Relevance-Zone-Oriented Proof Search for *Connect6*

I-Chen Wu, *Member, IEEE*, and Ping-Hung Lin

***Abstract*—Wu and Huang (*Advances in Computer Games*, pp. 180–194, 2006) presented a new family of k-in-a-row games, among which *Connect6* (a kind of six-in-a-row) attracted much attention. For *Connect6* as well as the family of k-in-a-row games, this paper proposes a new threat-based proof search method, named relevance-zone-oriented proof (RZOP) search, developed from the lambda search proposed by Thomsen (*Int. Comput. Games Assoc. J.*, vol. 23, no. 4, pp. 203–217, 2000). The proposed RZOP search is a novel, general, and elegant method of constructing and promoting relevance zones. Using this method together with a proof number search, this paper solved effectively and successfully many new *Connect6* game positions, including several *Connect6* openings, especially the Mickey Mouse opening, which used to be one of the popular openings before we solved it.**

***Index Terms*—Board games, *Connect6*, k-in-a-row games, lambda search, threat-based proof search, threat-space search.**

I. Introduction

A generalized family of k-in-a-row games, named *Connect*(m,n,k,p,q) [30], [31], was introduced and presented by Wu *et al.* Two players, named *Black* and *White*, alternately place p stones on empty *squares*[1] of an $m \times n$ board in each turn. Black plays first and places q stones initially. The player who first gets k consecutive stones of his own horizontally, vertically, and diagonally wins. Both players tie the game when the board is filled up with neither player winning. Games in this family are also called *Connect* games[2] in this paper. For example, *Tic-tac-toe* is *Connect(3,3,3,1,1)*, *Go-Moku* in the free style (a traditional five-in-a-row game) is *Connect(15,15,5,1,1)*, and *Connect6* played on the traditional *Go* board is *Connect(19,19,6,2,1)*. For simplicity, let *Connect*(k,p,q) denote the game *Connect*(∞,∞,k,p,q), played on infinite boards. For example, when played on infinite boards, *Go-Moku* becomes *Connect(5,1,1)* and *Connect6* becomes *Connect(6,2,1)*.

Among these *Connect* games, *Connect6* attracted much attention due to three merits: fairness, simplicity of rules, and high game complexity as described in [30] and [31]. Since *Connect6* was introduced, hundreds of thousands of *Connect6* games have been played on web sites, such as littlegolem.net [14] and cycgame.com [21]. Since 2006, several *Connect6* open tournaments [20] for human players have been held, such as NCTU Open, ThinkNewIdea Open, Russian Open, and World Open. *Connect6* has also been included as one of the computer game tournaments at the Computer Olympiad [24] and Chinese Computer Games Contest [9], since 2006 and 2007, respectively.

For *Connect6*, researchers in [30] and [31] mentioned a simple threat-based proof search method for solving *Connect(6,2,3)*. Section II shows that many more winning positions cannot be solved by such a method. This paper proposes a new threat-based proof search method, named relevance-zone-oriented proof (RZOP) *search*, developed from the lambda search proposed by Thomsen [22]. Section IV presents this novel, general, and elegant method of constructing and promoting relevance zones for *Connect6*. The proposed method is also generalized to all *Connect* games in the Appendix. Together with a proof number search [3], [28], it solved effectively and successfully many new *Connect6* game positions, including several *Connect6* openings, especially the Mickey Mouse opening, as described in Section V. This opening used to be one of the popular openings before we solved it. All definitions and notations used in this paper are given in Section III. Concluding remarks are made in Section VI.

II. Motivation

When *Connect6* was first introduced by Wu *et al.* [30], [31], they mentioned that threats are the key to winning *Connect6* as well as other *Connect* games, like *Renju*. According to the definitions by [30] and [31], one player has t and only t threats, if and only if t is the smallest number of stones that the opponent needs to place to prevent from losing the game in the next move. A move is called a single-threat move if the player who makes the move has one and only one threat after the move, a double-threat move if two, a triple-threat move if three, and a nonthreat move if none. In *Connect6*, one player clearly wins by a triple-threat-or-more move (a move with at least three threats).

In [30] and [31], Wu *et al.* showed a type of winning strategy, called victory by continuous double-threat-or-more moves (VCDT) in this paper. It is similar to victory by continuous four (VCF), a common term for winning strategies in the *Renju* community [15]. More specifically, the type of VCDT strategy is to win by making continuously double-threat moves and ending with a triple-or-more-threat move or connecting up to six in all variations, for example, in Fig. 1, White's VCDT 12–18 (18 is a triple-threat move) moves.

Soon after the introduction of *Connect6*, many experts found another type of winning strategy in which additional single-threat moves are involved, i.e., single-threat and double-threat

Manuscript received February 11, 2010; revised June 11, 2010; accepted July 12, 2010. Date of publication July 23, 2010; date of current version September 15, 2010. This work was supported in part by the National Science Council of the Republic of China (Taiwan) under contract numbers NSC 95-2221-E-009-122-MY2 and NSC 97-2221-E-009-126-MY3.

The authors are with the Department of Computer Science, National Chiao Tung University, Hsinchu 30050, Taiwan (e-mail: icwu@csie.nctu.edu.tw; bhlin@csie.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2010.2060262

[1]Practically, stones are placed on empty intersections of *Renju* or *Go* boards. In this paper, by squares, we mean intersections.

[2]The term of connect games defined in [10] covers the games such as *Hex*, *Connect Four*, etc. In this paper, *Connect* are capitalized to indicate all the games in the family of *Connect*(m,n,k,p,q).

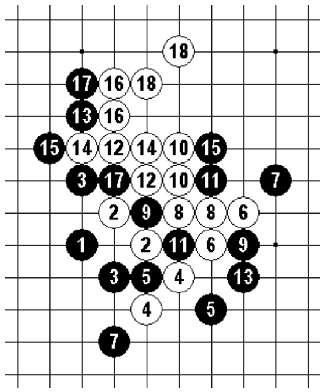

Fig. 1. Sequence of winning moves by White.

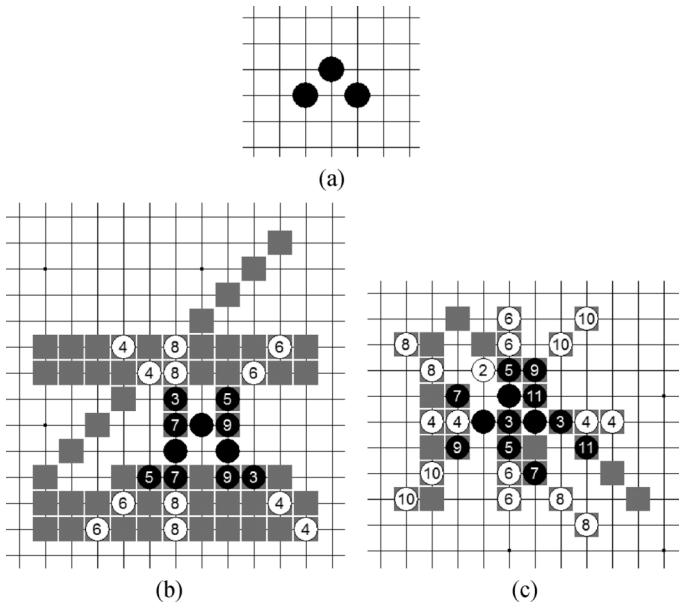

Fig. 2. (a) Black's winning move in *Connect(6,2,3)*. (b) VCDT for a null move in (a). (c) VCDT for a seminull move 2.

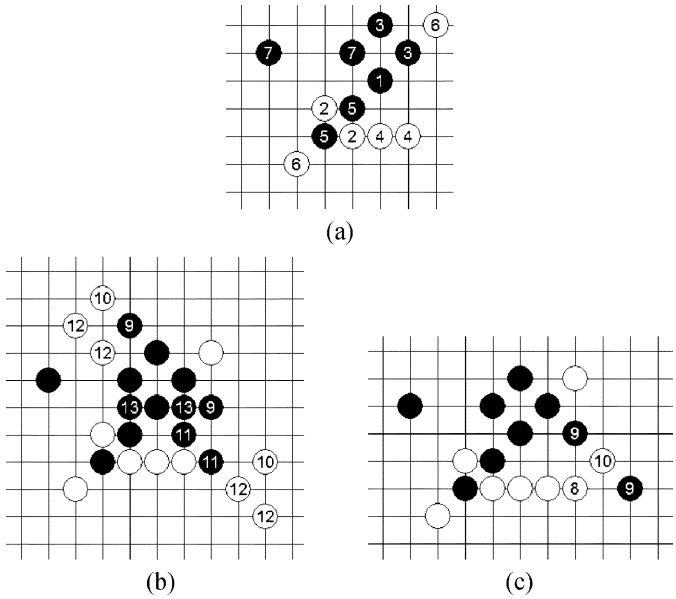

Fig. 3. (a) Position with Black winning. (b) VCDT for the null move in (a). (c) Winning single-threat move 9 for the seminull move 8.

moves are mixed (before ending with a triple-or-more-threat move). This type of winning strategy is herein called victory by continuous single-threat-or-more moves (VCST). For example, Lee [13], a *Renju* 3-dan player, found and claimed in late 2005 that White won starting from move 8 (both 8 and 10 are single-threat moves) in the game as shown in Fig. 1. Similarly, the type of winning strategy with additional non-threat moves involved is called victory by continuous nonthreat-or-more moves (VCNT).

