Influence Functions

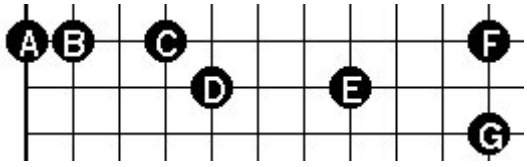
In the evaluation of board positions, many forms of heuristics are employed. One group of evaluation functions use Influence Functions. These are heuristics that depend on patterns of numeric values that result from the presence of a stone on a point, and which radiate outward from the stone, or group of stones, according to notions of projective power which is based upon the ability of the stone or string of stones to exert control over some part of the board, near or far.

Arithmetically, one color is assumed to radiate positive values, the other, negative. Summations are done for each point to give values to those points. Otherwise, influence functions may operate in various ways.

- Chen (1989) Influence is radiated onto unoccupied points
- Zobrist (1969), Ryder (1971) Radiate influence both to occupied and unoccupied points. [23]

Connectivity

Connectivity describes in numerical terms the likelihood that two stones are able to be connected to form orthogonal links, or capture interlopers threatening to cut and connect to



friendly stones on the other side.

1. AB: Connected, unless captured later on.
2. BC: Dependent upon presence of White stones in area In opening, or beginning of game then, this would be connected; later, as more white stones appear, connectivity diminishes.
3. CD: Connected, as long as move above D can be responded to below C, or reverse.
4. DE: Like 2, but would have less connectivity.
5. EF: Great Knight, similar to Knight; i.e. ladder dependent.
6. EFG: Generally either EF or EG can be connected, but ladders, proximity of side, and other White stones in area all affect connectivity.

One heuristic is to use influence functions to determine connectivity. Likelihood of connectivity is determined by higher values, and the reverse by lower. A "threshold of connectivity" can be set by the programmer to classify virtual links as secure or insecure.

Computational Problems Using Influence in Determination of Connectivity

Go programs are not presently able to determine connectivity successfully. Influence functions are by design local, and do not represent global factors affecting connectivity in particular areas. There is no symbolic or high level knowledge represented in the numerical value at a point. Connectivity does provide an example of the interface between perceptual and cognitive issues in Go. [24] Ladders are solved by Go programs with special algorithms used to determine the outcome of a ladder, *once it has been recognized* (Fotland, 1993) [Italics added]
Before recognizing a potential ladder breaker, any strategic decisions made concerning the connectivity of stones related to a ladder will be adversely affected. Humans are easily able to incorporate the effect of ladder breakers on tactical and strategic implications.

Ladder Breaker and Influence Function Simulations of Cognitive Perceptual Integration

In the simulations performed by the researchers, the influence map represented perceptual representation. Cognitive knowledge is represented by the knight's virtual link and factors beyond the local region of the link. The content of the cognitive knowledge is the existence of the ladder breaker and its effect on the *connectivity* of the link.[25] [Italic added]
Simulations realize the integration of cognitive knowledge into perceptual representation by the *modification of the numeric information* in the perceptual representation.[26] [Italics added] The influence function itself was based on previous usage (Chen 1989) and was

$$influence(st, pt) = m * f^{(d(st, pt) - 1)}$$

st = stone, pt = point
m = maximum value,
f = decay factor
d = distance between unoccupied points
influence < 1 was ignored

Varying the values in four simulations allowed the results to changed with regard to a threshold of connectivity set to be at a certain level of 30, which can be compared to the value in the figures.

The first figure [next page] shows the initial condition. A Knight's link exists between point (G,3) and (H, 4) .

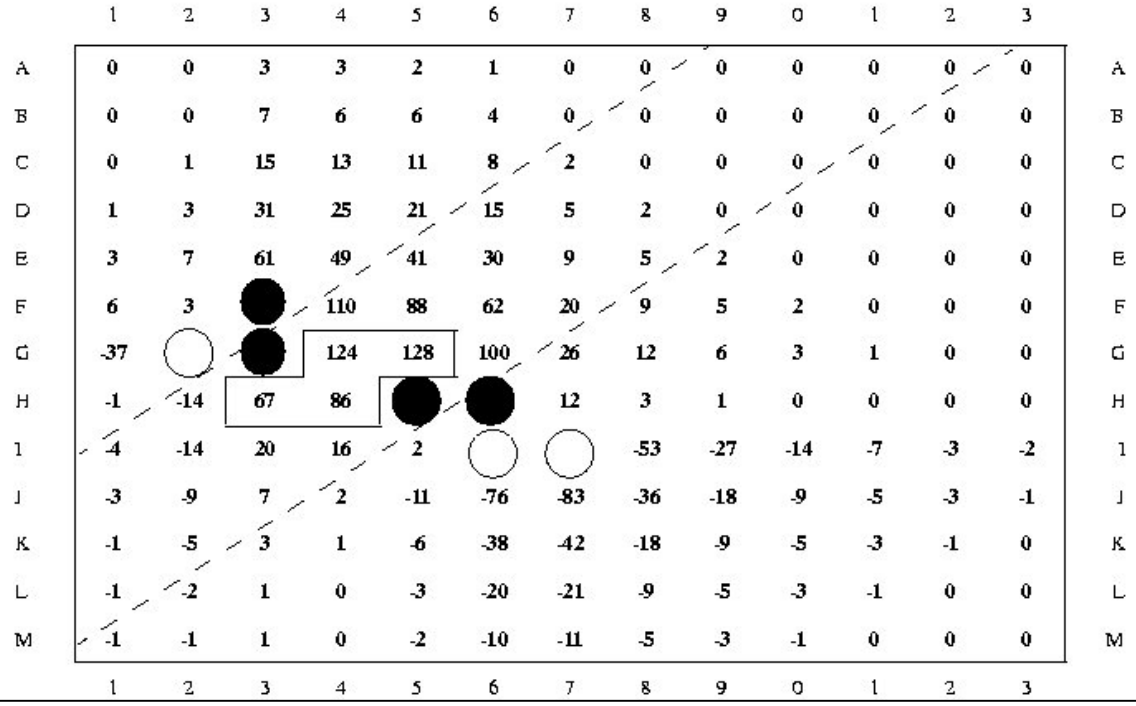
We will not enumerate the specific simulations but rather look at the main result.
In one variation, context-sensitivity was simulated by multiplying selected values described by ladder containment lines (lines within which a ladder breaker would take effect shown as dotted in the first figure) by a factor depending on whether they were positive or negative.

The second figure shows the result of one of twelve maps generated for variations of influence values. As the ladder breaker was moved further away, its influenced diminished within the connection points accordingly. At the most remote point, the influence **only extended as far as the greyed line;** it did not reach the connection points at all. However, **after** the context sensitizing with the modification of the influence values as described, the values were lowered below the connectivity threshold of 30. [Bold Added]

Functionally, this was achieved by multiplying Black values by 1.2 , and White values by -0.2.[27]**.** The ladder breaker stone reduced values that it influenced by 20%.

Conclusion to Cognitive - Perceptual Integration Simulation

To represent human thinking, methods need to be found that can incorporate high level cognitive understanding of domain experts into heuristic functions. The conventional approach in terms of connectivity in Go games with respect to ladder breakers has been to utilize algorithms that determine the effect of any ladder breakers after a ladder has begun. As such, perceptual knowledge governs the strategic realm, rather than the reverse: the program perceives the condition when a White stone attempts to cut the Knight's link, and a function does a ladder



	1	2	3	4	5	6	7	8	9	0	1	2	3	
A	0	0	3	3	2	1	0	0	0	0	0	0	0	A
B	0	0	7	6	6	4	0	0	0	0	0	0	0	B
C	0	1	15	13	11	8	2	0	0	0	0	0	0	C
D	1	3	31	25	21	15	5	2	0	0	0	0	0	D
E	3	7	61	49	41	30	9	5	2	0	0	0	0	E
F	6	3	●	110	88	62	20	9	5	2	0	0	0	F
G	-37	○	●	124	128	100	26	12	6	3	1	0	0	G
H	-1	-14	67	86	●	●	12	3	1	0	0	0	0	H
I	-4	-14	20	16	2	○	○	-53	-27	-14	-7	-3	-2	I
J	-3	-9	7	2	-11	-76	-83	-36	-18	-9	-5	-3	-1	J
K	-1	-5	3	1	-6	-38	-42	-18	-9	-5	-3	-1	0	K
L	-1	-2	1	0	-3	-20	-21	-9	-5	-3	-1	0	0	L
M	-1	-1	1	0	-2	-10	-11	-5	-3	-1	0	0	0	M
	1	2	3	4	5	6	7	8	9	0	1	2	3	

Above: Kogeima link at (G3, H5) Ladder breaker area considered as 4 lines within dashed line.
Below: Influence before contextual adjustment limited to gray line area; after adjustment, values reduced below connectivity threshold of 30 up through link area.
Caveat: Cutting stone at G4 can be captured in nets, or ladder shifted up with atari at G5. Also, a ladder-breaker path is 6 lines wide, not only 4.

