

THE STRUCTURE AND PERFORMANCE OF THE INTERIM.2 GO PROGRAM.

Walter Reitman and Bruce Wilcox
University of Michigan
205 Washtenaw Place
Ann Arbor, Michigan 48109

We describe interim variants of a Go program based on skilled human models. The program maintains a permanent, selectively-updated, multilevel network analogous to the skilled player's perceptual and cognitive representation of the game situation. Moves are chosen by a hierarchy of experts and critics operating on this network. Lookahead is a selective, goal-related process, one among many components of Go skill. After an overview of system organization, we illustrate the program's performance with three segments from actual games.

1. PROGRAM DESIGN CONSIDERATIONS

Berliner [1] suggests that though whole-board exhaustive search programs may one day play championship-level chess, "such an approach cannot possibly work for Go, and this game may have to replace chess as the task par excellence for AI." Go certainly has many features that recommend it for this purpose. Its primitive elements, black and white stones played on the 361 points of a 19 x 19 grid, generate complex patterns and problems that tax the highest levels of professional skill.

Go games typically run 200 moves or more, and each turn offers on average close to 250 legal plays. Brown and Dowsey [2] estimate the number of possible games of Go at about 10^{700} , compared with 10^{120} for chess, and they point out that exhaustive Go search only three moves deep would entail generating and evaluating about 8,000,000 whole-board positions. Yet videotaped game records [3] show that advanced amateur players can look 30 moves ahead, and the Go literature contains still deeper lookahead sequences. This means that lookahead in human Go play is a highly focussed, selective, goal-related process. The size of the possibility space appears to rule out a Go program organization analogous to the exhaustive whole-board lookahead and evaluation schemes used in most current chess programs.

The game also has a very strong visual or perceptual element. Local stone patterns and

large-scale spatial relations among groups and territories both are crucial. Most of these patterns and relations continue unchanged from any one move to the next. This argues for a permanent, selectively-updated, multilevel network of data structures to represent the elements, patterns, and properties, as well as the interrelations within and across levels.

Finally, both positive and negative interactions are extremely significant at all levels. To play Go well, it is necessary not only to determine the most important objective at any given time, but also to advance it in ways that simultaneously achieve as many secondary benefits as possible. This motivates the use of hierarchies of experts and critics, each responsible for a class of options, properties, or functions associated with elements of a given type and level.

2. DATA STRUCTURES AND CONTROL FLOW

With these considerations in mind, turn now to Fig. 1¹, which shows the flow of control in the INTERIM.2 program. At the top level, this consists of a two-part cycle. First, immediately following a play by either side,

-----¹-----
Fig. 1 is a simplified representation. Many function names have been changed for mnemonic reasons. Many other functions at the control levels depicted in the figure have been omitted altogether. To take an extreme example, DEVELOP.GROUP actually consists of about 20 distinct option specialists. Finally, PROBE is called at many points other than those shown, and is bracketed to indicate this.

¹ Support for this work under NSF grant MCS77-0880 is gratefully acknowledged.

```

MOVE
  UPDATE.POINT.TYPES
  UPDATE.STRINGS
  UPDATE.LENSES
  UPDATE.LINKS
  UPDATE.GROUPS
  UPDATE.WEBS
  UPDATE.SECTOR.LINES
  UPDATE.TERRITORIES
  UPDATE.TACTICAL.ANALYSES
  [PROBE]
  UPDATE.GROUP.STABILITY.ESTIMATES
REFLEX
  LOCAL.URGENT
  PLAY.JOSEKI
  STRING.ATTACK.AND.DEFEND
  LINK.ATTACK.AND.DEFEND
  RE.EDGE.LINK
  [PROBE]
  VITAL.SHAPE.POINT
  CONTACT.FIGHT
  DEVELOP.GROUP
  KILL.CUTTING.STRING
  EXTEND.AND.SQUEEZE
  COUNTERATTACK
  CROSS.SECTOR.LINE
  STABILIZE.POTENTIAL.TERRITORY
  RUN.TO
  ATTACK.GROUP
  WIPE.OUT
  SHAPE.POINTS
  [PROBE]
  SQUEEZE
  ENCLOSE
  RUN.TOWARD.DEAD.GROUP
  EXTEND.SECURE.INVADE
  DEVELOP.POTENTIAL.TERRITORY
  ENDGAME

```

Figure 1. Program Organization.

MOVE updates all those elements of GAMEMAP, the program's multilevel representation of the situation, that have been affected by the latest move. Next, if it is the program's turn to play, REFLEX chooses a move.