Although VCST was unknown then, Wu *et al.* [30], [31] were already able to solve a simple VCNT case, when Black wins *Connect(6,2,3)*. This clearly is a case of VCNT, since Black's first winning move, as shown in Fig. 2(a), must be a nonthreat move. To solve it, they used a simple threat proof search method involving null or seminull moves and relevance zones, as briefly described in the following. Let White place no stones, called a null move in [30] and [31]. Obviously, Black wins by VCDT 3–9 as shown in Fig. 2(b). Then, a relevance zone Z, the area of gray squares in Fig. 2(b), can be derived to indicate that White must place at least one of the two stones inside this zone, or Black wins by simply replaying the same VCDT. Next, all squares s in Z are verified as follows. Let White place one stone on s only, called a seminull move in [30] and [31]; for example, move 2 in Fig. 2(c). Again, Black is able to win by another VCDT 3–11. Thus, another relevance zone Z', the gray area in Fig. 2(c), can be derived again to indicate that White must place another stone inside Z', or Black wins by replaying the same VCDT. Finally, all s are verified such that Black wins over all moves placed at s and s', where s' is in the Z' corresponding to the seminull move at s. Hence, Black was proved to win.

In the above search method for solving the case *Connect(6,2,3)* with VCNT, both winning strategies for the null move [3–9 in Fig. 2(b)] and the seminull move [3–11 in Fig. 2(c)] must be VCDT. However, with more and more winning *Connect6* positions investigated, we found that winning strategies for null and seminull moves may be VCSTs or even VCNTs, thus making these positions much more difficult to solve.

For example, consider the two winning nonthreat moves (proved in this paper): moves 7 in Fig. 3(a) and 6 in Fig. 4(a), respectively. The former, found in 2006 [20], was the key used to help prove that Black wins at move 3 in Fig. 3 [see also the opening in Fig. 22(a)]; that is, the opening move 2 is solved. In this case, for the null move in Fig. 3(a), Black wins by a VCDT as shown in Fig. 3(b). However, for the seminull move 8 in Fig. 3(c), Black has no double-threat moves to win by a VCDT, though Black wins by a VCST starting at 9 in Fig. 3(c).

The latter, the position in Fig. 4(a) found by Huang [11], was investigated to see whether the seminull move 5 was safe enough, since the position at 5 was popular in the following sense. Among all the first-five-move positions of *Connect6* games played by the players ranked above 1800 in [14], about 2% covered (or superset) the position according to the statistics discussed in [20]. The proof for this position is extremely complicated. Even for a null move by Black, White has no

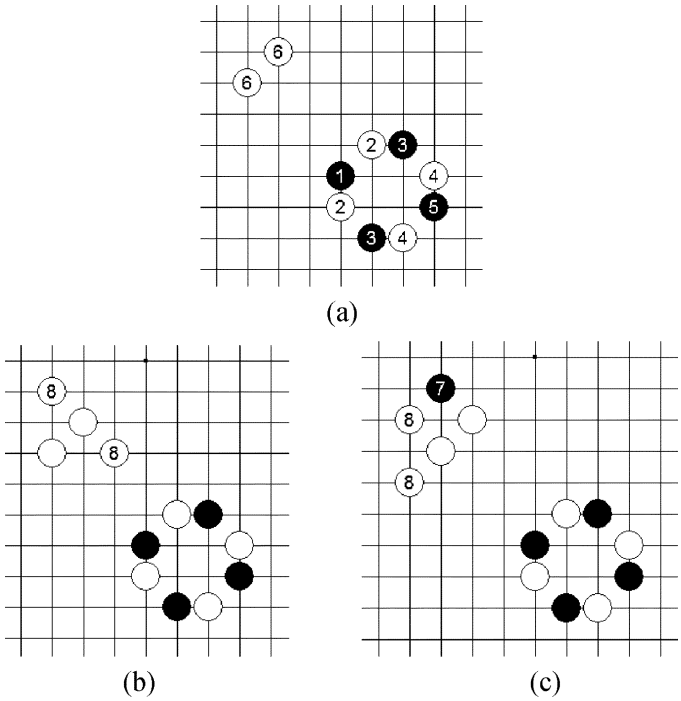

Fig. 4. (a) Position with White winning. (b) Winning single-threat move 8 for a null move in (a). (c) Winning nonthreat move 8 for a seminull move 7.

double-threat moves to win by a VCDT, but can actually win by a VCST starting at 8 as shown in Fig. 4(b). In addition, if a seminull move is made at 7 in Fig. 4(c), White cannot win by a VCDT or even a VCST, thus making the position in Fig. 4(a) much more complicated to solve.

In order to solve these as well as other positions shown in Section V, this paper proposes a new threat-based proof search method, named relevance-zone-oriented proof (RZOP) search, developed from the lambda search proposed by Thomsen [22]. In the past, many researchers [1]–[3], [6], [7], [22] have proposed threat-based search methods. Lambda search is to formalize the search trees with null moves and to solve positions of games such as *Go* and *Chess*. In lambda search, null moves are involved with different orders of threat sequences, also called lambda trees.

From the viewpoint of lambda search, a VCDT is a typical λ^1-tree with value 1 (cf., [22]). However, the definition of lambda search cannot be directly applied to *Connect6* or *Connect* games with $p \geq 2$. For *Connect* games, this paper modifies the definition of lambda search in Section III-D, and replaces the notation λ^i by Λ^i. Under the new definition, a VCST is a Λ^2-tree with value 1, the winning strategy for the position in Fig. 3(a) is a Λ^3-tree with value 1, while that in Fig. 4(a) is a Λ^4-tree with value 1. The Λ search formalized in this paper is able to solve Λ^1-trees to Λ^4-trees with value 1 for *Connect6*.

III. Definitions and Notation

This section gives definitions and notation related to *Connect* game positions, search trees, threats, lambda search, and relevance zones in Sections III-A–III-E, respectively.

A. Game Positions

In *Connect* games, a game position P includes the information of all the stones and their occupied squares on the board and the turn of whom to play. The player to be proved to win, either Black or White, is called the attacker and the other defender in this paper. Let $\sigma_A(s)$ denote the information of an attacker stone placed on the unoccupied square s, and $P+\sigma_A(s)$ denote the position after placing an attacker stone on s in position P without changing the turn. $\sigma_D(s)$ and $P+\sigma_D(s)$ are similarly defined for the defender. From the strategy stealing argument by Nash (cf., [4] and [30]), we obtain the following. If the attacker wins in P, he wins in $P+\sigma_A(s)$ as well; and if the attacker wins in $P+\sigma_D(s)$, he wins in P as well.

In this paper, $P \oplus M$ denotes the position after one player makes move M and before the other makes the next move. In *Connect6*, let $M_A(s_1, s_2)$ denote an attacker move where two attacker stones are placed on both unoccupied squares s_1 and s_2. $M_D(s_1, s_2)$ and $P \oplus M_D(s_1, s_2)$ are similarly defined for the defender. Note that in contrast to $P+\sigma_A(s_1)+\sigma_A(s_2)$, the position $P \oplus M_A(s_1, s_2)$ indicates changing the turn from the attacker to the defender.

In *Connect6*, one player, say an attacker, is allowed to make a null move, $M_{A,\phi\phi}$, that is, to place no stones; and a seminull move, $M_{A,\phi}(s_1)$, that is, to place one stone only on square s_1 in P. Thus, the position $P \oplus M_A(s_1, s_2)$ is equivalent to $(P \oplus M_{A,\phi}(s_1))+\sigma_A(s_2)$ and $(P \oplus M_{A,\phi\phi})+\sigma_A(s_1)+\sigma_A(s_2)$. From another viewpoint, null or seminull moves are to place some null stones while placing normal stones. In *Connect*(m, n, k, p, q), we place p null stones for a null move, while placing one to $p-1$ null stones for seminull moves.

In *Connect6*, a segment is defined to be a set of six consecutive squares horizontally, vertically, or diagonally on the board, while in *Connect*(m, n, k, p, q), a segment is a set of k consecutive squares. A segment is called an empty segment if all the squares on it are unoccupied yet. A segment is called an active segment of one player, if none of the squares are occupied by the opponent's stones. An active segment of one player is called a win segment of the player, if all the squares on it are occupied by the player. Obviously, one player wins if the player makes a win segment. From the definition of *Connect* games, a game ends when one makes some win segment or all the squares of the board are already occupied. According to this definition, it is impossible for both players to have win segments simultaneously.

B. Search Trees

This paper basically follows the definitions of search trees in [5] and [17]. A search tree is shown in Fig. 5(a), where rectangle and circle nodes indicate the positions in the attacker's and defender's turns,[3] respectively. The value of a leaf is 1, if the attacker makes a win segment, and 0, otherwise. The value of a search tree is the minimax value of the tree. The attacker wins in the root position if the search tree has value 1 and all the internal circles expand all defender's legal moves.