	1	2	3	4	5	6	7	8	9	0	1	2	3	
A	0	0	3	2	0	-3	-8	-16	-32	-64	-32	-16	-8	A
B	0	0	6	4	2	-4	-16	-32	-77	○	-64	-32	-16	B
C	0	1	15	12	9	4	-6	-19	-38	-64	-32	-16	-8	C
D	1	3	31	25	20	13	0	-7	-16	-32	-16	-8	-4	D
E	3	7	61	49	41	6	1	1	-6	-16	-8	-4	-2	E
F	6	3	●	110	18	12	19	7	1	-6	-4	-2	-1	F
G	-37	○	●	25	26	100	26	11	4	-1	-1	-1	0	G
H	-1	-14	13	17	●	●	12	3	0	-2	-1	0	0	H
I	-4	-14	20	16	2	○	○	-53	-27	-15	-7	-3	-2	I
J	-3	-9	7	2	-11	-76	-83	-36	-18	-9	-5	-3	-1	J
K	-1	-5	3	1	-6	-38	-42	-18	-9	-5	-3	-1	0	K
L	-1	-2	1	0	-3	-20	-21	-9	-5	-3	-1	0	0	L
M	-1	-1	1	0	-2	-10	-11	-5	-3	-1	0	0	0	M
	1	2	3	4	5	6	7	8	9	0	1	2	3	

check and finds the ladder breaker. This corresponds to the difference between low level human understanding in a beginner who has learned the operational aspect of the rules, and more experienced knowledge where strategic knowledge governs the tactical choices. The ability to evaluate and understand positional development provides the human with an effective pruning, or discrimination heuristic. With the influence map adjusted for context sensitive, the ladder breaker's effect became available to the connectivity evaluation of the Knight's link area, without search or pattern matching. This simulates human perceptual ability without search and would provide access to strategic manipulation.

The difficulty remains in the translation of cognitive semantics from the domain expert to programmatic language, or form. The connectivity domain in Go presents the possibility of tractable efforts in this regard. In contrast, the authors indicate that while any translation; cognitive to perceptual or the reverse, will lose information in the translation process due the brain's requirement that only salient features be extracted, information that should be extracted can not be determined prior to its actual use. This insight was the key motivator the Copycat model and was illustrated in the letter-string puzzles. That is ..

> Attempting to translate perceptual information into a context independent cognitive form is putting the cart before the horse.[28]

Ladders can cover large areas of the board; a ladder starting in one corner can run diagonally across the board, covering 40- 50 or more ply easily. Also, circumstances can occur where the ladder 'ricochets' off friendly stones and changes direction, covering even more area. Therefore, the resulting combinatorial explosion involved prevents conventional encoding in a bottom-up perceptual process in order to look for potential ladders in all the right places.

V Spatial Representation

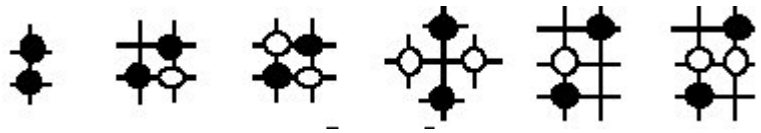
Go is an activity that is understood by the players in terms of abstract concepts and also pattern and search. In general, all of these may be subsumed under a category understood as spatial reasoning. Spatial reasoning is a type of comprehension that appears prominently in visual arts such as painting, drawing, calligraphy, spatial gesture, and dance. If natural language is

understood to represent a bias towards analytical thought, then spatial reasoning can be described as similarly organized within a synthesizing, holistic, comprehensive mode of thinking. The eight strategic concepts mentioned at the beginning of the paper, such as Thickness, Ambivalence, etc., are terms that represent the verbal interpretation of spatial reasoning; as such, they are usually illustrated by examples from Go games: indeed, books are written specializing in one term and they devote many chapters to the illustration of how that term may manifest, evolve and relate to various game phases, or how particular situations exemplify the term, or how such terms can be used to understand the generation of the themes that games represent, or how individual professional players utilize the terms in their own way: their 'style'.

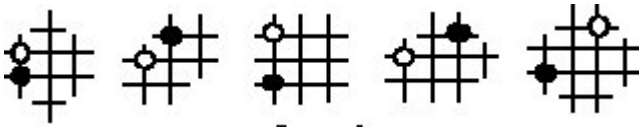
Here the concept of spatial representation in general will be presented using material that was used in the construction of a Go program that enjoyed some success at one time, competing favorably against conventional rule based systems without a spatial representation component.

Essentially, the method start with pattern mapping of strings on the board, and then uses a mathematical formalism to model the cognitive representation of humans. Set theory and notation is used to represent the formation of groups and relations between the groups.

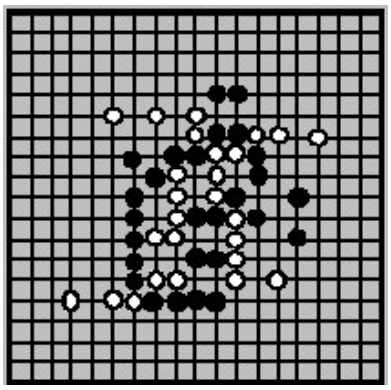
First a graphic set of stones represent both connectors and dividers:



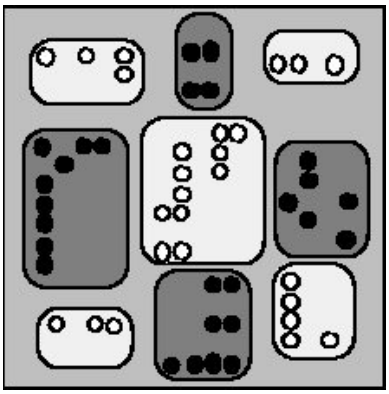
Then, a set of strings are used to represent elementary contacts:



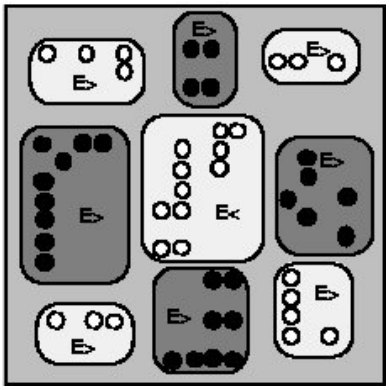
These two pattern sets are used in an iterative fashion to build more complex relations. [29]



Dia. 1



Dia. 2



Dia. 3

Initially, a start position is given on which the notation will work, as shown in Diagram 1. Then, iterations are performed which take the elementary connectors and dividers, matching them to stones of the same color which are in the same area, and stopping the iteration when no more stones of the same color appear. Dia. 2 shows group that have been formed in this way.

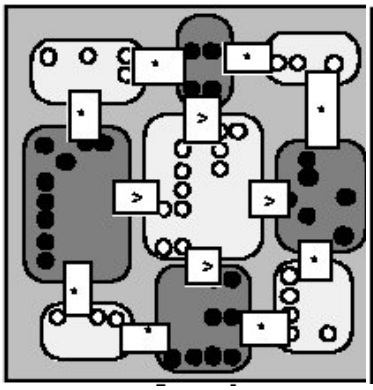
Fractions are defined as a set of intersections that do not belong to a divider. Using fractions, opponent groups are found. Opponentship O(x) determines which groups are being encircled by which. Using distance formulas and preset thresholds to limit how close an opponent group is to another before it is to be defined as 'encircling' the other, iterations are performed over the whole board to determine group encircling, or Emptyship. Dia. 3 illustrates the results of this process. E> indicates Emptyship is Positive, E < , Negative and E = would indicate a Fuzzy evaluation. Positive means that the group is not being encircled, whereas E< indicate that the group is being encircled, as shown on the third Dia. 3. This does conform with Go sense, as groups with space on the side are more easily able to form eye shape, preventing capture, than those in the center.

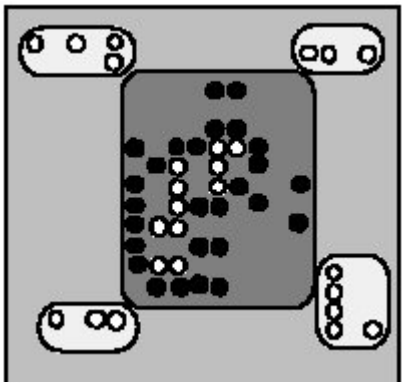
Relative strengths of two groups x and y can also be determined using the spatial representation process. R(x,y) is found by using rules as shown:

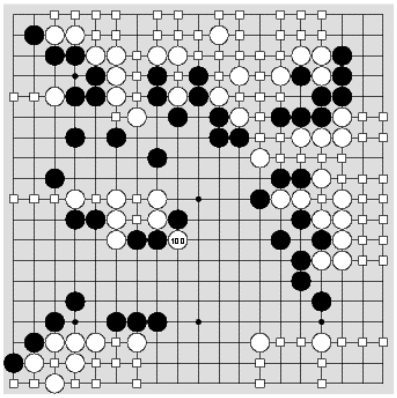
$(\exists \alpha \in \{x,y\}: E(\alpha) = \text{Fuzzy}) \Rightarrow (R(x,y) = \text{Fuzzy})$
$(E(x) = \text{Negative} < E(y) = \text{Positive}) \Rightarrow ((R(x,y) = \text{Negative}) \wedge (R(y,x) = \text{Positive}))$
$(E(x) = E(y) = \text{Positive}) \Rightarrow (R(x,y) = \text{Fuzzy})$
$(E(x) = E(y) = \text{Negative}) \Rightarrow (R(x,y) = \text{Fuzzy})$

If the Emptyship value of (x) is negative, it is weaker than a Positive Emptyship value of (y); else it appears Fuzzy, or undetermined.

Undeterminedness is a value used by Go players intuitively when evaluating positions that have not been fully 'settled', or 'played out'. [30]







Dia. 4 Dia. 5 Dia. 6

Adjacent group relationships are shown in Dia. 4. The Opponentship of groups depends on the R(x,y) strength relation found above. Fuzzy relations yield fuzzy Opponentship, as do Positive and Negative relations. As before, > = Positive, < = Negative, and * = Fuzzy.