GAMEMAP is indexed through a set of interrelated GAMEBOARDS. Each board is an array that points to all data structures of a particular type. At the lowest level, the TYPEBOARD gives quick access to bit pattern information about specific points (empty, occupied by black, part of a white territory, etc.). Some boards, e.g., STRINGBOARD and LINKBOARD, index the basic game elements at the various levels (strings, linkages, groups, territories, and potential territories). Thus

in Fig. 2, for example, looking up the board point B8 in the LINKBOARD array enables any program function to access the data structure

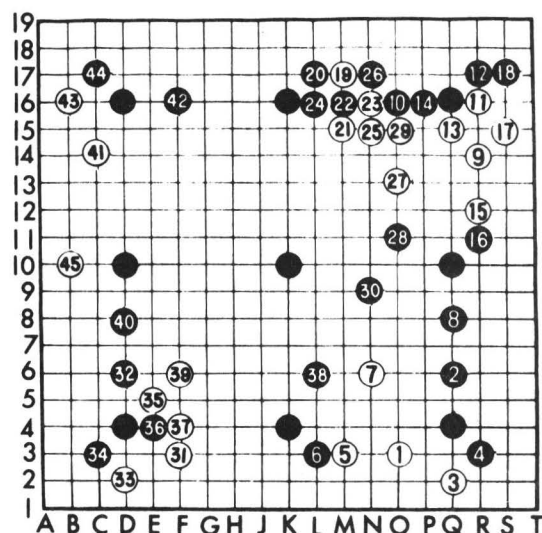


Figure 2. The First 45 Moves.

representing the linkage from the D8 string to the left edge of the board. Other boards, e.g., LENSBOARD, WEBBOARD, SECTORBOARD, and TACTICSBOARD, index elements analogous to those in a skilled Go player's perceptual and cognitive representation of the current situation. Lenses represent and monitor local stone patterns. Web data structures are surrogates for certain kinds of visual scanning. Webs follow the external configuration of groups, radiating out and indexing the surrounding board points. Thus they are useful in analyzing and detecting changes occurring around individual groups. Referring again to Fig. 2, looking up B8 in the WEBBOARD, for example, gives access to the webs around white B10 and around the black group on the lower left. Sector lines, e.g., the one between B10 and C14, correspond to those looser relations among stones that dynamically divide the board into tenuously defined regions. These sector barriers help define potential territories and strategic threats to groups. Finally, the TACTICSBOARD indexes tactical data structures associated with individual strings, linkages, and significant board points. These data structures record the results of analyses produced by calls to PROBE, the entry point to

the subsystem responsible for all INTERIM.2 lookahead. This is selective, best-first lookahead. Each such analysis answers some specific tactical question about the associated element. The resulting tactical data structure is maintained as part of GAMEMAP unless invalidated by subsequent moves that affect strings or board points involved in the analysis, at which point PROBE is called to reanalyze the situation.

2.1 Updating the Representation after a Move

To fix ideas, consider the situation shown in Fig. 2. Ignore the order of play for the moment. As soon as white plays move 45 at B10, MOVE is called to update GAMEMAP. This updating is selective. Only structures directly or indirectly impacted by the move are affected. UPDATE.POINT.TYPES alters the TYPEBOARD entry for B10 from UNOCCUPIED to WHITE-OCCUPIED, and UPDATE.STRINGs sets up a string data structure representing B10. Now UPDATE.LENSES creates a lens that will monitor any future board changes around the pattern formed by white B10 and black D10. In addition, it sets up a shape lens which it associates with B10. This lens, when queried, will suggest locally good developmental plays for white, e.g., at B8 and B12. The existing shape lens associated with D10 is modified, since white B10 alters the shape development possibilities for D10. Finally, the existing lens monitoring the D8-D10 linkage also is altered, since white B10 changes the possibilities available for attacking and defending that linkage.

White 45 disrupts the linkage between black D10 and the edge of the board. Therefore, when UPDATE.LINKS is called, it deletes from GAMEMAP the data structure representing this linkage. Such changes in higher-order data structures may induce further changes in lower-order ones. This change, for example, results in UPDATE.POINT.TYPES deleting from the TYPEBOARD the information that A10, B10, and C10 are involved in a black linkage to the edge of the board.