[3]When we say that a position P is in the attacker's (defender's) turn, we mean that the attacker (defender) is to move next in P.

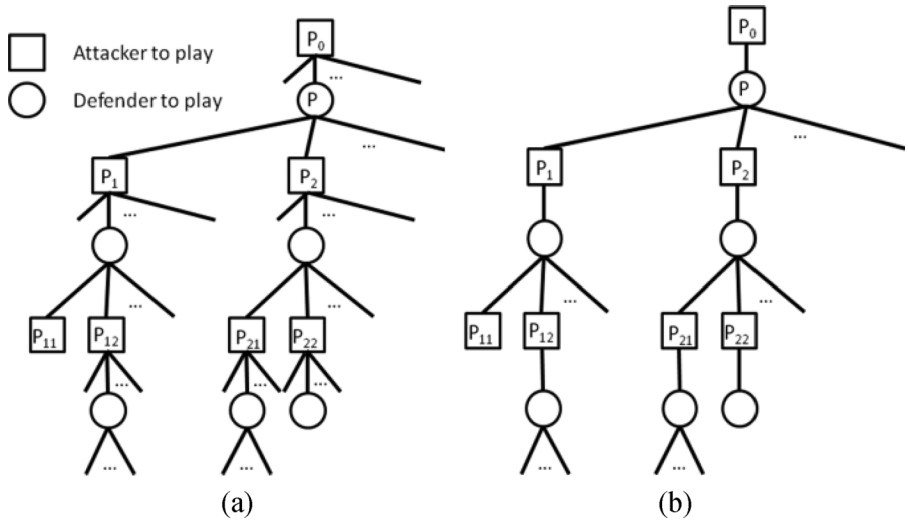

Fig. 5. (a) Search tree. (b) Solution tree.

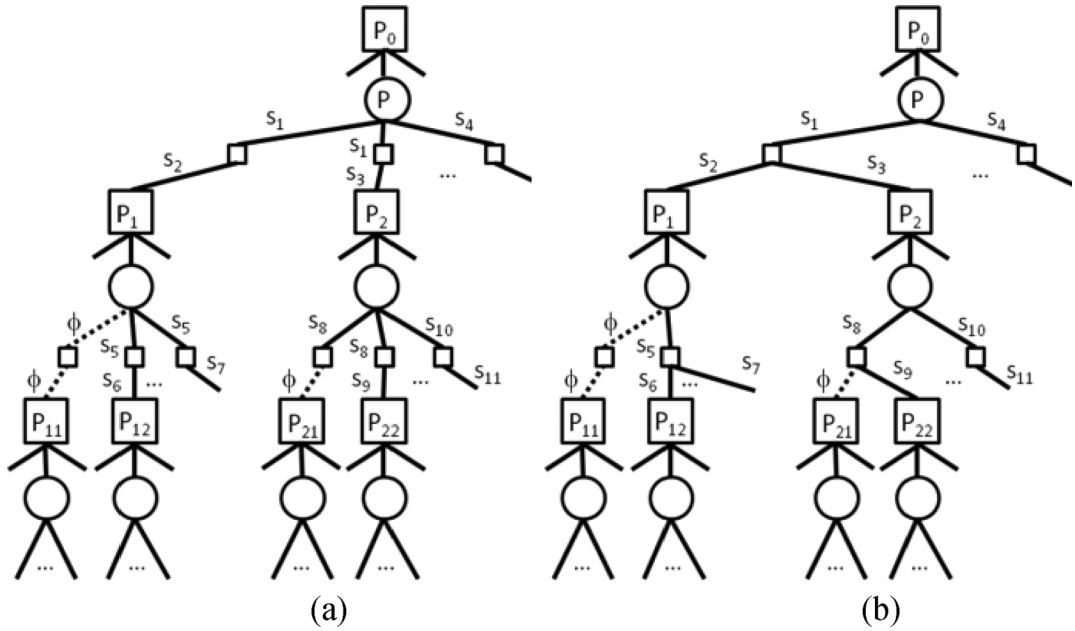

Fig. 6. (a) Marking squares of moves by inserting small boxes. (b) Combining the same edges from (a).

A strategy S of the attacker is viewed as a move-generating function of positions P that are in the attacker's turn. Namely, $S(P)$ indicates the move that the attacker chooses to make according to the strategy S. In a search tree following S, each position P expands at most one move $S(P)$. A strategy S of an attacker is called a winning strategy for position P, if and only if the value of the search tree rooted at P is 1 following S and all defender's legal moves are generated in the tree. Thus, we obtain Corollary 1. A tree as shown in Fig. 5(b) is called a solution tree in [5] and [17].

Corollary 1: The attacker wins in a position P if and only if there exists at least one winning strategy of the attacker in P. ■

In order to investigate more closely squares of defensive moves, insert small rectangles onto the corresponding edges that are broken into two, marked s_1 and s_2, respectively, as shown in Fig. 6(a). Furthermore, the edges are combined with the same s_1, as shown in Fig. 6(b). Note that null stones are marked as ϕ and the corresponding edges are indicated by dashes.

A verifier V (for the attacker) is to verify whether the attacker wins in a position P by following a strategy S. Specifically, if $V(P, S)$ returns the value 1, then the attacker wins in P and S is a winning strategy for P. A straightforward verifier is to verify it by traversing exhaustively the whole solution tree. Clearly, it is infeasible in most cases, especially in case of very large boards or even infinite boards. Fortunately, in *Connect* games, the traversal of the search tree for proof can be greatly reduced according to threats, as described in Section III-C. The traversed search tree for proof by a verifier is called a proof search tree.

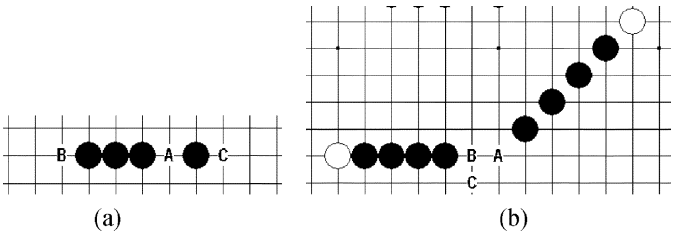

Fig. 7. (a) Normal critical defense. (b) Relaxed critical defense.

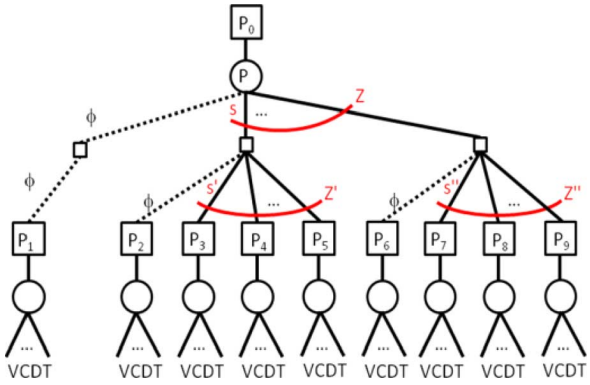

Fig. 8. Proof search tree for solving *Connect(6,2,3)*.

C. Threats

In *Connect6* (other *Connect* games are similar), threats are the key to great reduction of the proof search tree. An active segment in which the attacker occupied four or five squares is called a threat segment of the attacker. The segment poses a threat and the defender has to block it, or the attacker wins by making a win segment in the next move.

Section I has already presented the definition of threat numbers. Examples of the line patterns with one, two, and three threats can be found in [30] and [31]. The defensive moves that block all the threats are called critical defenses, while removing any stones in the moves unblocks some threats. For example, White's seminull moves $M_{D,\phi}(A)$ and moves $M_D(B,C)$ in Fig. 7(a) and (b) are critical defenses, while moves $M_D(A,B)$ are not, because the threats are still blocked without B. (Note that null moves are also critical defenses in positions without any threats according to the above definition.) Critical defenses are said to be normal if the numbers of stones in the defenses are the same as the numbers of threats; and relaxed, otherwise. For example, in Fig. 7, seminull moves $M_{D,\phi}(A)$ are normal, while moves $M_D(B,C)$ are relaxed. In *Connect6*, relaxed critical defenses are not played frequently due to their inefficiency (using two stones to block only one threat).

As described above, threats are the key to great reduction of the proof search tree without going through the entire defensive search tree. For example, for double-threat moves, there are at most four defensive moves. In addition, even for a nonthreat move such as the game *Connect(6,2,3)* described in Section I, Wu *et al.* [30], [31] were able to solve it by using a much smaller proof search tree through considering those defenses in the gray areas shown in Fig. 2. Fig. 8 shows the proof search tree [for solving *Connect(6,2,3)*] that expands $M_{D,\phi\phi}$ first, then $M_{D,\phi}(s)$ for all $s \in Z$, and $M_D(s,s')$ for all $s' \in Z'$, where Z' is the zone derived from $M_{D,\phi}(s)$.