When a group is unable to make eye shape, it is described as 'dead'. This is represented by a Negative Opponentship. When a group of stones encircling another 'links up', the stone strings are 'fused'. This is represented by a Fusion operator that permits the new grouping that merges into one around another. This is shown occurring in Dia. 5. [31]

Summary of Spatial Representation

This program (Indigo) attempts to model groups of stones in terms of spatial relations in a formalism derived from set theory. This contrasts with other programs that would start from a perspective of linkage. Dia. 6 shows a conventional link point map of white groups. These maps are related by issues relating to connectivity which must be determined by searches that attempt to cut lines roughly described by a 'connect the dots' in straight lines or of contiguous stones. To the extent that the iterations relating elementary pattern maps avoid searching as such, the spatial representation seems to provide a faster way to aggregate large numbers of stones into groups in much the same way as players do intuitively, as well as providing ways to represent relative strengths of groups without searching for life and death sequences. Still, L&D sequences must always have an element of search; we can say that the spatial representation

methods allow the program to select weak groups for strengthening, or attacking potential, prioritizing what otherwise would have to be determined in a methodically but more extensive and otherwise random process. Identifying weak from strong groups is in fact one of the first 'senses' developed by a beginner Go player progressing into intermediate levels. In some sense, all Go players develop this sense of group strength identification proportionate to the level of their rank: within a particular rank, this sense serves as a selection procedure that permits the player to change the focus of the board evaluation and allows the development of plans for the game progression. With increasing ranks, the player has a more precise 'impression' of what potential development, tactically speaking, is available in a particular position, or what types of weaknesses are present in terms of L&D. For example, there are many types of Ko's: 1 step approach move, 2 step approach moves, double ko, ko leading to a seki, and others. A beginner may not be able to 'see' the weakness of an L&D position to such an extent, just that the group is weak and needs re-enforcement dependent upon the numerical value of the position; as soon as that value is approached, or there is nothing of greater value left on the board, the reinforcement is made. A stronger player might see that one or more moves could be made within the weak group by the opponent before affecting the status as a Ko, Seki, or liveliness.

Spatial representation is another form of cognitive representation being embedded into the program, as with the influence map experiment which provided context sensitization. In addition, the Indigo program incorporated a semantic network generator. Explanation facilities are thereby possible, allowing the program to present explanations for decisions made. [32] Natural language incorporation reflects another level of abstractive material in Go programming; for more than heuristics themselves, natural language explanation is an important tool in ascertaining the processes that the program uses in the decision process. Bouzy writes that according to one source, [Herskovits 1986], "spatial prepositions are crucial to spatial discourse understanding even if it is difficult to model them." Just as the rule of Go are more easily encoded than Chess, or other games, so too are the prepositions used in Go: 'up', 'down', 'left', 'right', 'middle', 'edge', 'corner', 'in', 'touch', 'circled'. More research is forthcoming on this subject by the author and programmer.

VI Planning in Abstract Domains

Mention has been made in this paper about the relation between bottom-up search and top-down abstract cognitive directed conceptually informed strategizing. In the Perceptual Integration section a method was presented that was able to simulate the unification of these two polar opposites in the modified influence map. However, as with the pattern recognition approaches, planning itself was not the focus. Here we shall see a planning methodology which has achieved the ability to incorporate both antipodes in one technique which is more intuitively consonant with human playing modality. It is a modification of a Hierarchical Task Network for Adversarial Planning.

In most adversarial domains such as game domains, rules drive possible action generation, such as state space search, and then evaluate the results, choosing the best outcome. Another approach has been to use a goal driven approach: Abstract goals are decomposed into action specific primitives. Sample abstract goals would be kill-group, satisfiable by an abstract plan containing subgoals such as: surround-group, reduce-space, and prevent-eye-shape.[33] Advantages are twofold:

1. The need for full enumeration of all possible actions in each state is removed. It is therefore the goals that determine the size of the search space, not simply the options presented by the number of possible leaf nodes.
2. Evaluation function complexity reduction. In state base searching, the data encountered must be evaluated according to a plenitude of criteria. Goal driven approach focuses on whether a low level primitive action satisfies the goal on the abstract level. [34]

In Go especially, the almost transfinite quantity of tree space complexity at 200^{250} , and more since a good proportion of professional Go have 280 moves in a game, not only 200, rules out search as has been indicated earlier. Reasoning above the level of individual actions allows particular actions to be chosen only if the systems finds the action to be in accord with the satisfaction of a goal; i.e., in the *context* of a plan of action. Goals at the lowest level of

abstraction are usually simple, so checking for goal conformity is easier than judging the value of an entire state. [35]

However, little recent work has been done on HTN Adversarial Planning as it had two major disadvantages:

1. Complexity: HTN-AP is inherently more complex than standard planning. The linear dimension of state searching is replaced by a tree of contingencies for all possible orderings of the opponent**.** The work load is difficult to manage.
2. Modeling abstract knowledge. This too is often difficult to represent the abstract knowledge in the domain well enough to be of use. [36]

The modifications made to the standard HTN-AP format include:

- Two agents interacting are modeled.
- Use of linearisations during goal decompositions
- Decomposition of some goals to the lowest level of abstraction as soon as possible
- World model can be used to add data driven aspects to the system

Note: Partial ordering involves setting sequence order for some steps, not others.
Total ordering involves listing the steps.
Linearization involves adding ordering constraints to a plan. [37]

Standard HTN Goal Decomposition

At each step, the standard scheme will:

1. A goal is chosen for decomposition
2. A promising decomposition schema is chosen
3. Substitute chosen schema in place of decomposed goal
4. Return to step 1. [38]

Problems with applying Standard HTN Goal Decomposition in Adversarial Domains

Standard HTN presents two main obstacles when applied to adversarial domains.

- As the solution plan moves between different levels of abstraction, the solution plan has to be kept in track, particularly to avoid redundancy in search.
- Successful planning using ATN depends on the abstract knowledge of the interactions between goals at each level. As knowledge decreases, the amount of wasted work increases.[39]

The first of the two difficulties is an essential feature of the domain and cannot be avoided. Search for prospective sequences, in L&D or other phases requires the 'reading' out of possible sequence sets, by humans as well as by machines.

The second kind can be avoided to some extent by ensuring that enough information is encoded in the abstract operator to effectively reason about the domain and about possible goal interactions. [40] It is difficult to represent or foresee such goal interactions. Two goals that may seem unrelated may interact at the lowest level in the plan. In chess, X: Keep knight pinned, and Y: Queen a pawn may interact if the pawn must advance between the pinning piece and the pinned knight in order to get to the board edge. [41]

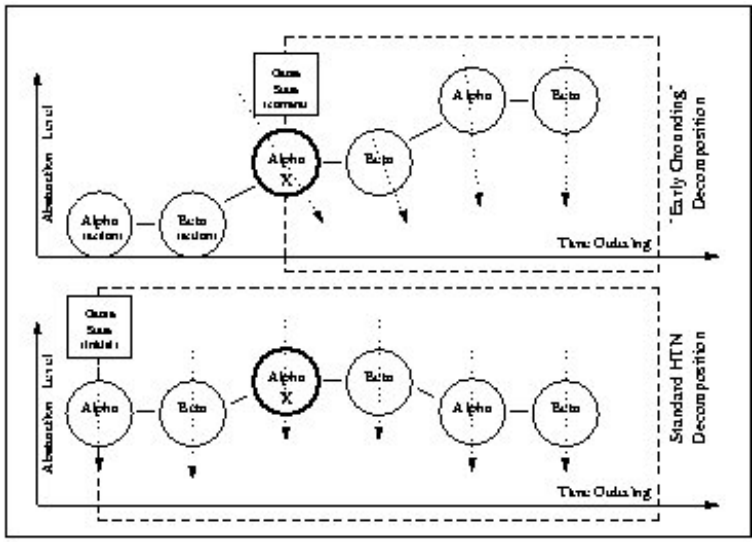
Early Grounding HTN

The approach used by the team was to apply an "Early Grounding HTN". The basic idea is to make goals as concrete as possible. The scheme uses the following steps:

1. Choose a goal to decompose.
2. *Mark this goal to be next in the order of actions.*
3. Select a promising decomposition schema (taking into account the current world model).
4. Substitute this schema into the plan i place of the decomposed goal.
5. *Choose one of the new goals added into the plan by the decomposition.*
6. : If this goal is primitive Then perform action in world model and return to step 1.
   : Else decompose this goal further by returning to step 2. [42]

Steps 2 and 5 liveries the plan by selecting a single goal to decompose first, making this a total order planner. Step 6 then forces all primitive actions to be 'acted out' in the world model. Each goal can then be decomposed in the context of its final place in the plan. [43]

In the figure, the Early Grounding and Standard HTN schemes are represented. The difference between them is that with the Standard HTN, (lower diagram ,) the planner reasons about all goals at their current level and about the world in its *initial* state. The EG HTN reasons about the outstanding goals and about the *current* state of the world. [44]



In the Chess example, there were two goals, X: Keep knight pinned and Y: Queen a pawn.
In the standard plan, both goals would independently be decomposed towards the concrete level (along with any other goals) and despite all searching and analysis, each result would fail when the interaction between the two goals X and Y of the pawn intercepting the pinner and pinnee on the way to the edge became explicit. [45]

However, in the Early Grounding version, one goal (Y) is first decomposed leading to one or more moves in the world model. Then another (X) is decomposed in the context of this new, updated model. (X) reaches the lowest level of decomposition and finds the moving pawn interfering with the pin. The world model immediately shows that these two plans interact and may cause the whole plan to fail. [46]

Therefore, EG ATN allows the system to make choices for every decomposition step which are "feasible" with respect to all previously made decompositions. Some of the dependence on

having accurate abstract representations of all important interactions is hereby removed. So EG ATN allows for planning using incomplete plan operators. [47]

This system along with some additional components was tested on Go L&D problems and performed well. Knowledge was limited to plans involving the attacking and defending of groups and strings on the boar. Plans were held in the system at five levels of abstraction, the lowest level being moves on the board and the highest being goals such as kill-this-group or save-this-string. [48]

Conclusion to EG ATN

This approach combined the ability to control tactical choice by abstract goal representation with a methodology that simultaneously prunes the searching space. This in general is how humans choose their plans; according to higher level thinking which summarizes the 'situation' as it currently exists on the board, allowing 'reading out', the human equivalent of search, to be concentrated in areas that relate to the strategy at hand.