Some plays, e.g. white 43 and black 44 at the top left of Fig. 2, add stones to existing groups. Others, such as white 45 at B10, create new groups. UPDATE.GROUPS is responsible both for modifying existing group data structures and, as in the present case, creating new ones. GAMEMAP represents white B10 at both the string and group levels because the analyses and decisions at the two levels entail quite different kinds of considerations. Next, UPDATE.WEBS creates a web data structure

indexing the board points surrounding B10. The web reaches out to the white group at B16 and C14, the open area around E12 and E13, and the black group beginning at D10 and running down toward the bottom of the board. Finally, as the B10 stone impinges on the existing webs radiating out from the white group on the upper left and the black group running down the left side, UPDATE.WEBS also modifies those webs, to reflect the presence of this white stone in their vicinities. These web data structures may be used by any REFLEX functions considering attacks on adjacent enemy groups, or defensive plays running a group toward friendly groups or out into unoccupied areas.

UPDATE.SECTOR.LINES now creates a data structure representing an imaginary line running from B10 to C14. This sector line is used by UPDATE.TERRITORIES when it creates a potential territory data structure corresponding to the area bounded by the two white groups on the left of the board. UPDATE.TERRITORIES also modifies the territory data structure running between the lower left edge of the board and the black group running down from D10. Before white 45, that territory was bounded at the top by the linkage from D10 to the left edge. Since UPDATE.LINKS has deleted that linkage, the territory must be redefined. This is done by designating the next nearest intact linkage, from D8 to the edge, as the new upper bound. This higher-level modification also induces changes in the TYPEBOARD, since points such as B9 and C9, which formerly were black territory points, now no longer are.

Now UPDATE.TACTICAL.ANALYSES calls PROBE to assess the tactical status of strings and linkages created or potentially affected by the most recent play. No actual lookahead is required for the B10 string, which is found to be tactically secure on the basis of context information from GAMEMAP. When contextual information is insufficient for a decision, PROBE may generate lookahead sequences running from a few hypothetical board positions to 50 or more. It is worth reemphasizing two points here. (1) All PROBE lookahead is intended to answer a specific question, e.g., about the tactical security of a string or linkage, posed by the function invoking PROBE. (2) PROBE records its results for each such element in a tactical data structure which is maintained as part of GAMEMAP until the analysis is called into question by some subsequent move. This tactical data structure can be accessed both from the data structure representing the string or linkage, and also through the TACTICSBOARD (see Section 5 for an example).

Finally, UPDATE.GROUP.STABILITY.ESTIMATES provides a stability rating for the B10 group. It also alters the stability rating for the black group running from D10 to C3. This is because white 45 has undercut the side territory associated with the group. This information will now be available to DEVELOP.GROUP when next it is invoked from REFLEX.

To summarize, MOVE and its subfunctions serve at least three vital purposes. They carry out, at progressively higher levels, the aggregation and analysis of the patterns and structures representing the game situation. Thus they provide the basic elements the REFLEX functions deal with. They maintain the data base that allows the REFLEX functions to deal with each element in context. This enables the program to locate interactions and find moves that do several jobs at once. Finally, MOVE updating corrects the representation after each play, deriving its immediate implications at every level it affects. This is how the program finds out the consequences of its own moves, and those of its opponent.

2.2 Choosing the Program's Next Move

Returning now to Fig.1, REFLEX is a temporary control structure that steps through an ordered list of functions and accepts the first move returned from them. LOCAL.URGENT and its subfunctions are responsible for dealing with any urgent problems that may have arisen as a consequence of the immediately preceding black-white move pair. Each subfunction uses information generated by MOVE, and each has a limited ability to assess the local significance of the class of problems it deals with. If STRING.ATTACK.AND.DEFEND finds that a significant friendly string is threatened with capture, it will attempt to generate a move to save it. If the threatened string serves no important function, however, control passes to the next subfunction. The situations dealt with by the LOCAL.URGENT subfunctions are illustrated in the examples that follow.

If there are no significant locally urgent situations resulting from the immediately preceding moves, control passes to DEVELOP.GROUP. This function, working both with the group stability data computed under MOVE, and with assessments of the values of each of the program's groups, selects the friendly group it considers to be most worth developing. It then invokes its subfunctions in the order shown in the figure. Each has information about a class of developmental

options, the conditions under which these options are appropriate, and the means available for realizing them. KILL.CUTTING.STRING, for example, determines whether there are any enemy strings separating the group to be developed from an adjacent friendly group. If so, it will try to kill one of them. Several other DEVELOP.GROUP subfunctions, e.g., CROSS.SECTOR.LINE and STABILIZE.POTENTIAL.TERRITORY, also are described in the examples that follow.