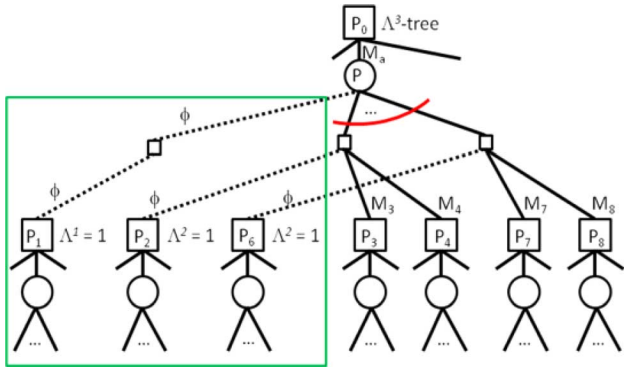

Fig. 9. A Λ^3-tree.

D. Lambda Search

In [22], Thomsen proposed using the lambda search to express how a direct attacker can achieve a goal. In *Connect* games, the goal is normally to make a win segment. The formalization of lambda search is modified for *Connect* games as follows.

Definition 1: In *Connect* games, a Λ^r-tree is a search tree which comprises all legal Λ^r-moves. If a Λ^r-move is an attacker move, the following condition holds. For all subsequent null moves or seminull moves M_D made by the defender, if M_D have exactly u null stones, where $1 \le u \le p$, there exists at least one subsequent Λ^i-tree with value 1, where $0 \le i \le r-u$ or $i = 0$ if $r < u$. If a Λ^r-move is a defender move, the following condition holds. There exist no subsequent Λ^i-trees with value 1, where $0 \le i \le r-1$. In a Λ^r-tree, a node is a leaf (without any children) if there are no Λ^r-moves following it. The value of a leaf is 1 if the defender is to move, and 0 if the attacker is to move. The value of a Λ^r-tree is either 1 (indicating that the attacker wins) or 0 (otherwise), derived using minimax calculation. The value of a Λ^0-tree (where the attacker is to move) is simply 1 if the attacker makes a win segment in the next move. ■

In case of $p = 1$, the definition of Λ^r is the same as that of λ^r (the goal is to win) in [22]; that is, a Λ^r-tree is a λ^r-tree and a Λ^r-move is a λ^r-move, and *vice versa*. In case of $p = 2$, such as *Connect6*, a Λ^3-tree is illustrated in Fig. 9 and move M_a in the tree is a Λ^3-move, since the values of Λ^1-tree and all Λ^2-trees in the left box are all 1. In addition, moves M_3, M_4, M_7, and M_8 are Λ^3-moves, if the attacker has no subsequent Λ^0-moves, Λ^1-moves, or Λ^2-moves. By following the proof of Theorem 1 in [22], we derive the following theorem (whose proof is omitted).

Theorem 1: For a Λ^r-tree rooted in a position P, if a minimax search on it returns value 1, the attacker wins in P. ■

Definition 2: A winning strategy is called a Λ^r-strategy for a position P, if the subsequent nonnull moves following the strategy are all Λ^i-moves, where $0 \le i \le r$. ■

From the above definition, a VCDT is a Λ^1-strategy, while a VCST is a Λ^2-strategy. For example, there exists a Λ^2-strategy for winning position 7 in Fig. 1 (the attacker is White), where moves 8–10 are all Λ^2-moves. VCNTs are Λ^3-strategies or strategies of higher orders, as illustrated in the following. In Fig. 2(a), move M_{623} is a Λ^3-move, and the rest of the attacker

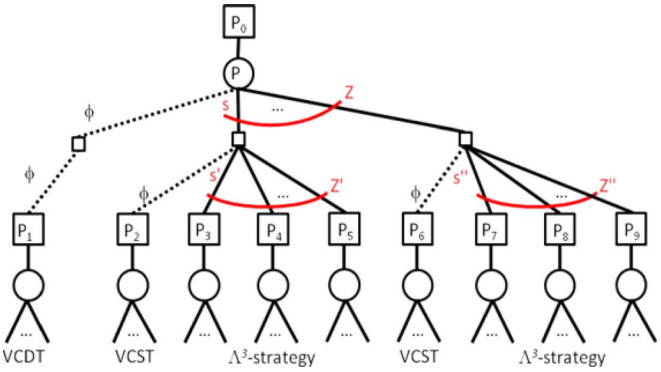

Fig. 10. A Λ^3-strategy.

moves are Λ^1-moves, so it is a Λ^3-strategy for *Connect6(6,2,3)*. In Fig. 3(a), move 7 is a Λ^3-move, and the rest of the attacker moves are Λ^1-moves or Λ^2-moves, so it is a Λ^3-strategy. Fig. 10 shows a general Λ^3-strategy. However, it is more complicated in Fig. 4(a), where move 6 is a Λ^4-move. Section V shows that it is a Λ^4-strategy.

From Definition 2, a Λ^r-strategy, $r \geq 1$, also implies that for a move with u null stones the attacker has a Λ^{r-u}-strategy. For example, in the Λ^3-strategy in Fig. 10, the attacker has a Λ^1-strategy for the null move and Λ^2-strategies for all the seminull moves.

E. Relevance Zones

As seen in Section III-E, the lambda search is a powerful method for proving the winning positions with different orders of threat sequences. The next important issue for lambda search is to construct relevance zones to reduce greatly the search space. In general, different applications construct relevance zones in different ways. In *Connect* games, it is critical to construct relevance zones in order to propagate relevance zones across different orders of threat sequences. For example, in Fig. 10, the relevance zones derived in the VCDT (Λ^1-strategy) or VCSTs (Λ^2-strategies) can be used in the whole search tree (Λ^3-strategy).

This section defines such relevance zones, which are elegantly employed to solve *Connect* games. A set of squares on the board is called a zone. A sequence of zones with size r, $\Psi = \langle Z_1, Z_2, \ldots, Z_r \rangle$, is incremental, if the condition $Z_1 \subseteq Z_2 \subseteq \cdots \subseteq Z_r$ holds. In the rest of this paper, sequences of zones with different sizes are all incremental and are thus not explicitly specified. In addition, these zones usually indicate the squares to be chosen for stones to be placed on, so only unoccupied (or empty) squares are of interest.

In a position P, its unoccupied zone, denoted by $Z_{un}(P)$, is the zone that comprises all the unoccupied squares. That is, $Z_{un}(P) = Z_{\text{board}} \backslash Z_P$, where Z_{board} is the zone for the whole board and Z_P is the set of all occupied squares in P. Let $\neg_P(Z)$ denote $Z_{un}(P) \backslash Z$ and indicate the set of unoccupied squares outside Z. Consider a sequence of zones $\Psi = \langle Z_1, Z_2, \ldots, Z_r \rangle$ in P. A sequence of unoccupied squares $\varphi = \langle s_1, s_2, \ldots, s_{r'} \rangle$, where $r' \leq r$, is said to be outside Ψ or irrelevant to Ψ, if all $s_i \notin Z_i$ or $s_i \in \neg_P(Z_i)$. Let $\varphi \in \neg_P(\Psi)$ denote the relation that φ is irrelevant to Ψ in P.

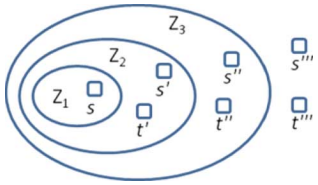

Fig. 11. Sequence of zones $\langle Z_1, Z_2, Z_3 \rangle$.

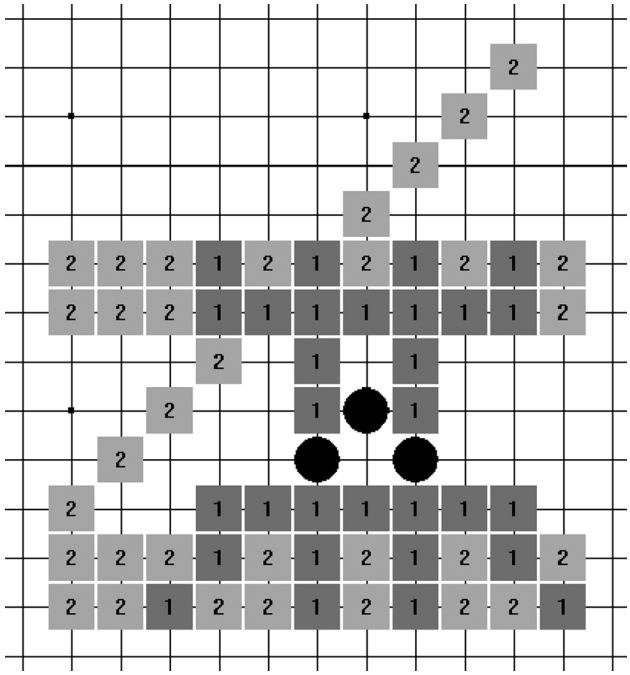

Fig. 12. Sequence of relevance zones $\Psi = \langle Z_1, Z_2 \rangle$ for the winning position in Fig. 2(a).