This model was only applied to low rank beginner L&D problems. Therefore, the levels of abstraction did not involve the kinds of globalizing abstraction applied to the whole board as people do use. In other words, more than five levels are required: for example: take the concept of 'thickness', which relates to influence, connectivity and virtual linkage between stones and groups, as was explained in the Cognitive Integration section above. This idea applies to individual stones and groups.

In a real game, the thickness of stones, groups and positions is one of many other strategic concepts used to understand the current situation. Thickness evaluation of these categories is applied and re-evaluated throughout the game for the same material. Based on these ongoing and changing evaluations, tactical sequences are devised, and implemented. The levels of abstraction therefore include not only top down levels for one key stone or group, but lateral comparisons as well. In a sense, the eight basic strategic concepts are applied to any and all stones, groups, sequences and shapes at the same time. So there is a large interpenetration of

conceptual manipulation, to varying degrees conscious to the player depending on that player's rank, between concepts as well as the practical decomposition into sequence development.

Still, the employment of the EG ATN model represents a key point of progress in the Strategic control by higher levels of abstract goals over tactical choices.

VII Epistemological levels of Abstraction

> " .. Space and time contain a manifold of pure a priori intuition, but at the same time are conditions of the receptivity for our mind-conditions under which alone it can receive representations of objects .. but if this manifold is to be known, the spontaneity of our thought requires that it be gone through in a certain way, taken up, and connected. This act I name synthesis. By synthesis, .. I understand the act of putting different representations together, and of grasping what is manifold in them in one [act of] knowledge. Such a synthesis is pure, if the manifold is not empirical but is given a priori, as is the manifold in space and time. Before we can analyze our representations, the representations must themselves be given, and therefore as regards content no concepts can first arise by way of analysis. Synthesis of a manifold .. given empirically or a priori .. is what first gives rise to knowledge. .. "
>
> **C**ritique of **P**ure **R**eason, I. Kant, Macmillan, 1965., p.111-2.

In all description of concepts, the epistemological tradition has a tendency to be omitted in the computing domain: only recently in regard to Go and AI specifically has the realm of cognitive analysis of perceptual relations to encoding been accomplished in varying methodologies at a basic, if fundamental level by various researchers, and computer programmers. Still, when we describe the need for generalization in the context of planning in AI, the epistemological language, particularly that of Kant, can be helpful in providing a standard against which progress in abstraction of knowledge representation can be measured: For Philosophy has provided the most general forms with which knowledge can be represented and understood, and to the extent that the discipline of AI attempts to develop programs capable of performing in ways similar to human thinking processes, ought Philosophy, and in particular Kant, to be disregarded?

Many who play Go find it to be able to represent many aspects of fields of knowledge. For example, relations between Go and Economics, Social Sciences, War simulation, (guerilla mode) Linguistic, Biology and Earth Sciences have been presented. [49.] Also, practitioners of Chemistry and Architecture note similarities in Go to their professions. Go is general in its appearance, and lends itself thereby to such analogizing. But the main cause is that it is the working of the mind that itself allows this: it is the common denominator behind all such

observations. The main source of the ability of the mind to generalize Go to all these other areas is that which allows generalization to occur at all on the epistemological level: the Categories.

The categories are that which provides synthesis for the objects of intuition, that through which we perceive the world. The categories themselves go back to Aristotle, but Kant used them to account for the possibility of knowledge, as well as the necessity for the categories to form the basis of science.

" ... [the categories] contain the list of all original pure concepts of synthesis that the understanding contains with itself a priori. Indeed, it is because it contains these concepts that it is called pure understanding; for by them alone can it understand anything in the manifold of intuition, that is**, think** an object of intuition."

[CPR., p. 113-4., (Bold by writer)]

I	**II**
Of Quantity	**Of Quality**
Unity	Reality
Plurality	Negation
Totality	Limitation
III	**IV**
Of Relation	**Of Modality**
Inherence & Subsistence	Possibility - Impossibility
Causality - Dependence	Existence - Non-existence
Reciprocity	Necessity - Contingency

These categories are the representation of the source for the linkages between things in general, generalization as such, and Go and things in general in particular. Go ideally maps these categories to its operation in the following ways:

- **Quantity**
  - **Unity** - In unity we see the availability of comprehending diverse operational and strategic concepts applied amongst and between units and groups.
  - **Plurality** - Plurality allows for the separation of multiple group and unit interaction as they intermingle during development of sequence.
  - **Totality** - Totality provides for the result of diverse objectives to be summed for conclusion.

å Quality

- **Reality** - That a prospective plan can emerge from application of Force is given by the concept of Reality.
- **Negation** - The removal, physically and conceptually, of units and the plans conceived for their dispersal is here delimited.
- **Limitation** - The potential of Negation upon Reality appears in the circumscription of the protagonist when considering the possibilities for the opponent.

å Relation

- **Inherence** - Within any group or unit, a source of self or identity permeates relations with others.
- **Causality** - Throughout the continuum of event subsistence, each unit and group contributes to Force.
- **Reciprocity** - Given Inherence and Causality, that both units and groups are the locus of Force, both responding to and causing the response of particularity, or choice.

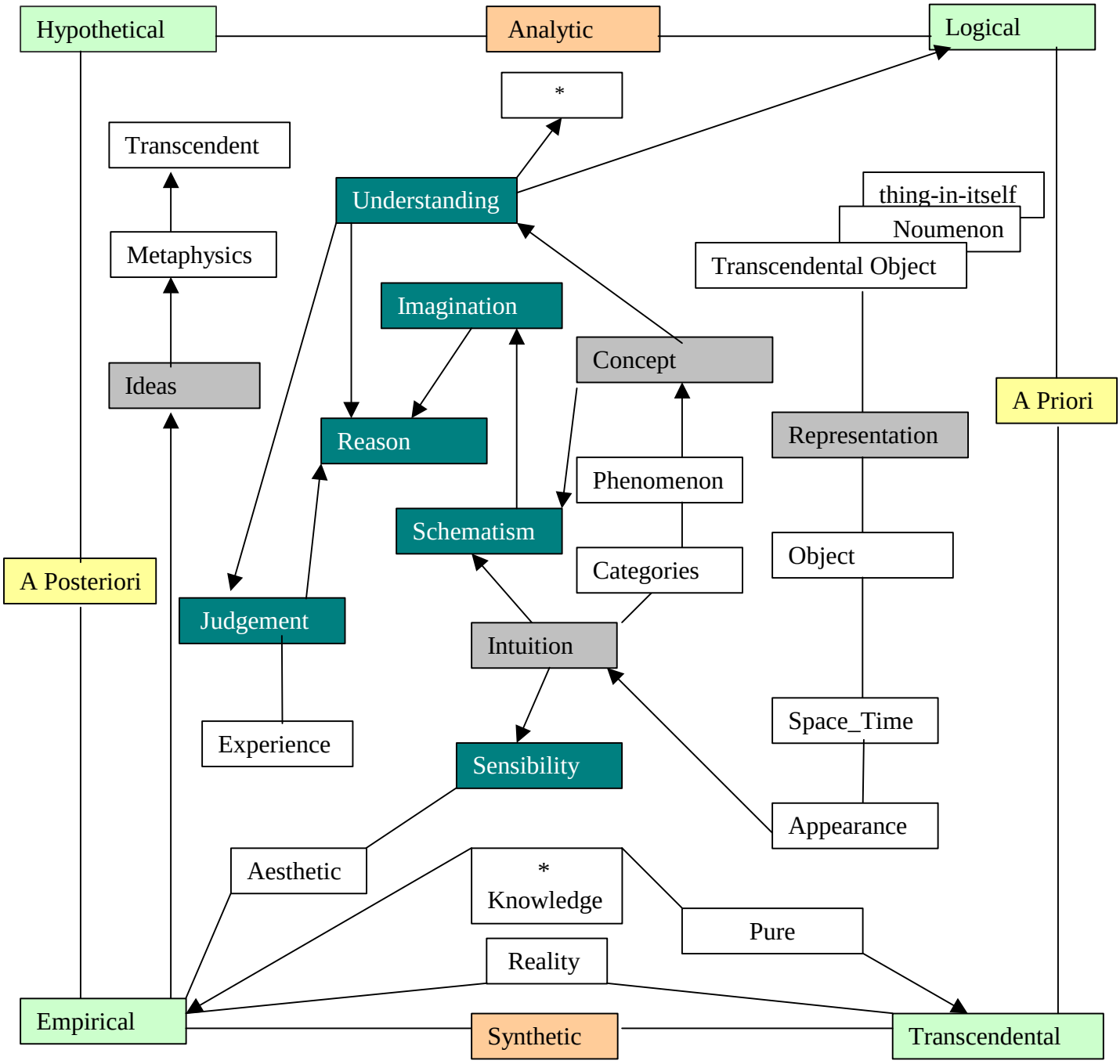
å Modality

- **Possibility** - What is possible for conditions at a point in time, regulating what is referred to as "timing".
- **Existence** - This is itself a source of contention, as each knows only within the limits of individual "reading".
- **Necessity** - Simple synthesis of the two preceding, later expressed as non-being + being --> becoming. The extraction of Existence from Possibility, and its deposition in the unit. [50]

Now, the interpretation of the categories in Go terminology may need further explanation and improvement to be understood by or used in Go programming; nevertheless, it does indicate a level of abstraction above that used in the main strategic concepts of Go as mentioned at the beginning of the paper: 'Thickness', 'Force', 'Potential', 'Probe', 'Lightness', 'Heavyness', 'Exchange' and 'Ambivalence' (for 'miai' ,) as these again represent a layer of abstraction above goal specific abstractions such as black-to-kill.