If there are no valuable friendly groups in need of development, control passes to ATTACK.GROUP. This function considers the various enemy groups, again ordered by value and stability, and attempts through its subfunctions to generate suitable attacking moves against its target. Section 4 presents a detailed illustration of attacks generated by the SHAPE.POINTS subfunction.

If there are no attackable enemy groups, control passes to the remaining midgame and endgame functions, in the order indicated. Thus RUN.TOWARD.DEAD.GROUP is a midgame function that attempts to reduce the scale of capture of a friendly group. This function may be called for any of its groups the program considers to be strategically dead, i.e., incapable of development to the point of absolute stability.

To summarize, the move choice process proceeds from general problems and options to more specific ones. The system relies on the experts at each level to know a good deal about the options available there. This has two consequences. First, usually only the best few possibilities at any level are considered. Second, if higher level experts have chosen properly, then lower levels also need only consider the best more specific options. Thus, referring ahead to Section 4, if REFLEX is correct in concluding that at that point in the game its best option is to attack, and ATTACK.GROUP is correct in its choice of an appropriate target and a specific means of attack, then SHAPE.POINTS need only come up with the plays that best carry out this hierarchy of objectives in this situation. There may be other considerations at a lower level that might improve the final choice there, but that is the responsibility of the critics at that level. Since the functions at each level generally provide an ordered set of choices, the critics have alternatives to work with. And if the situation changes, so that the specific low-level objective no longer has priority, both the situational change and its

immediate implications will be picked up on the next pass through the MOVE-REFLEX cycle.

2.3 Implementation

The INTERIM.2 programs are written in LISP/MTS and run on the AMDAHL/470. They compile into about 330k words, use about six cpu seconds per black-white pair of moves, and create about 86 hypothetical board positions per move pair. Real response time usually is a good bit faster than that of the program's human opponents. For additional information about lenses, webs, PROBE, and all aspects of the program code, see [4], [5], [6], and [7], respectively.

3. OPENING GAME PLAY

We begin our examples with Fig. 2, which shows the first 45 moves from a nine-stone handicap game against a 10 kyu opponent. White's first two moves, and the program's responses, form a standard, mutually equitable, sequential pattern (joseki). The program plays out such sequences from information stored in a joseki lens. This is accessed under the control of the PLAY.JOSEKI subfunction of LOCAL.URGENT. We include in the program's knowledge base only a handful of the thousands of such patterns, because we are mainly concerned with developing the program's ability to analyze situations and generate appropriate responses on its own.

With black 6 at L3, the program leaves the joseki. Black 6 is generated by LOCAL.URGENT, because white 5 undercuts a black potential territory to the left of K4. Black 8 is motivated by white's jump at 7 to N6. Prior to white 7, the black group in the lower right is considered stable, in part because it is adjacent to an unthreatened potential territory (the area along the right side between Q6 and Q10). Edge potential territory is considered secure if neither of its side boundaries is (1) undercut, or (2) defined by a single stone flanked by an enemy stone (in which case an enemy invasion of the potential territory would, simultaneously, threaten the boundary stone), and if (3) there is no nearby enemy stone with access to the outer boundary. This third condition is assessed by determining whether there are any enemy webs that cross into the potential territory. After white 7 at N6, MOVE finds this third condition violated. The potential territory is therefore declared unstable, and this then causes UPDATE.GROUP.-

STABILITY.ESTIMATES to declare the adjacent black group unstable.

When control passes from MOVE to REFLEX, since there are no significant LOCAL.URGENT situations, DEVELOP.GROUP is called to restabilize the lower right black group. Finding no good attack on an adjacent enemy group, DEVELOP.GROUP calls STABILIZE.POTENTIAL.TERRITORY in an attempt to restabilize the adjacent potential territory. This produces the play at Q8, which restabilizes the potential territory, unites the Q4 and Q10 groups, and lays claim to what is now a real territory along the side. After white 9 at R14 and black's joseki lens response at O16, white 11 at R16 branches off into a joseki not in the program's repertoire. Once MOVE has updated GAMEMAP, REFLEX calls LOCAL.URGENT, which finds three alerts. White 11 threatens black's claim to corner territory. It initiates a contact fight, because white's stone has been played in contact with one of black's [8]. And it creates a vital shape point at Q15. Reestablishing the territorial claim takes priority because the territory is large and contributes to the stability of the Q16 group. Therefore, RE.EDGE.LINK is called. Using lens pattern data for such a local configuration, RE.EDGE.LINK comes up with two move candidates, R17 and Q17. Of these, R17 is preferred because it gives white less room to maneuver. R17, however, would set up a linkage to Q16 that might, in some board configurations, be subject to a successful white cut. Thus RE.EDGE.LINK now invokes PROBE, to see whether the cut can be sustained in this context. PROBE finds it can not be, so REFLEX returns R17 for black 12.