Implicitly, $\neg_P(\Psi)$ denotes $\langle \neg_P(Z_1), \neg_P(Z_2), \ldots, \neg_P(Z_r) \rangle$. For example, in Fig. 11, $\langle s', s'', s''' \rangle$, $\langle s', s'' \rangle$, $\langle s'', s''' \rangle$, $\langle s' \rangle$, $\langle s''' \rangle$, and even the empty sequence $\langle\rangle$ are all irrelevant to $\langle Z_1, Z_2, Z_3 \rangle$, while $\langle s \rangle$, $\langle s', t' \rangle$, $\langle s', s'', t'' \rangle$, $\langle s', s'', s''', t''' \rangle$, $\langle s'', s' \rangle$, and $\langle s, s', s'' \rangle$ are not. For simplicity, let $\sigma_A(\varphi)$ denote $\sigma_A(s_1) + \sigma_A(s_2) + \cdots + \sigma_A(s_{r'}) = \Sigma_{1 \leq i \leq r'} \sigma_A(s_i)$. Similarly, $\sigma_D(\varphi) = \Sigma_{1 \leq i \leq r'} \sigma_D(s_i)$.

Definition 3: A sequence of zones Ψ is called a sequence of relevance zones for the attacker in a position P, if and only if the attacker wins in $P + \sigma_D(\varphi)$ for all irrelevant φ; that is, $\varphi \in \neg_P(\Psi)$. Let $RZ(P)$ denote the set of all the sequences of relevance zones for the attacker in P. (Use the notation $RZ(P)$ instead of $RZ_A(P)$, since only relevance zones for the attacker are discussed in this paper). ■

From Definition 3, if $RZ(P)$ is not empty, there must exist some Ψ in $RZ(P)$. This implies that the attacker wins in P by choosing the empty sequence of squares $\langle\rangle$ for φ, since φ is irrelevant to Ψ as described above. Thus, Corollary 2 is obtained.

Corollary 2: If there exists at least one sequence of zones Ψ in $RZ(P)$, then the attacker wins in P. ■

For the winning sequence in Fig. 2(b), Fig. 12 illustrates relevance zones $\Psi = \langle Z_1, Z_2 \rangle$, where Z_1 is the set of empty squares marked with a small "1," and Z_2 marked "1" and "2." Note that in the rest of this paper, a sequence of zones is shown in this manner. Interestingly, Z_2 is the same as Z in Fig. 2(b). From

observation, Black still wins over all irrelevant $\varphi \in \neg_P(\Psi)$. That is, if White places one in $\neg_P(Z_1)$ and the other in $\neg_P(Z_2)$, Black still wins by replaying the winning sequence in Fig. 2(b). The result is slightly stronger than that in [30] and [31].

Lemma 1 shows an important property that appending extra Z_{board} to a sequence of relevance zones is still in $RZ(P)$. Note that we use Z_{board}, instead of $Z_{un}(P)$, in order to be independent of the position P, for simplicity. For example, in Fig. 12, $\langle Z_1, Z_2, Z_{\text{board}}\rangle$ is also in $RZ(P)$.

Lemma 1: Assume that $\Psi = \langle Z_1, Z_2, \ldots, Z_r\rangle$ is in $RZ(P)$. Then, $\Psi' = \langle Z_1, Z_2, \ldots, Z_r, Z_{\text{board}}\rangle$ is also in $RZ(P)$.

Proof: Consider all irrelevant $\varphi \in \neg_P(\Psi')$. For this lemma, it suffices to prove that the attacker wins in $P+\sigma_D(\varphi)$. Since $\neg_P(Z_{\text{board}})$ is empty, φ must not have the $(r+1)$th item. From the definition, we also obtain $\varphi \in \neg_P(\Psi)$. Since Ψ is assumed to be in $RZ(P)$, the attacker wins in $P+\sigma_D(\varphi)$ due to $\varphi \in \neg_P(\Psi)$. ■

From Lemma 1, two sequences of relevance zones with different sizes can be adjusted to those with the same size by appending extra Z_{board} or removing Z_{board} at the end. For simplicity of the discussion, this paper uses some more notations for operations on sequences of zones with the same size in P, say $\Psi = \langle Z_1, Z_2, \ldots, Z_r\rangle$ and $\Psi' = \langle Z'_1, Z'_2, \ldots, Z'_r\rangle$, as follows.

- Let $\Psi \subseteq \Psi'$ indicate that Ψ is contained in Ψ' pairwise; that is, $Z_i \subseteq Z'_i$ over all $1 \le i \le r$.
- Let $\Psi \cup \Psi' = \langle Z_1 \cup Z'_1, Z_2 \cup Z'_2, \ldots, Z_r \cup Z'_r\rangle$.
- Let $\Psi \cup Z = \langle Z_1 \cup Z, Z_2 \cup Z, \ldots, Z_r \cup Z\rangle$ and $\Psi \backslash Z = \langle Z_1 \backslash Z, Z_2 \backslash Z, \ldots, Z_r \backslash Z\rangle$, where Z is a zone.
- Let $\Psi \ll 1$ denote $\langle Z_2, Z_3, \ldots, Z_r, Z_{\text{board}}\rangle$ and indicate promotion of the zones in Ψ (that is, shifting zones to the left by 1) with extra Z_{board}. Similarly, let $\Psi \ll 2$ denote $(\Psi \ll 1) \ll 1$, and $\Psi \ll i$ denote $(\Psi \ll (i-1)) \ll 1$, where $i \ge 2$.

From the above notation and definitions, more properties are shown in Lemmas 2 and 3 as follows.

Lemma 2: Assume that Ψ is in $RZ(P)$ and $\Psi \subseteq \Psi'$. Then, Ψ' is also in $RZ(P)$.

Proof: Let $\Psi = \langle Z_1, Z_2, \ldots, Z_r\rangle$ and $\Psi' = \langle Z'_1, Z'_2, \ldots, Z'_r\rangle$. Consider all irrelevant $\varphi \in \neg_P(\Psi')$. It suffices to prove that the attacker wins in $P+\sigma_D(\varphi)$. Since $\Psi \subseteq \Psi'$, the condition $\varphi \in \neg_P(\Psi')$ also implies $\varphi \in \neg_P(\Psi)$. Since Ψ is in $RZ(P)$, the attacker wins in $P+\sigma_D(\varphi)$ due to $\varphi \in \neg_P(\Psi)$. ■

Lemma 3 shows important properties that are employed to improve the verifiers in Section IV.

Lemma 3: Assume that $\Psi = \langle Z_1, Z_2, \ldots, Z_r\rangle$ is in $RZ(P)$. The following two properties are satisfied.

1) Assume that $\neg_P(Z_1)$ is not empty. Let the unoccupied square be $s \in \neg_P(Z_1)$. Then, $\Psi \ll 1$ is in $RZ(P+\sigma_D(s))$.
2) Let φ be a sequence of unoccupied squares $\langle s1, s2, \ldots, sr'\rangle$ in $\neg_P(\Psi)$, where $r' \le r$. Then, $\Psi \ll r'$ is in $RZ(P+\sigma_D(\varphi))$.

Proof: It suffices to prove the first property, since the first implies the second by induction.

Let $\Psi' = \Psi \ll 1$ and consider all irrelevant $\varphi' = \langle s_2, \ldots, s_{r'}\rangle \in \neg_P(\Psi')$, where $r' \le r$. For the first property, it suffices to prove that the attacker wins in $(P+\sigma_D(s))+\sigma_D(\varphi')$.

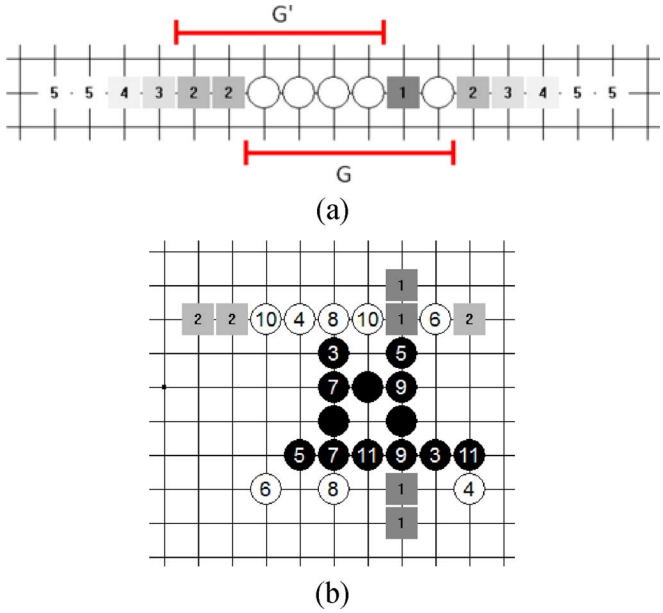

Fig. 13. Relevance zones (a) in a line and (b) in a board, upon winning with a win segment.

Let $\varphi = \langle s, s_2, \ldots, s_{r'}\rangle$. Then, the condition $\varphi \in \neg_P(\Psi)$ holds due to $s \in \neg_P(Z_1)$. Since Ψ is in $RZ(P)$, the attacker wins in $P+\sigma_D(\varphi)$ due to $\varphi \in \neg_P(\Psi)$; that is, the attacker wins in $(P+\sigma_D(s))+\sigma_D(\varphi')(= P+\sigma_D(\varphi))$. ■

IV. RZOP Search for *Connect6*

For solving positions in *Connect6*, this section investigates a verifier $V(P,S)$ that also constructs recursively a sequence of zones $\Psi(P) = \langle Z_1(P), Z_2(P), \ldots, Z_r(P)\rangle$ with the following property.