Just as the cognitive sciences allow the generation of programming approaches based on insights derived from psychological studies, so too we can assert that insights from the philosophical domain may allow new strategies to be used in knowledge representation for AI applications.

The categories are but one element in Kant's world view, here presented in the figure on the next page. This term map only serves to indicate one philosophical interpretation of knowledge by Kant but is certainly a developed one, both in terms of the original author and commentary made in the centuries since.



Main Critical perspectives about Knowledge, in interpreting Reality and Experience	
Way to gain knowledge with-a posteriori, or without-a priori, Experience	
Representation types of object determination	Faculty: Functions of Rationality
Truth based on Intuition-Synth., or Logical law-Analysis	Terms related to Kantian thinking processes

Neural Nets toward an A Priori Event Horizon

One approach to broach the a priori side of computing has been to incorporate basic rules into a neural net, forming an 'a priori' set of encoded principle generator potential and let the NN teach itself with training routines. Instead of using conventional AI techniques such as pattern matching, rule based systems and goal oriented selective search, which the author points out as presenting difficulties of management due the complexities incurred by the manifold requirements of Go, the author devised various methods to integrate expert Go knowledge into a learning artificial neural network. [51] An implementation of these methods was able to achieve parity with conventional Go program playing in the Kyu range, on a 9 * 9 size board.

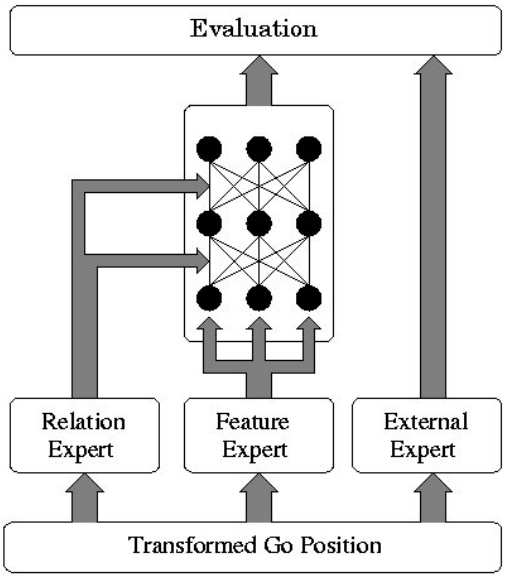
Conventional programs using the usual methods become increasingly unmanageable with size, and also incur problematic side effects of newly integrated knowledge and unforeseen rule interaction. [52] Other programmers had shown that neural networks could learn to beat conventional programs if translational and color symmetries of the pattern recognition task were carefully reflected in the network architecture, despite not having integrated expert knowledge into the system. [53]

System Architecture

The system contains a neural network with one hidden layer. Three expert modules integrate a priori knowledge.

The system is given a board position and produces an evaluation for every intersection on the board. The probability that an intersection will belong to a color is the interpretation of the evaluation. First, the board position is transformed into sets of strings and empty intersections, determined by the safety of a group; if safe, then the color will remain at the game's end; else, it will most likely be the territory of the opponent. Then a priori knowledge is integrated by three expert modules consisting of a Relation, Feature, and External Expert. (See Figure next page.).

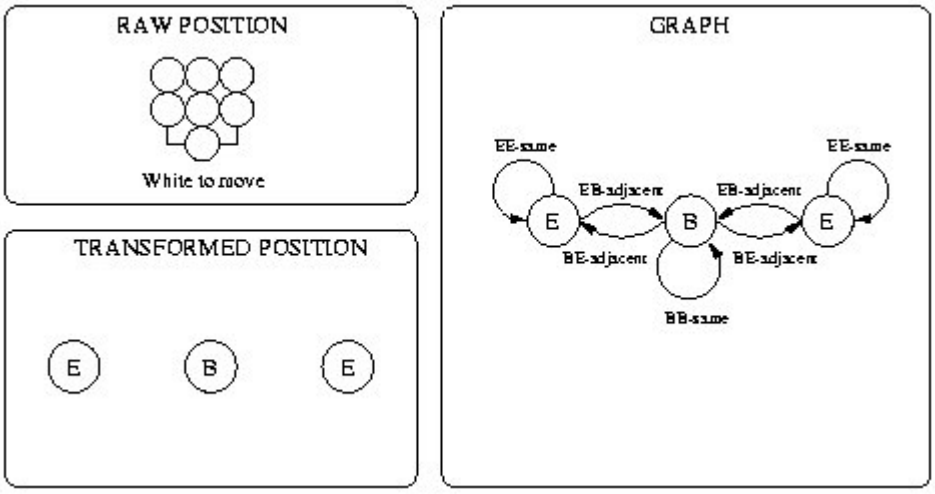
Both strings and empty intersections are referred to as units.



Relation Expert: A graph to determine the connectivity of the network is constructed using units as nodes and the relations as edges. The expert decides which of two relations is the most important if they both appear in the same direction between units.

Two neurons of adjacent layers are considered connected only if the units are related. Weights used for the connection depend on their relation and unit type. The location of the units on the board do not influence the weights. [54]

This method incorporates not only translational and rotational invariance but also Go specific invariance's such as: "Two eyed strings cannot be captured". This knowledge can be easily learned according to the author regardless of the size and location of the string, as long as important relations are specified.



The figure shows an example of the network using the transformed Go position and the Relation expert. A position is given on a 3 * 3 board. It is transformed into B = Black and E = Empty

intersections. The Graph section of the diagram shows that the expert only recognizes "adjacent" and "same". [55]

Feature Expert: The feature expert pre-processes input, instead of feeding the network with the position only. It detects features of a single unit and calculates the activation of the input neurons corresponding to that unit.

This expert was able to recognize the following features listed in the table:

black or white string	empty intersection
number of liberties (1, 2, 3, 4, ≥5)	liberties of Black if he plays here (1, 2, 3, 4, ≥5)
number of stones (1 or 2, 3, ≥4)	liberties of White if he plays here (1, 2, 3, 4, ≥5)
can be captured in a ladder if string colour plays first	Black can be captured in a ladder, if he plays here
can be captured in a ladder if string colour plays second	White can be captured in a ladder, if he plays here
	eye for Black, n moves missing ($n = 0, 1, 2, \geq 3$)
	eye for White, n moves missing ($n = 0, 1, 2, \geq 3$)

Liberties are empty intersections, a stone placed on an intersection has at most 4 points orthogonal to the stone. To have only 1 liberty would mean that stone (or group to which that stone is a part, can be captured.) Ladders are identified from the point of their start after the pre-ladder pattern has taken shape. **56**

External Expert: Only one external expert existed in the implementation of the NN. It was an extension of an algorithm that could detect a certain class of unconditionally alive strings and territory, along with extensions from another programmer as well. **57**

Playing and Learning

The system chooses the highest value after evaluating all potential moves for a position.

Weight changes are calculated by a backpropagation rule, and are applied instantly after each target is presented to the network. All errors in the evaluation of a string unit are multiplied by the number of stones in the unit. This is done to allow the network to learn how to predict the

overall score accurately. The program was able to achieve equal level of play against a conventional commercial Go program on level 3 of 9 after 4500 games. The programmer concluded that it could perform better with more expert knowledge integrated into it. If experiments were to be performed on the standard 19 * 19 size board, instead of 9 * 9, large computational resources would be required. Also, a way to integrate the evaluation function into a selective search strategy would need to be devised.

VIII Conclusion

Humans use synthesizing methods to understand positions globally and guide tactical maneuvering. Computer programs have traditionally employed search and pattern matching techniques to drive decision making, or rule application. The complexity of Go prevents this low-level, perceptually based approach; more human like reasoning is required. In this paper, we have seen some conventional techniques used, and illustrated the potential for a different kind of knowledge representation that attempts to encode high level cognitive functioning in a hybrid pattern/ algorithmic form of the influence map, adjusted for context. This adjustment substitutes for search and matching and provides a cognitive level knowledge representation from which prospective pattern generation could begin, according to evaluations of overall board balance and strategic concepts.

We see the effect and importance of the cognitive arts' involvement in algorithmic design; Psychology can provide insight into new ways of looking at function design. In the next decades more research in cognition and perception, and other mental features such as memory usage by Go players may provide further tools for AI programming. In the meantime, some programmers have intuitively explored the area of cognitive interpretation and made programs that incorporate the spatial representation analogue of visual perception using set theory based notation. Both the experiment with the integration of cognitive knowledge into perceptual representation on influence maps and the spatial representation material have yet to incorporate analytical components into their models, yet the important point is that the realm of cognition as humans use it in the game of Go has begun in a way that never was required with Chess, relying as it can

on library information and fast hardware, and less on domain knowledge, at least in proportion to what is required in Go.

The Early Grounding model of HTN showed not only an effective way to reduce search space dimensions, but more importantly that it could do so in concert with a representation model that allowed high level abstract descriptions of strategic goal setting to direct the tactical search. Again, this program was unleashed upon the small scale of L & D problem sets, but, as we have seen, the L & D problems are a domain which can expand to full Chess size problems requiring weeks to solve professional Go level problems. Progress at this level is a necessary if insufficient condition to moving out of the amateur Go range, as the Go programs have been for over a quarter century.