White 13 at Q15 also triggers several LOCAL.-URGENT alerts. The first priority is to protect R17. Any move proposed is checked by PROBE lookahead to see if it also secures the threatened Q16-R17 linkage. The proposed move is further checked against two other sets of criteria, those applying to contact fights and those for plays around territorial boundaries. In this case, the move finally chosen satisfies both the priority objective and these others as well.

White 15 again threatens to undercut a territory, and so provokes black 16 at R11. This is appropriate since the opponent is stronger than the program, the territory is large, and protecting the territory helps keep the black side group stable. White 17 at S15 triggers a LOCAL.URGENT shape alert, since it threatens the followup at S17. The program

* Go ranking is explained in Section 6.

therefore plays 18 at S17, to deny white that point. The idea is to preempt and defuse such situations whenever it is locally advantageous for the program to do so.

With 19, white invades the program's top potential territory before it gets too large and secure. The program's response at L17, its first bad move, results from a bug. White 21 also is an error, however. And now through black 26 the program takes advantage of white's error to repair its own. MOVE updating discloses the threatened black linkage between L17 and O16, and this triggers a priority LOCAL.URGENT alert. When white 23 threatens to reconnect his stones and capture black M16, the program connects at L16. It does this under LOCAL.URGENT control, because the PROBE lookahead indicates that this starts a sequence that saves this stone, protects the L17-O16 linkage, and captures white M17. M17 is a cutting string, and capturing it unites the two black groups. Thus, when white 25 connects at N15, the program concludes the capture sequence with black 26 at N17. Note that the program plays 24, even though its shape critic complains about the inefficient shape it produces, because no other move achieves its priority objective.

White 27 and 29 put no LOCAL.URGENT pressure on black, so for the first time since black 8, control falls through to DEVELOP.GROUP. Now is black's chance to take the initiative. The program, however, believes that the whole black group on the lower right is unstable, so CROSS.SECTOR.LINE is called, and wastes 28 and 30 trying to restabilize it. Why does the program worry about this group? Because white Q2 undercuts the supporting territory. A human player can see the intervening corner, which mitigates this effect, but the program lacked the appropriate check. As this example suggests, making effective use of the information provided by the system's analogues for perception is a recurring problem.

Moves 31-34 in the lower left are joseki again. When MOVE updates after white 35, however, PROBE finds that although white E5 does not threaten any black strings or linkages, it does create a white linkage between F3 and E5 that can be cut. The program classifies this as an easy cut. The first move creates no new string and thus does not lead to complications. A LOCAL.URGENT alert is triggered, the program plays black 36 at E4, and white responds at F4. The idea here is to introduce defects in the opponent's position. These defects may or may not be utilized subsequently, depending upon

the opponent's response and the surrounding conditions. In this case, after white 37 the program does not cut at F5 because such a play would not be supported by any existing black group. Here we see a real limitation of the INTERIM.2 program, the lack of strategic analysis. A strong Go player would have made a strategic analysis before playing 36, found he could make no immediate use of the cut at F5, and deferred the preparatory move at E4 until he could see some strategic purpose to the sequence.

White 37 does not produce any LOCAL.URGENT alerts, but since white 35 created a sector line between E5 and N6, DEVELOP.GROUP calls CROSS.SECTOR.LINE, which proposes black 38 at L6. This takes the unstable black bottom center group out across the line. White 39, played to protect the cut point at F5, destabilizes the lower left black potential territory and group, and induces black 40 at D8. The motivations and operations are the same as those described for white 7, black 8 on the right side. The next four moves are joseki again, followed by a white slide to B10, at which point the opening terminates and the game moves into a protracted midgame fight.

Summarizing the results of the play to this point, although the program has made several errors, mostly owing to minor bugs, it nonetheless has secured all of its groups, and it has center influence, 60-70 points of reasonably secure territory, and prospects for more. White still has insecure groups, only about 15 points of secure territory, and no immediate prospects of gaining much more. This is a high handicap game, so we should expect black still to be ahead. But white can hardly be satisfied with his progress.