Property RZV: In the case that $V(P,S)$ returns value 1, the sequence of zones $\Psi(P)$ constructed by $V(P,S)$ is in $RZ(P)$.

This section presents such a verifier, named $V_{C6}(P,S)$, with a new proof search method for *Connect6*. This method will be generalized to all *Connect* games in the Appendix. The verifier $V_{C6}(P,S)$ is described in Sections IV-B–IV-D respectively for three distinct kinds of P, namely, endgame positions, positions in the attacker's turn, and positions in defender's turn. Finally, Section IV-E concludes with Theorem 2, showing that the verifier satisfies Property RZV in all cases.

A. Endgame Positions

If the attacker does not win in the endgame position P, the verifier simply returns the value 0. If the attacker wins in P (i.e., the attacker has a win segment in P), the verifier returns 1 and constructs $\Psi(P)$ in the following operation.

EP-1) For each active segment G of the defender containing exactly i unoccupied squares, these squares in G are all added into $Z_i(P)$ or higher order zones; that is, $Z_j(P)$ for all $j \ge i$. In other words, for each active segment G of the defender containing at most i unoccupied squares, add all of these squares in G into $Z_i(P)$.

Let us illustrate the above operation by the line shown in Fig. 13(a), where the defender is White. Following the operation, the square marked with "1" is in Z_1, those marked with

"1" or "2" are in Z_2, and so on. For example, segment G has only one unoccupied square that is in Z_1 or higher order zones, while segment G' has two unoccupied squares that are in Z_2 or higher order zones. It is observed that placing one white stone on the square in Z_1 forms a counter win segment (e.g., G) or an inversion that may prevent the attacker from winning. Note that if the defender has an inversion, this position P is unreachable since neither can have win segments simultaneously (as described in the previous section); who wins first is thus unknown. On the other hand, the attacker still wins if one white stone is placed in square s_1, where $s_1 \notin Z_1$. Similarly, the attacker still wins if one white stone is placed on s_1, where $s_1 \notin Z_1$, and the other on s_2, where $s_2 \notin Z_2$. The above can be generalized to higher orders, and to all lines (or segments) on a board. An example of constructing zones $\langle Z_1, Z_2\rangle$ on a board is illustrated in Fig. 13(b). Note that move 10 in the figure is simply one of all the defenses and is chosen for an illustration. In addition, since move 9 clearly wins already, Section IV-D will describe how to speed up the establishment of relevance zones.

From the above observation, it can be derived that the constructed $\Psi(P)$ in operation EP-1 is in $RZ(P)$. This implies that $V_{C6}(P, S)$ satisfies Property RZV in the case of endgame P, as shown in Lemma 4.

Lemma 4: Assume P to be an endgame position. Property RZV is satisfied for $V_{C6}(P, S)$.

Proof: Omitted. ■

In *Connect6*, all $Z_i(P)$ with $i \geq 6$, are nearly the same as $Z_{un}(P)$, except for those unoccupied squares covered by none of the active segments of the defender. For example, if an unoccupied square is surrounded by the attacker's squares, it is clearly covered by none of the active segments of the defender and is not included in these $Z_i(P)$. However, there are normally not many such squares, especially when board sizes are large and only a small number of stones are in positions. Practically, we simply ignore all $Z_i(P)$ with $i \geq 6$ or use $Z_{un}(P)$ whenever needed.

B. Positions in the Attacker's Turn

In such positions, the attacker simply follows strategy S to make the move $S(P)$ in P. Let P_A denote $P \oplus S(P)$. This verifier first performs $V_{C6}(P_A, S)$ recursively. If $V_{C6}(P_A, S)$ returns the value 0, this verifier $V_{C6}(P, S)$ also returns 0. On the other hand, if $V_{C6}(P_A, S)$ returns 1, this verifier $V_{C6}(P, S)$ returns 1 as well, and constructs $\Psi(P)$ in the following operation.

AT-1) Let $\Psi(P) = \Psi(P_A) \cup Z_S$, where $Z_S = \{s | s \in S(P)\}$.

Intuitively, placing any stones on the squares in Z_S by the defender in advance may block the attacks and prevent the attacker from winning. In this sense, the squares in Z_S are relevant and are therefore contained in all $Z_i(P)$ (or $\Psi(P)$).

In fact, the above operation AT-1 also implies the property $\neg_P \Psi(P) = \neg_{P_A} \Psi(P_A)$ for the following reason. From the operation, the condition $Z_i(P) = Z_i(P_A) \cup Z_S$ holds for all i. In addition, since $P_A = P \oplus S(P)$, it is clear that $Z_{un}(P_A) = Z_{un}(P) \backslash Z_S$ or $Z_{un}(P) = Z_{un}(P_A) \cup Z_S$. Thus, for all i, we derive

$$\begin{aligned}\neg_P Z_i(P) &= Z_{un}(P) \backslash Z_i(P) \\ &= (Z_{un}(P_A) \cup Z_S) \backslash (Z_i(P_A) \cup Z_S) \\ &= Z_{un}(P_A) \backslash Z_i(P_A) = \neg_{P_A} Z_i(P_A).\end{aligned}$$

From this property, Lemma 5 shows that this verifier $V_{C6}(P, S)$ satisfies Property RZV if $V_{C6}(P_A, S)$ satisfies Property RZV.

Lemma 5: Assume a position P in the attacker's turn. From the above, assume that $V_{C6}(P_A, S)$ satisfies Property RZV, where $P_A = P \oplus S(P)$. This verifier $V_{C6}(P, S)$ satisfies Property RZV.

Proof: Assume that this verifier $V_{C6}(P, S)$ returns the value 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. From the above operation, $V_{C6}(P_A, S)$ must also return 1. Since $V_{C6}(P_A, S)$ satisfies Property RZV from the lemma, $\Psi(P_A)$ is in $RZ(P_A)$.

Consider all irrelevant φ, where $\varphi \in \neg_P \Psi(P)$. It suffices to prove that the attacker wins in $P + \sigma_D(\varphi)$. Since the property $\neg_P \Psi(P) = \neg_{P_A} \Psi(P_A)$ is satisfied as described above, the condition $\varphi \in \neg_{P_A} \Psi(P_A)$ holds as well. Since $\Psi(P_A)$ is in $RZ(P_A)$ from the above, the attacker wins in $P_A + \sigma_D(\varphi)$ due to $\varphi \in \neg_{P_A} \Psi(P_A)$. Since the attacker wins in $P_A + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus S(P)$, the attacker wins in $P + \sigma_D(\varphi)$ by choosing the move $S(P)$. ■

C. Positions in the Defender's Turn

For positions in the defender's turn, Lemma 6 shows a very important property used in this section as well as in the Appendix.

Lemma 6: Assume a position P in the defender's turn. For a given sequence of zones Ψ, assume that for all defender moves M_D there exists some Ψ_D such that $\Psi_D \subseteq \Psi$ and Ψ_D is in $RZ(P \oplus M_D)$. Then, Ψ is in $RZ(P)$.

Proof: Consider all irrelevant $\varphi \in \neg_P \Psi$. For this lemma, it suffices to prove that the attacker wins in $P + \sigma_D(\varphi)$.

Now, consider all defender moves M_D in $P + \sigma_D(\varphi)$. From this lemma, there exists some Ψ_D such that $\Psi_D \subseteq \Psi$ and Ψ_D is in $RZ(P \oplus M_D)$. Since $\Psi_D \subseteq \Psi$, the condition $\varphi \in \neg_P \Psi$ implies $\varphi \in \neg_P \Psi_D$. Since squares in M_D and $\sigma_D(\varphi)$ are mutually exclusive, $\varphi \in \neg_P \Psi_D$ also implies $\varphi \in \neg_{P \oplus M_D} \Psi_D$. Since Ψ_D is in $RZ(P \oplus M_D)$ from the above, the attacker wins in $(P \oplus M_D) + \sigma_D(\varphi)$ due to $\varphi \in \neg_{P \oplus M_D} \Psi_D$. Since $(P \oplus M_D) + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus M_D$, the attacker also wins in $(P + \sigma_D(\varphi)) \oplus M_D$. From the above, since the attacker wins in $(P + \sigma_D(\varphi)) \oplus M_D$ over all defender moves M_D, the attacker wins in $P + \sigma_D(\varphi)$. ■

A straightforward verifier is to verify whether the attacker wins for all defender moves, as follows. The verifier $V_{C6}(P, S)$ returns value 1, if the recursive $V_{C6}(P \oplus M_D, S)$ returns 1 for all defender moves M_D; otherwise, it returns 0. In the case that this verifier $V_{C6}(P, S)$ returns 1, the zones $\Psi(P)$ are constructed in the following operation.

DT-1) Initialize all zones in $\Psi(P)$ to be empty. Then, for all defender moves M_D, let $\Psi(P) = \Psi(P) \cup \Psi(P \oplus M_D)$.