Epistemological areas may contribute to AI, using the experimental model practiced by the Psychological realm. Already some have used neural nets with some success at the low Kyu level on 9 line boards. Whether the term 'a priori' actually is represented by the neural network example is a question more of semantics; the important point is that the phenomenon is recognized, and that over the next decades perhaps those working in the philosophical domain may contribute more precisely other aspects of the Kantian realm; the Kant map illustrated many terms that would provide different perspectives on elements or functions yet to be applied to Go programming. The Categories themselves were shown to have analogues to Go processes, which might be incorporated into the natural language features of the neural net example.

J. McCarthy's quote at the beginning of this paper came from an book chapter entitled "Chess as the Drosophilia of AI". Drosophilia is a fly species whose DNA was used along with less developed organisms such as yeast, and the E. Coli bacterium Their DNA provided models for protein sequence discovery in the Genome Project.[58] The human genome can be understood as having 200 books of 1000 pages each; the fruit fly has 10 such books, and yeast 1 book, E. Coli 300 pages. Each provided a stepping stone to understanding more developed levels; as such, Chess be thought of as helping to provide a stepping stone to the human genome project of the AI domain: Go!

The importance of success in the area of Go programming is contained in the fact that achieving expert level playing ability would mean that the program was able to perform in an analogue of human, synthetic functioning. This would imply that the program could generalize from a particular example, and be therefore capable of learning and not be dependent on rule based heuristics that are merely prescriptive in nature, as opposed to regulative. Humans take the regulative rules of games such that Go use, and learn to reason based on principles derived from the experience of the effects of experimenting with the game over long periods of time.

Applicability to other AI areas would not be far removed.

Sources:

I Introduction

01 AI: A Modern Approach, Russell-Norvig Prentice-Hall 1995
02 Kant, Critique Pure Reason St. Martin's Press , 1965, p 113
03 Allis L.V. Searching for Solutions in Games and Artificial Intelligence Ph.D. thesis. University of Limburg, Maastricht. 1994 , Chpt. 1 p. 4 http://www.cs.vu.nl/~victor/thesis.html
04 http://www.usgo.org/computer/
"McCarthy, J.", "Chess as the Drosophila of AI", from Book: "Computers, Chess, and Cognition","Springer Verlag", "1990","Marsland, T. A. and Schaeffer, J.",NY,NY. pg. "227--237"
05 AI: Modern Approach, p. 122
06 ibid. p. 137
07 All Systems Go, http://www.cns.nyu.edu/~mechner/compgo/
08 ibid.
09 ibid.
10 op cit Allis, Chapter 6 p. 4 of 30.
11 ibid. p. 7 of 30.
12 http://www.etl.go.jp/etl/divisions/~7236/Go/ [diagram source link not active]
13 http://www.qmw.ac.uk/~ugah006/gotools/diamond/diamond.html
14 http://www.qmw.ac.uk/~ugah006/gotools/

II Searching Limitations

15 http://www.andromeda.com/people/ddyer/go/search.html

III Pattern Interpretation

16 http://www.andromeda.com/people/ddyer/go/shape-library.html
17 http://cns.nyu.edu/~mechner/compgo/

IV Cognitive Knowledge and Perceptual Representations in Go

18 Integration of Cognitive Knowledge into Perceptual Representations in Go p 1. http://www.psy.uq.edu.au/~jay/research/papers.html
19 ibid. p. 2.
20 ibid. p. 2.
21 ibid. p. 2.
22 ibid. p. 3.
23 ibid. p. 3.
24 ibid. p. 4.
25 ibid. p. 5.
26 ibid. p. 5.
27 ibid. p. 9.
28 ibid. p. 10.

V Spatial Representation

29 Spatial Reasoning in the game of Go, Bruno Bouzy, p. 3 http://www.math-info.univ-paris5.fr/~bouzy/publications/SRGo.article.ps.Z
30 ibid. p. 5.

31 ibid. p. 5.
32 ibid. p. 6

VI Planning In Abstract Domains

33 Adversarial Planning in Complex Domains, Willmott et al.
http://www.concentric.net/~Jgberg/Research/Go/Zipped/
AdversarialPlanningComplexDomain-Willmott.ps.zip
34 ibid. p. 1 col. 2.
35 ibid. p. 2. col. 2.
36 ibid. p. 1 col. 2
37 A.I. A Modern Approach, Russel, Norvig, Prentice Hall, 1995
38 op cit Adversarial Planning... p. 3 col. 1.
39 ibid. p. 4 col. 1.
40 ibid. p. 3 col. 2.
41 ibid. p. 3. col. 2.
42 ibid. p. 4. col. 1.
43 ibid. p. 4.
44 ibid. p. 4 col. 1.
45 ibid. p. 4. col. 2.
46 ibid. p. 4 . col. 2.
47 ibid. p. 5 . col. 1.
48 ibid. p. 6 col. 1.
49 Shared concepts between complex systems and the game of Go by computer programmers. Bouzy, Casenave
http://www.math-info.univ-paris5.fr/~bouzy/publications/UES96.article.ps.Z
50 Office Automation, Policy and Procedure HTML Project, 1997
Used this portrayal of Go and Categories there.

VII Epistemological Levels of Abstraction

51 The Integration of A Priori Knowledge into a Go Playing Neural Network
http://cgl.ucsf.edu/go/Programs/NeuroGo.html
52 ibid. p. 2.
53 ibid. p. 2.
54 ibid. p. 3.
55 ibid. p. 3 & 4.
56 ibid. p. 4 & 5.
57 ibid. p. 5

VIII Conclusion

58 http://www.genome.iastate.edu/edu/doe/prim3.html
http://www.genome.iastate.edu/edu/doe/fig14.html [See figure there]
Used in the Analogical Reasoning for Knowledge Discovery in a Molecular Biology DB paper earlier this semester in CAP 6680.

Cover Game Image: http://www.cwi.nl/people/jansteen/go/games/japan/masters/masters/shusaku/index.html
Note 1: Kant map page 32 made in CAP 5605 Spring 1999 for a response to Turing Test question.
Note 2: Most Go links accessible from http://home.t-online.de/home/markus.enzenberger/compgo_biblio.html
Note 3: Sections **I -IV: Cognitive Knowledge and Perceptual Representations in Go** used in paper for Intro. AI, CAP 5605 Spring 1999, including appendix. It had a different theme, and some re-writing done.

Appendix:

Illustrated Rules

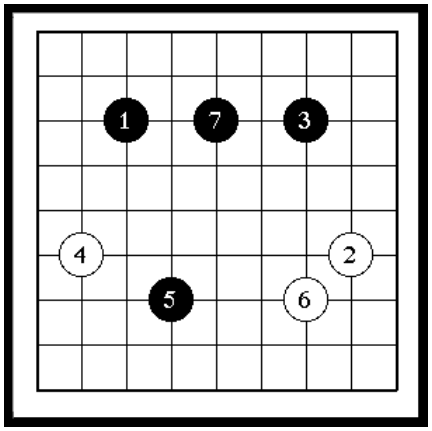


Diagram 1

Start

Stones on intersection.

Pass allowed.

Remain on point unless captured.

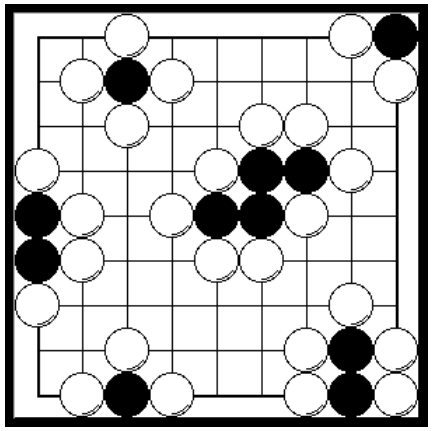


Diagram 2

Capture

Black stones captured and removed.

Orthogonal points surrounded.

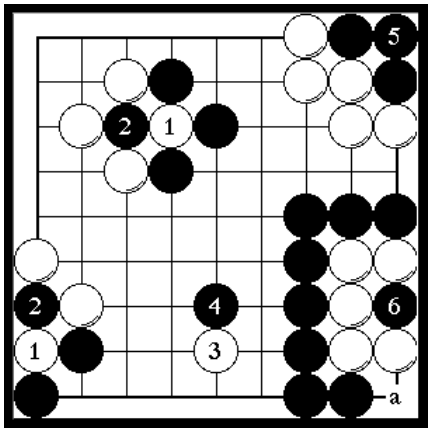


Diagram 3

Illegal
2: KO: Requires move elsewhere, like at 3: then 2.

5 & 6 suicide.

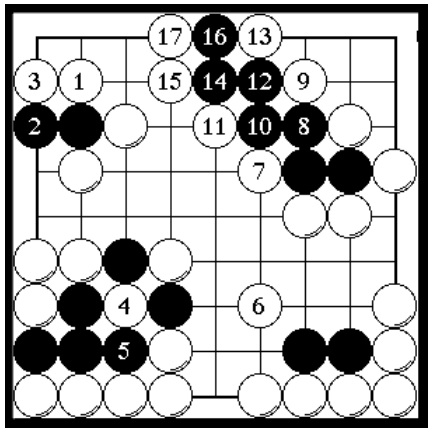


Diagram 4
Capture Sequences:
1 drive against edge.
4 a sacrifice: then at 4 after 5 captures.
6 a net.
7 a ladder.

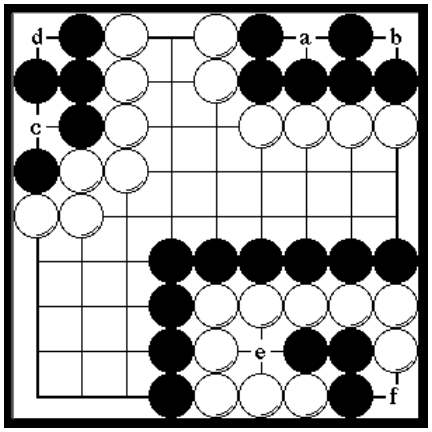


Diagram 5
Living shape with 'a' & 'b'.
'c' **false 'eye'.**
Black dead in **Ko.**

Last group **Seki**, no points, not dead.