4. MIDGAME SHAPE ATTACKS

The second sequence we consider starts from the situation shown in Fig. 3. The sequence itself is presented in Fig. 4. (Move 111 is shown in the figure as 11, etc.). White 111 at M4 triggers no LOCAL.URGENT or DEVELOP.GROUP alerts, so control falls through to ATTACK.GROUP. There is only one attackable candidate, the white group at the upper right. The program has been making preparatory moves against this white group for some time, and now at move 112 the program tries for the kill. The white group has one internal liberty (eye) at P15, but to live it needs either to get one more, or else to connect to some other secure white group. With white already bounded and enclosed, the attempt to kill is largely a

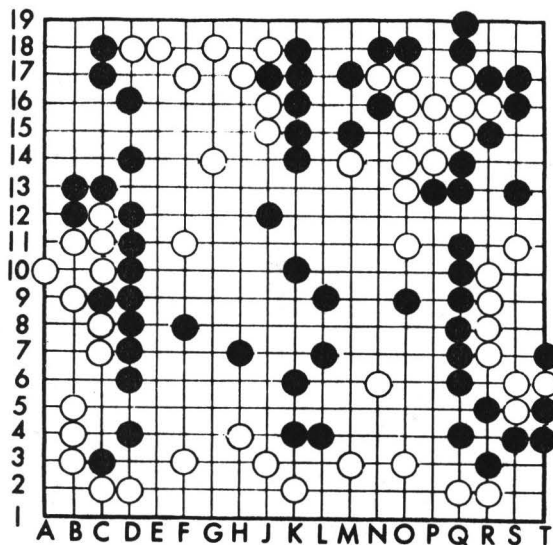


Figure 3. Board Prior To Attack.

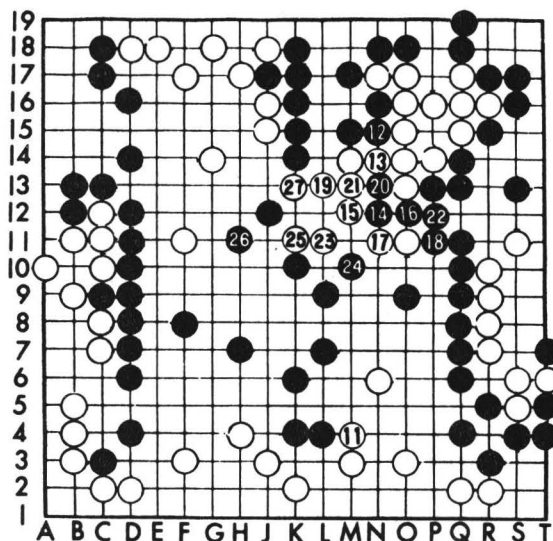


Figure 4. The Shape Attack Sequence.

matter of shape attacks to keep white from forming eyes.

The attack is under the immediate control of

SHAPE.POINTS. To understand how shape attacks are made, recall from Section 2.1 that whenever any stone is played, a shape lens is set up to monitor its shape-development possibilities. The prototype fields for the lens will check for significant proximal stones. Take the white stone at M14 in Fig. 3. When this stone was added at move 99 to the existing white group, one of the prototype shape fields checked for a two-stone pattern at a one-point jump from the focal M14 stone, and found white O14 and O13. A field covering this base shape of three stones (M14, O14, and O13) was then instantiated as part of the lens monitoring the local area around M14. Every completed lens includes fields monitoring each of the recognized base shapes involving the focal stone. Each field is maintained until its occupancy preconditions are violated by a subsequent move. Every completed lens also lists good moves for developing, and spoiling, the eye potential of the board area monitored by the lens. Thus in our example SHAPE.POINTS can now read off white's shape development moves, e.g., M12 and M13, as well as the local black shape attack points. In general, the more stones in the base shape of a field, the faster the empty points in the field can be used to create a good eye, using the eye potential of all the base stones working together. The final shape attack move is chosen on the same principle, i.e., it is the move that deprives the largest possible subset of enemy stones of their eye-forming potential. There presently is one severe restriction on permissible shape attack moves. The play finally chosen must form a connection with an existing friendly group. This is a temporary restriction, to reduce the possibilities for complications, and does not apply to eye-destroying moves, e.g., throw-ins, played inside real territory.