From the above operation, the condition $\Psi(P \oplus M_D) \subseteq \Psi(P)$ clearly holds for all M_D. Assume that all the recursive $V_{C6}(P \oplus M_D, S)$ satisfy Property RZV. Then, all $\Psi(P \oplus M_D)$ are in $RZ(P \oplus M_D)$ for all defender moves M_D. From Lemma 6, we obtain that $\Psi(P)$ is in $RZ(P)$; and therefore, the verifier satisfies Property RZV. By induction, the above straightforward verifier satisfies Property RZV in all cases.

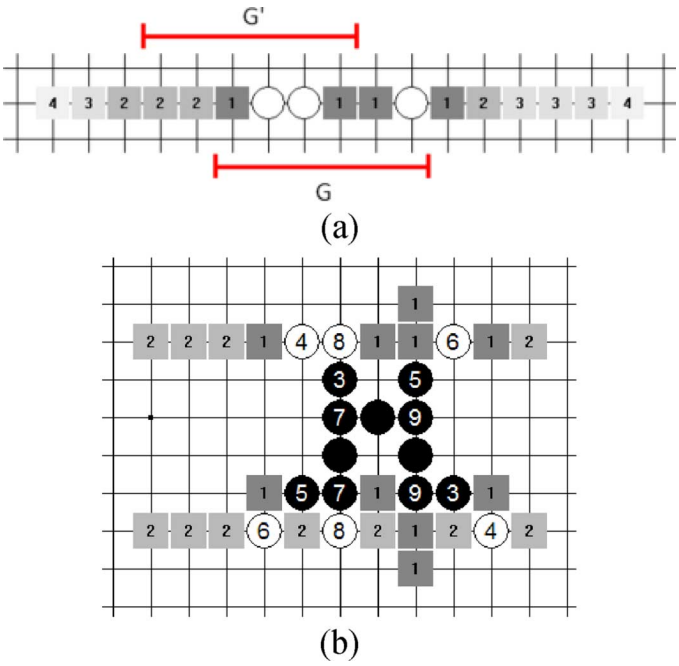

Fig. 14. Relevance zones (a) in a line and (b) in a board, upon winning with three or more threats.

However, the above straightforward verifier is apparently inefficient, since it searches exhaustively all defender moves, even when the attacker moves have some threats. The situation is even worse in the case that the board size is very large or infinite. In this section, an efficient and elegant verifier is devised to reduce the search space by making use of both threats and relevance zones. In *Connect6*, the position P (in the defender's turn) can be classified into the following four cases. The number of the attacker threats in P is 1) three or more, 2) two, 3) one, and 4) zero. The four cases are discussed, respectively, in the following.

1) Three Threats or More: In this case, the attacker is sure to win by simply following the strategy S_{3T} as follows. For each defender move, since the move must leave some threat segments unblocked, the attacker wins simply by making a win segment from the unblocked one. Since the strategy is a sure win, the verifier returns value 1 and constructs the zones (initialized to be empty) in the following operations.

T3-1) Add all unoccupied squares s on threat segments into all $Z_i(P)$.

T3-2) For each active segment G of the defender containing exactly $i+2$ unoccupied squares, all these squares in G are added into all $Z_j(P)$ or higher order zones. In other words, for each active segment G of the defender containing at most $i+2$ unoccupied squares, add all these squares in G into $Z_i(P)$.

Let us illustrate the above operations by the line shown in Fig. 14(a), where the defender is White. Zones in the line are marked in a way similar to that in Fig. 13(a). It is observed that placing one white stone in G or Z_1 results in a counter threat segment or an inversion that may threaten the attacker to defend in some of his earlier moves and prevent the attacker from winning. On the other hand, the attacker still wins if one white stone is placed on other squares s_1, where $s_1 \notin Z_1$. Similarly, the attacker still wins if one white stone is placed on s_1, where $s_1 \notin Z_1$, and the other on s_2, where $s_2 \notin Z_2$. The above can be generalized to higher orders, and to all lines (or segments) on the board. An example of constructing two zones $\langle Z_1, Z_2 \rangle$ on a board is illustrated in Fig. 14(b). Lemma 7 shows that in this case the verifier satisfies Property RZV; that is, $\Psi(P)$ is in $RZ(P)$.

Lemma 7: Assume that the defender is to move and the attacker has three or more threats in P. The verifier described above satisfies Property RZV.

Proof: For this lemma, it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. Consider all defender moves M_D. The attacker simply follows a strategy S_{3T} to connect six from an unblocked threat segment. Let $P_D = P \oplus M_D$ and $P_6 = P_D \oplus S_{3T}(P_D)$. From Lemmas 4 and 5, $\Psi(P_6)$ and $\Psi(P_D)$ are in $RZ(P_6)$ and $RZ(P_D)$, respectively.

To prove that $\Psi(P)$ is in $RZ(P)$, it suffices to prove from Lemma 6 that $\Psi(P_D) \subseteq \Psi(P)$, since $\Psi(P_D)$ is already in $RZ(P_D)$. From Section IV-C, $\Psi(P_D) = \Psi(P_6) \cup Z_S$, where $Z_S = \{s | s \in S_{3T}(P_D)\}$. From operation T3-1, all squares in Z_S are added into $\Psi(P)$. Thus, it suffices to prove that $\Psi(P_6) \subseteq \Psi(P)$.

Since the attacker connects six in P_6, operation EP-1 (in Section IV-B) is employed to construct zones $\Psi(P_6)$. The operation is restated as follows. For each active segment G of the defender containing at most i unoccupied squares in P_6, all the squares in G are added into $Z_i(P_6)$. Since one move has at most two squares, at most two occupied squares in G were occupied by move M_D. Therefore, G contains at most $2+i$ unoccupied squares back in P (before making move M_D). From operation T3-2, all these unoccupied squares are also added into $Z_i(P)$. For example, let both lines in Figs. 13(a) and 14(a) be, respectively, in positions P_6 and P, where move M_D is placed on the two leftmost squares marked "1" in segment G in Fig. 14(a). Thus, the two squares marked "2" in segment G' in Fig. 13(a) are also added into $Z_2(P)$ in Fig. 14(a). From the above observation, we can derive $\Psi(P_6) \subseteq \Psi(P)$. ■

Since all active segments G of the defender contain at most $6 (= 4+2)$ unoccupied squares in *Connect6*, all these squares in G are added into all $Z_i(P)$ from operation T3-2, where $i \geq 4$. Thus, these $Z_i(P)$ are nearly the same as $Z_{un}(P)$, except for the unoccupied squares not covered by any active segments of the defender, e.g., the unoccupied squares surrounded by all the attacker squares. Similar to the argument in Section IV-C, we construct zones with size three, and simply use $Z_{un}(P)$ for those higher order zones, whenever needed.

2) Two Threats: When the attacker has two threats in P, the defender must defend by blocking the two threats. In this case, the verifier performs the following operations.

T2-1) For each defender move M_D that blocks the two threats, perform the following.
   a) Return value 0 if the recursive $V_{C6}(P_D, S)$ returns value 0, where $P_D = P \oplus M_D$.
   b) Let $\Psi(P) = \Psi(P) \cup \Psi(P_D)$.

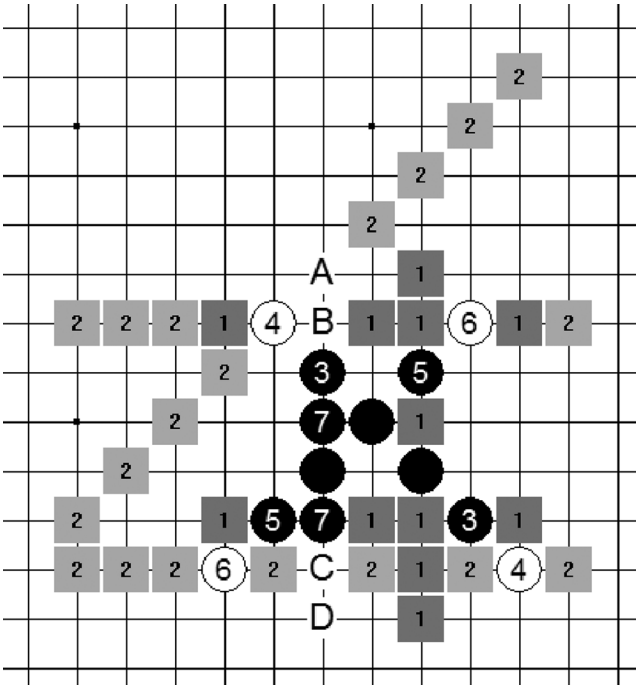

Fig. 15. Winning position with two threats for Black (the attacker) and the constructed $\Psi(P)$.

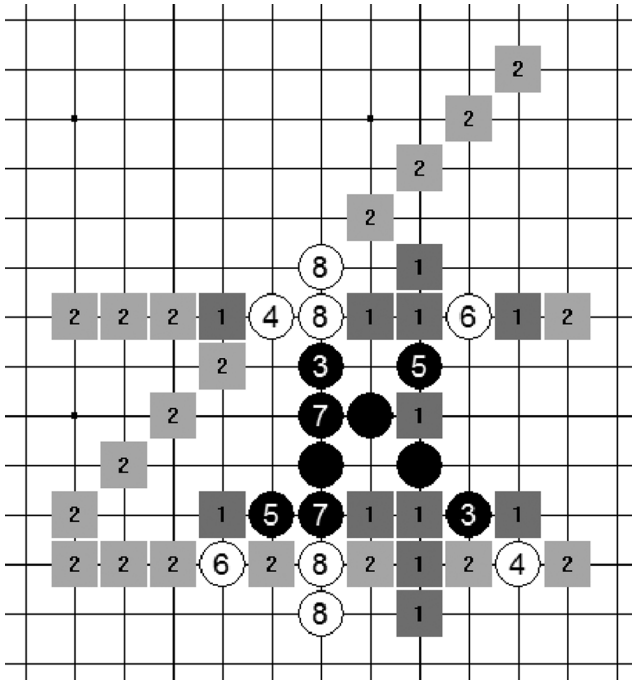

Fig. 16. Combining three defensive moves into one with four stones.