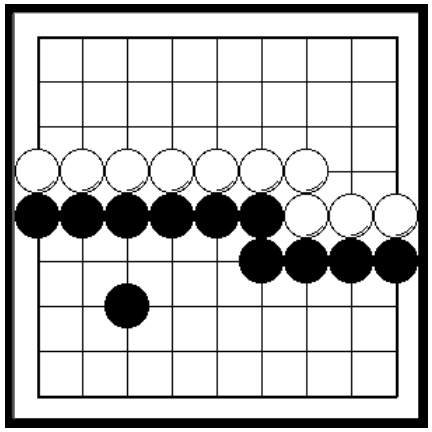


Diagram 6
Scoring:
1. Captured stones put in area of same color.
2. Count vacant points

From http://www.best.com/~webwind/go/goLes/goLes1.htm

Diagram 5: Life and Death concerns the viability of the playing pieces, called **"stones**", to be immune to capture. 'a' and 'b' are called **'eyes'**, and these two points allow the Black group to live in Dia. 6.

Whereas Chess games base the capturing property on not allowing two objects to occupy the same space at the same time, Go **'life'** is based upon the principle of not allowing two moves in a row to be made at a time. If 'a' and 'b' in Dia. 6 could be played at once, the Black group would be captured, as all points orthogonal to each member of the group would be occupied. However, each move alone is **'suicide'**, as by itself a White stone at 'a' or 'b' does not fulfill the capturing rule, and 'suicide' is **illegal.**

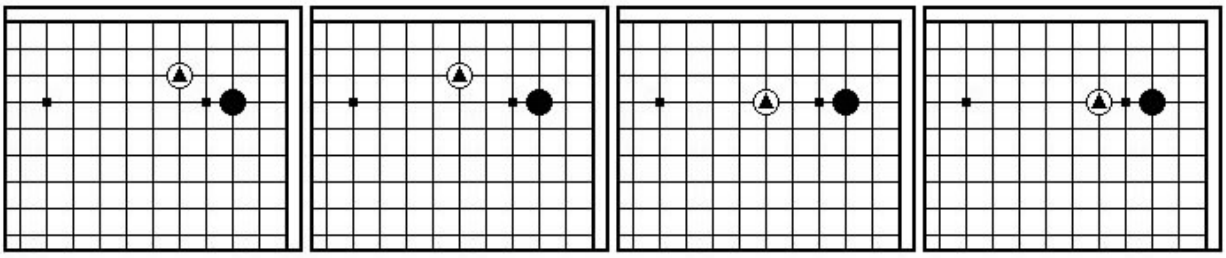
White 'c' surrounds a single stone and can capture at the same time, so this is legal. As such, the Black group is '**dead**', as it cannot make two eyes, though the Black stones there would not be removed from the board and thus '**captured'** until the end of the game, or, unless White were forced to play 'd' in order to remove them.

Note that the third position is a stalemate as far as L & D is concerned; a move by either color would result in the capture of the other. White would be able to make two 'eyes' after capturing Black if Black made a move at 'e' or 'f', but White would lose all if moving first. So neither will play, and no points accrue for either. Therefore, L & D has, as far as search routines are concerned, three states as a result for a group: **Alive, Dead**, or **Seki**, or 'frozen'.

Opening

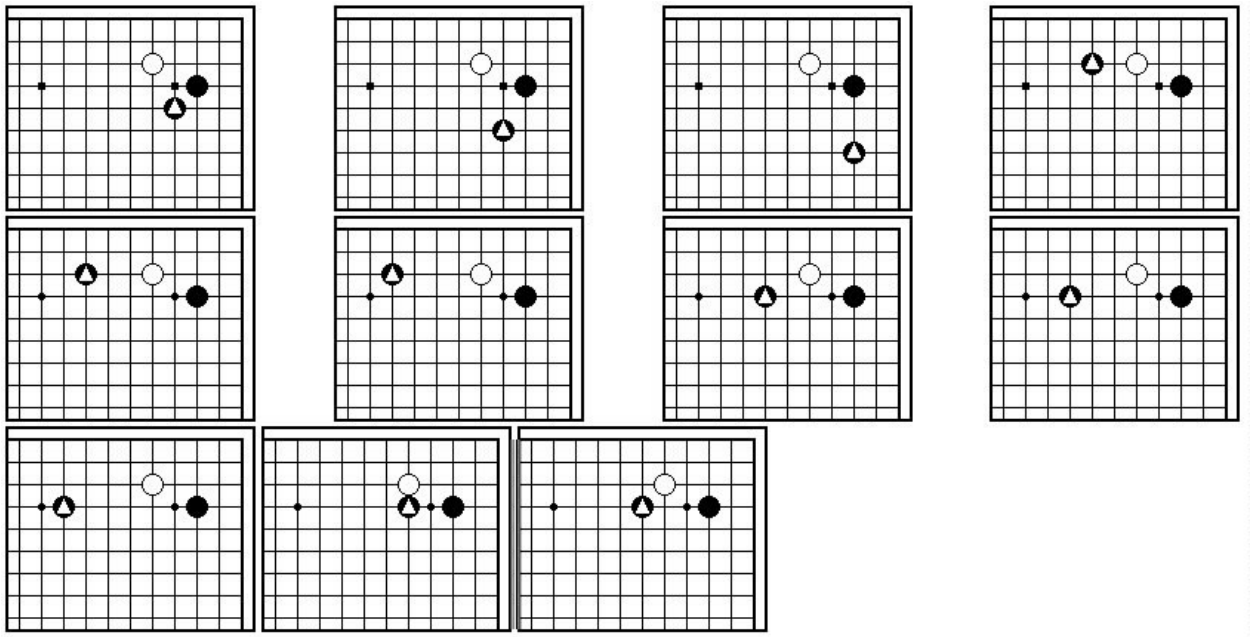
Initial Deployment

Historically the first moves have been played in the corners, perhaps for intuitive reasons. The corners provide the option to make territory or move to sides or center.
Usually 8 points are used . Three are duplicates but with regard to other corner patterns are still relevant. Here are standard options for a response to the 3 4 point deposition:



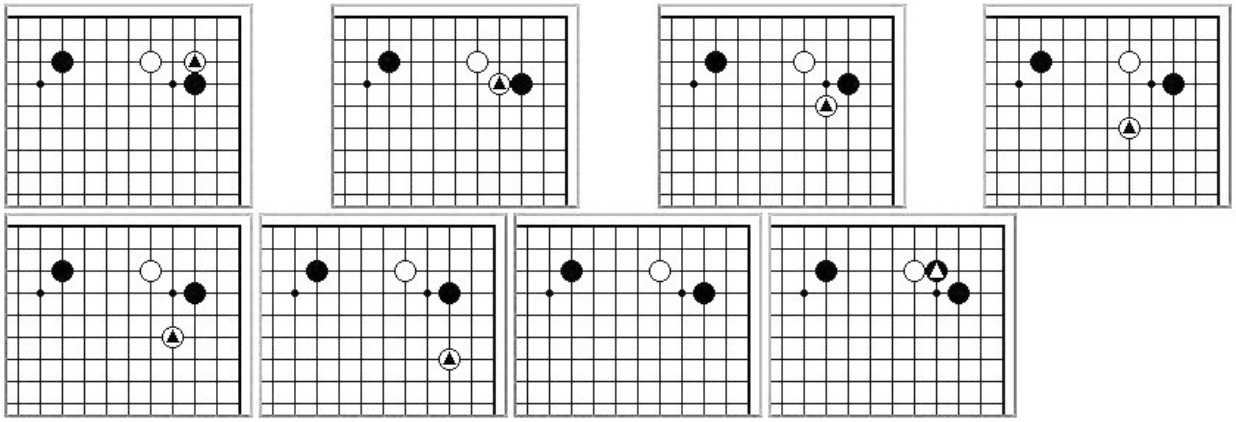
Joseki

From this point Joseki can develop. Jo means 'even' and 'seki' means established, fixed, or researched; so a fair result, locally speaking.



Options are to extend to one side, or 'pincer' the approach stone, attach to the approach, or press down upon it.