After black N15 and white N14, the dominant white lens field is the one defined by the triple of M14, N14, and O13. This suggests attacking moves at M12 and N12. Both form connections with black groups, but the lens information designates N12 the more severe. It destroys white's shape in this area completely. In addition, N12 also threatens the white O11-O13 linkage. Black 114 induces 115 at M12. This is the only reasonable move for white. Black 116 is played from a LOCAL.URGENT alert, to save N12. N11 would have been better, to further reduce white's shape possibilities, but the early INTERIM.2 versions cannot exploit interactions across the top level REFLEX functions, in this case across LOCAL.URGENT and ATTACK.GROUP. After white 117 at N11,

LOCAL.URGENT proposes P11, to save the black string. P12 would have been better, but this again would require multiplexing across top-level functions. White 119 at L13 both continues trying to make shape, and also threatens one of the enclosing black linkages. Black 120 directly blocks an eye, but L11, aiming at falsifying the eye and also denying white some additional shape possibilities, would have been better. Finding this move takes lookahead, however, and the shape attack system presently uses PROBE only for determining safety and connection. After the M13, P12 exchange, white continues to try to make shape with 123 at L11. At this point, black has two possibilities, at M10 and K12. The program, again not able to call PROBE, plays 124 at M10, the wrong choice. White 125 at K11 contributes both to his shape and his breakout possibilities. Not being capable of strategic analysis, the program is compelled to abort the shape attack at this point in order to reenclose white with 126 at H11. As white 127 at K13 threatens both to break out and also to form an eye, the white group is now secure, and the program abandons the attack.

5. PROBE LOOKAHEAD PRODUCES A KILL

We conclude with an example of the INTERIM.2 system at its best. Fig. 5 shows a midgame situation from a game against a 3 kyu opponent.

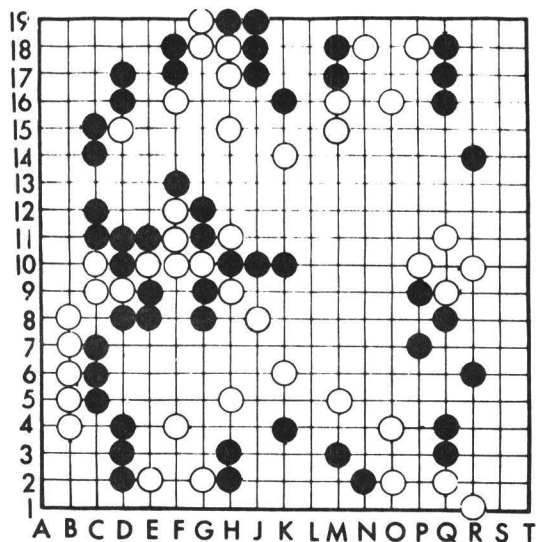


Figure 5. Board Prior To PROBE Reanalysis.

Although white has lost a small group around F10, he is otherwise in fine shape. In particular, white has enclosed the program's top and bottom groups tightly enough that both black and white are treating them as strategically dead, incapable of achieving absolute stability. Per Fig. 6 (move 101 shown as 1, etc.), the program plays its next move, black 86, at H13. White evidently

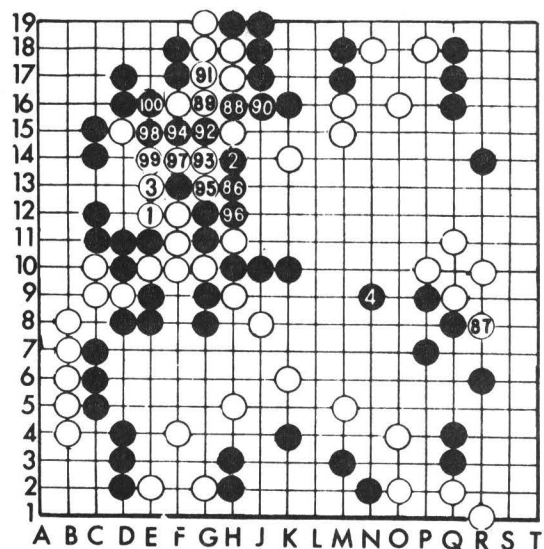


Figure 6. PROBE Lookahead Produces A Kill.

regards this move as locally insignificant, and he plays 87 on the right side. The program then plays black 88 at H16, and in the 16 moves that follow, not only kills the white G19-G18-H18-H17 string, but thereby also saves his entire top group.

How did this come about? Ever since white's efforts to save his F10 stones led to black and white plays around G12, the program has been reanalyzing the tactical status of the white H17 string after every move in the G12 area. This is because PROBE previously had determined that any change in the status of the empty points around G12 might invalidate its conclusions about the tactical security of the H17 string. Therefore it put those points in the relevant points list of the tactical data structure associated with the H17 string, and also stored a pointer to this tactical data structure with each cell in the TACTICSBOARD corresponding to one of these relevant points. Thus, once black has played 86 at H13, on the

next entry to the MOVE-REFLEX cycle, MOVE updating accesses this tactical data structure through the TACTICSBOARD cell for H13 and passes it to UPDATE.TACTICAL.ANALYSES. This time, when PROBE reanalyzes the tactical status of the white H17 string, it comes up with a killing sequence. When REFLEX takes control, LOCAL.URGENT discovers that an important enemy string has just been flagged as killable, and it returns H16, the initial move proposed by PROBE, as the program's next play. The program's performance here is at the intermediate amateur level or better, both in detecting the opportunity, and in following through.

6. OVERALL PROGRAM PERFORMANCE LEVEL

Human Go players are ranked on a scale consisting of about 44 unit-interval steps, labelled as follows. A total novice is ranked 35 kyu. As a player's skill improves, the numerical value of his kyu ranking decreases. Thus intermediate amateurs conventionally rank from about 5 kyu to 1 kyu. The next rank after 1 kyu is 1 dan. Advanced amateur ranks run from 1 dan to an upper limit approaching 9 dan. Professional players roughly overlap the region from 6 to 9 dan on the amateur scale.

The groundbreaking programs of Ryder [9] and Zobrist [10] rank close to the total novice level. The one game Ryder reports was played against such a novice. The program's play was quite interesting at many points, but it eventually lost by a substantial margin. The stronger of Zobrist's two versions achieved a rank of about 31 kyu. The first of our programs, INTERIM.1 [4], ranked about 27 kyu. The initial INTERIM.2 versions, from which the examples presented above were taken, rank about 20 kyu. For an evaluation of an entire INTERIM.2.0 game, see Mann [11]. Introduction of the PROBE system is responsible for most of the difference in playing strength between the INTERIM.1 and INTERIM.2 systems. Our current version, INTERIM.2.3, which allows multiplexing at top REFLEX levels, ranks around 15 kyu.

7. CONCLUSION

We have described the operations that create, update, and use the INTERIM.2 program's multilevel representation to play Go. We also have illustrated the system's performance with three examples from actual play. As these indicate, the interim versions of the projected system produce sustained play, serving reasonable goals, at approximately the level of a knowledgeable beginner. The system has a good way to go, however, before it can claim to

be an adequate solution to Berliner's AI task par excellence.

ACKNOWLEDGEMENTS

We are much indebted to D. B. Benson, H. J. Berliner, D. J. H. Brown, B. Hilditch, D. McArthur, R. Nado, and J. Reitman for their very helpful comments and suggestions.

REFERENCES

- [1] Berliner, H.J. A chronology of computer chess and its literature. Artif. Intell., 1978, 10, 201-214.
- [2] Brown, D.J.H., and Dowsey, S. The challenge of Go. New Sci., 1979, 81, 303-305.
- [3] Kerwin, J. and Reitman, W. A Go protocol with comments. IP-20, unpublished, Univ. Michigan, 1973.
- [4] Reitman, W. and Wilcox, B. Pattern recognition and pattern-directed inference in a program for playing Go. In D. Waterman and R. Hayes-Roth (Eds.), Pattern-directed inference systems. New York: Academic Press, 1978. Pp. 503-523.
- [5] Reitman, W., Nado, R., and Wilcox, B. Machine perception: what makes it so hard for computers to see? In C.W. Savage (Ed.), Perception and cognition. Minneapolis: Univ. of Minnesota Press, 1978. Pp. 65-87.
- [6] Reitman, W. and Wilcox, B. Modeling tactical analysis and problem solving in Go. Proc. Tenth Annual Pittsburgh Conf. on Modeling and Simulation. In press.
- [7] Wilcox, B., Reitman, D., and Reitman, W. Tutorial documentation for the INTERIM.2 Go program. IP-35, unpublished, Univ. Michigan, 1978.
- [8] Wilcox, B. Instant Go: contact fights. Am. Go J., 1978, 13 (1), 4-21.
- [9] Ryder, J.L. Heuristic analysis of large trees as generated in the game of Go. Ph.D. Dissertation, Stanford Univ., 1971.
- [10] Zobrist, A.L. Feature extraction and representation for pattern recognition and the game of Go. Ph.D. Dissertation, Univ. Wisconsin, 1970.
- [11] Mann, B. [Untitled]. Personal Comput., 1979, 3 (6), 83-86.