To take the 3 space pincer, sample replies are:

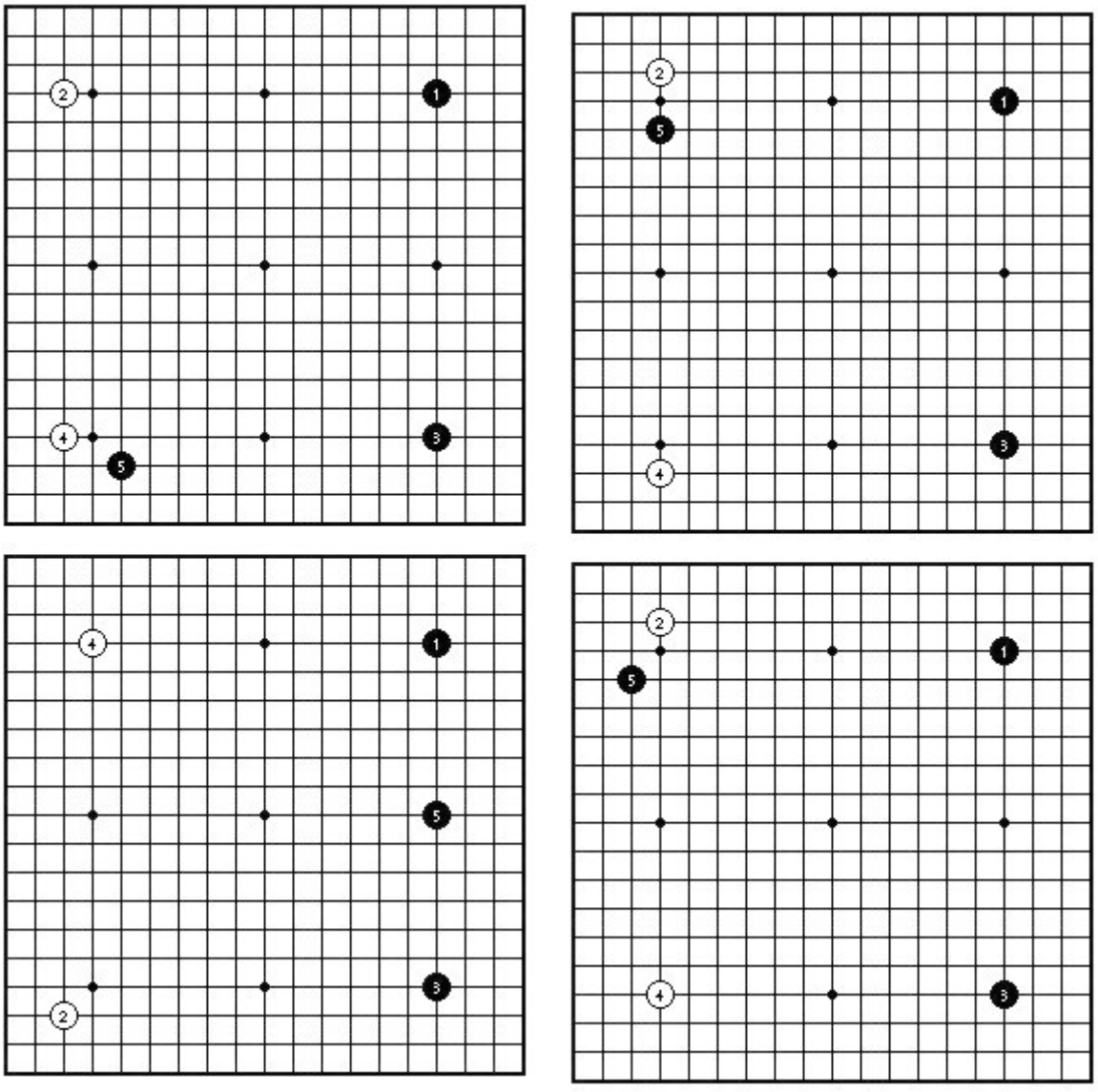


A joseki can be complete at half a dozen moves up to 40 or more. It can contain ladders, ko, seki, life-death issues.
Here we only present the initial moves. Joseki ideally are chosen, or read out, to be appropriate vis a vis the other corners. A mistake in direction, or even a misplacement of one line can upset the balance immediately. Joseki are studied for principles, not for random application. 2000 sets comprise an initial study, 40,000 were catalogued a decade ago.

Fuseki

Properly speaking, Fuseki is the Opening, comprising the first 25 moves until fighting, or contact plays on the sides begin. The initial 4 moves in the 4 corners provide the initialization for fuseki.

Minus duplications due the 8 rotations, this gives 2080 parallel Initial Deployments and 1056 diagonal ID's with the first 4 moves, if each is placed in a separate corner. Then comes the approach to the initial corner move. Here are P1414, P1616, P1115, & P1116 in Rotation 7.

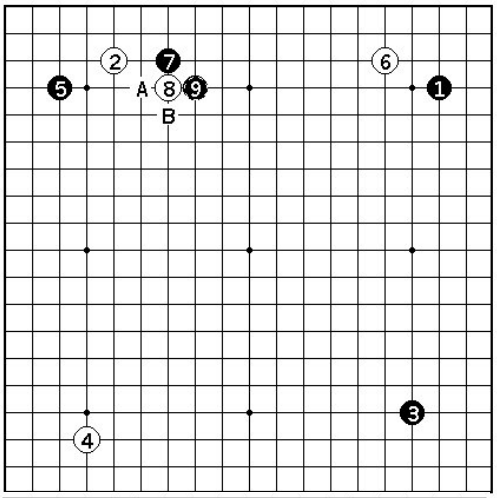


From each initial position of the first half dozen moves thousands of fuseki can be generated;

again, fuseki is studied for gleaning principles, not rote memorization.

JOSEKI, FUSEKI images from http://www.cwi.nl/people/jansteen/go/

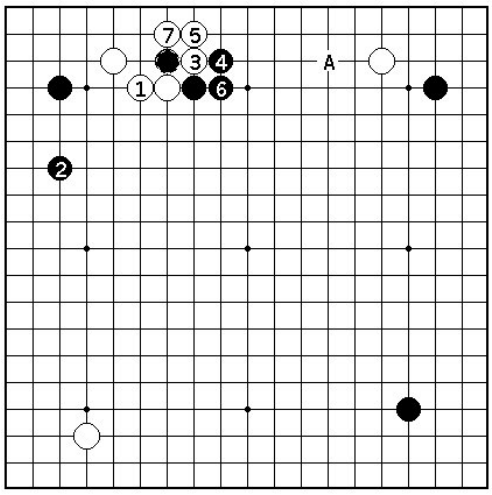
Joseki must be matched to other corners:

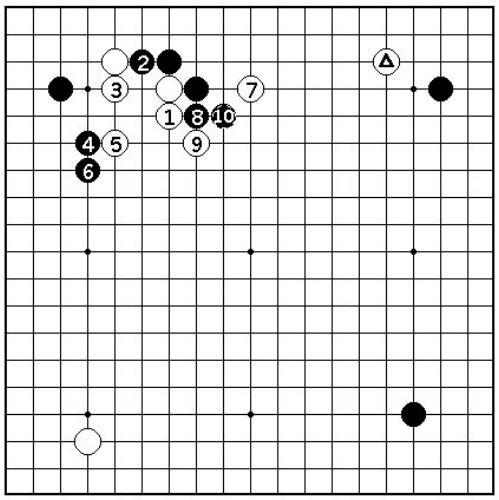


After Bl. 9, Wh. can choose from A or B according to joseki.

Below left, Wh. connects at 1. The result is that Bl. can pincer at A. The attack on the lone white stone is to White's disadvantage.

Below Right, White correctly plays extension with Wh. 1.
The result is that 7 together with the marked White stone allows an attack on the Black string forming with Bl. 2 , 8, and 10.



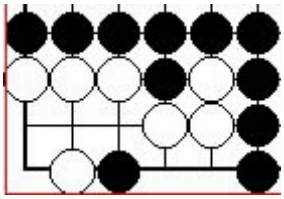


From Whole Board Thinking in Joseki, Y. Yang, 9 Dan, with P. Straus .
http://personal.rdu.bellsouth.net/rdu/4/t/4thline/

This illustrates idea that joseki must be 'chosen' with respect to the whole board development.

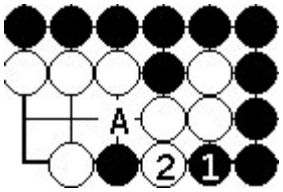
Also, it is generally understood that one must learn principles of Joseki, Fuseki, and their relation in order to play proper openings. A case based program could apply previous Fuseki and Joseki as long as no new move is played. A new move could be a mistake, a trick, an overplay, or a genuine new move based on understanding, reading, or research on the part of the human player.

L&D Search Example.

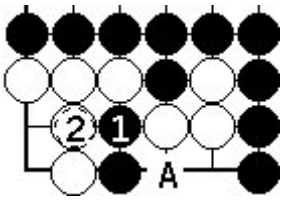


0) Start of Search.

Black to Kill Unconditionally; no Ko, no Seki.

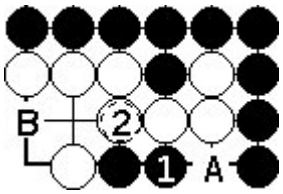


1) Now Black captures at A.
But White A instead of 2 allows eyes at 1-2 point if Black 2,
or with White 2 capture if Black 1-2 point.
So White can live if Black 1.



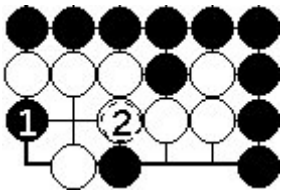
2) Black Cut; After 2, White captures either at A,
or to right of A after Black A.

White lives.



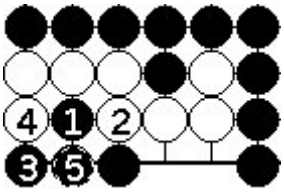
3) Black A, White B gets 2 eyes.
Black B, White A gets 2 eyes.

White lives.

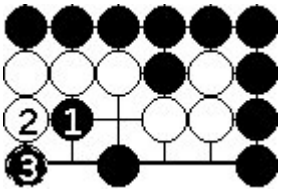


4) Black stone below 2 dead, so one eye there
and one in left area.

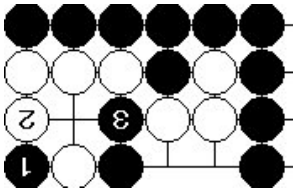
White lives.



5) [White stone at 5 to start] After Black captures that stone with 3,
White atari answered with connection at 5. White captures 4 stones,
but then Black at 5 again. This is 'vital point' destroying eye shape;
White can only get 1 eye here. Black Kills. But ...



6) KO: White 2 instead of 2 above allows Ko with capture of 3. White
could live if wins Ko fight.



7) Solution: Black sacrifices a stone to produce 'shortage of liberties'
for White. Now, any White move results in immediate capture of
half the stones. White 2 to right of 2 same result.
B. Terry: American Go Journal: Vol. 25 No. 4, Fall 1991, p. 18.