T2-2) Continue to construct zones by both operations T3-1 and T3-2, and return 1.

For example, for position P in Fig. 15 [the grandparent of the position in Fig. 14(b)] where Black has two threats, White has three defensive moves at (B,C), (A,C), and (B,D). Obviously, since Black still wins for each of the three moves, Black wins in P. From the above operations, this verifier returns value 1 and constructs $\Psi(P)$ as shown in Fig. 15. Lemma 8 shows that this verifier satisfies Property RZV if the verifier satisfies Property RZV for all the defensive moves as well. From this lemma, $\Psi(P)$ in Fig. 15 is in $RZ(P)$.

Lemma 8: From the above, assume that the defender is to move and the attacker has two threats in P. Assume that all the recursive $V_{C6}(P_D, S)$ in operation T2-1 satisfy Property RZV. Then, the verifier $V_{C6}(P, S)$ satisfies Property RZV as well.

Proof: Assume that this verifier $V_{C6}(P, S)$ returns 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. Since $V_{C6}(P, S)$ returns 1, all the recursive $V_{C6}(P_D, S)$ in operation T2-1 must return 1. Since these $V_{C6}(P_D, S)$ satisfy Property RZV from this lemma, all constructed $\Psi(P_D)$ are in $RZ(P_D)$.

To prove $\Psi(P) \in RZ(P)$, it suffices to prove from Lemma 6 the following. For all defender moves M_D, there exists some Ψ_D such that Ψ_D is in $RZ(P \oplus M_D)$ and $\Psi_D \subseteq \Psi(P)$. All defender moves M_D are classified into the following cases.

1) All defender moves M_D that block both threats. From the above, $\Psi(P_D)$ are in $RZ(P_D)$. In addition, since these $\Psi(P_D)$ are merged into $\Psi(P)$ in operation T2-1b, we obtain $\Psi(P_D) \subseteq \Psi(P)$. Thus, $\Psi(P_D)$ is the Ψ_D.
2) All defender moves M_D that leave some threat segment unblocked. The attacker wins by connecting six on the segment, like strategy S_{3T}. Since operation T2-2 follows those steps in T3-1 and T3-2, we simply follow the proof of Lemma 7 to prove that there exists some Ψ_D such that $\Psi_D \subseteq \Psi(P)$ and Ψ_D is in $RZ(P_D)$. ■

Assume that the subsequent winning moves of the attacker are the same for all the defensive moves. Then, we can optimize the construction of zones by combining these defensive moves together. For example, in Fig. 15, the three defensive moves, (B,C), (A,C), and (B,D), can be combined into a macromove (A, B, C, D) as shown in Fig. 16. Since the subsequent winning sequences of the attacker are the same, the sizes of relevance zones are relatively smaller and the threat-based search is also greatly reduced. However, note that the segment containing both A and B (the same for C and D) in Fig. 15 should be considered to have one white stone only for zone construction. Since the winning sequences in Fig. 2(b) are the same for all defensive moves, the relevance zones are constructed as shown in Fig. 12.

3) One Threat: When the attacker has one threat, the defender must defend by blocking the threat. In this case, the verifier performs the following operations.

T1-1) For each normal critical defense (defined in Section III-C), $M_{D,\phi}(s)$, where square s blocks the threat, perform the operation of seminull-move proof search as follows.
   a) Return value 0, if the recursive $V_{C6}(P_s, S)$ returns 0 where $P_s = P \oplus M_{D,\phi}(s)$.
   b) Let $\Psi(P) = \Psi(P) \cup (\Psi(Ps) \ll 1)$.
   c) For each defensive move $M_D(s, s')$, where $s' \in Z_1(P_s)$, perform both operations T2-1a and T2-1b.

T1-2) For all relaxed critical defenses $M_D(s, s')$, perform both operations T2-1a and T2-1b.

T1-3) Perform both operations T3-1 and T3-2, and return 1.

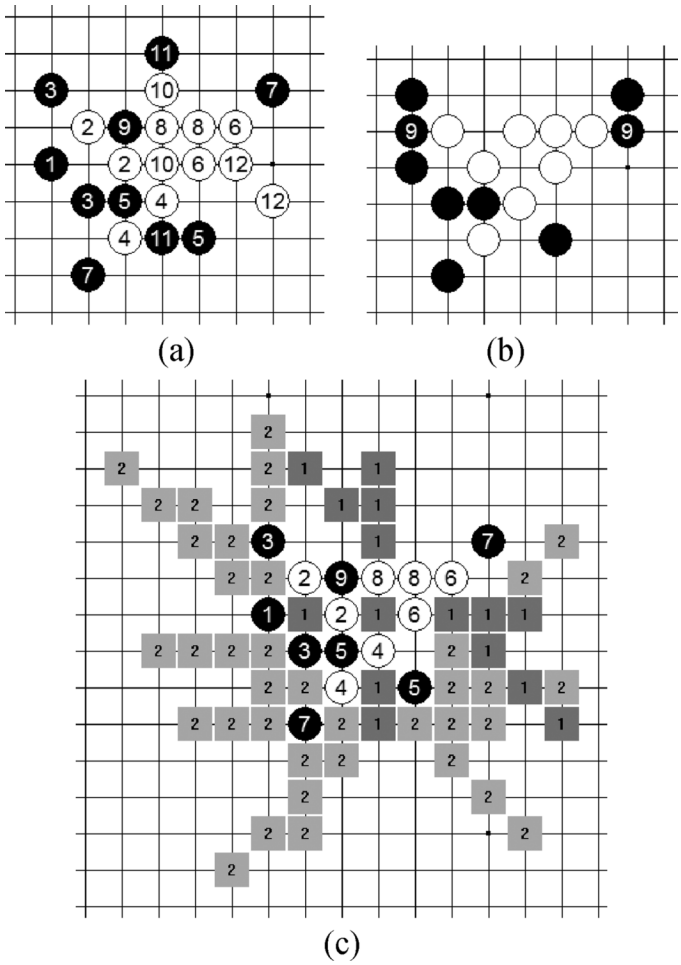

Fig. 17. (a) VCDT for the seminull move 9. (b) Relaxed critical defense at 9. (c) Constructed zones for the seminull move 9 in (a).

Consider a position P, 8 in Fig. 17(a) (the same as 8 in Fig. 1), and another P_s, with a seminull move added at 9. White (the attacker) wins in P_s by the winning sequence in Fig. 17(a). The above operations construct the zones $\Psi(P_s) = \langle Z_1(P_s), Z_2(P_s), Z_3(P_s)\rangle$, with the first two zones shown in Fig. 17(c). According to operation T1-1b, both zones $Z_2(P_s)$ and $Z_3(P_s)$ are shifted and merged into $Z_1(P)$ and $Z_2(P)$, respectively. For all defensive moves $M_D(s, s')$, where $s' \in Z_1(P_s)$, operation T1-1c follows both T2-1a and T2-1b to construct zones and verify whether $V_{C6}(P \oplus M_D(s, s'), S)$ return 1. In addition, operation T1-2 also performs the same for all relaxed critical defenses, such as the one in Fig. 17(b). From Fig. 17(c), since the number of squares in $Z_1(P_s)$ is only 15, the number of recursive V_{C6} is relatively small, even in very large or infinite boards.

Lemma 9 shows that the verifier satisfies Property RZV if all the recursive V_{C6} satisfy Property RZV.

Lemma 9: From the above, assume that the defender is to move and the attacker has one threat in P. Assume that all the recursive V_{C6} in both operations T1-1 and T1-2 satisfy Property RZV. Then, the verifier $V_{C6}(P, S)$ satisfies Property RZV as well.

Proof: Assume that this verifier $V_{C6}(P, S)$ returns 1. For this lemma, it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. Since $V_{C6}(P, S)$ returns 1, all the recursive V_{C6} in both operations T1-1 and T1-2 must also return 1. Since all the recursive V_{C6} satisfy Property RZV from this lemma, all $\Psi(P_s)$ constructed from T1-1a are in $RZ(P_s)$ and all $\Psi(P_D)$ from T1-1c and T1-2 are in $RZ(P_D)$.

To prove $\Psi(P) \in RZ(P)$, it suffices to prove from Lemma 6 the following. For all defender moves M_D, there exists some Ψ_D such that Ψ_D is in $RZ(P \oplus M_D)$ and $\Psi_D \subseteq \Psi(P)$. All defender moves M_D are classified into the following cases.

1) All defender moves $M_D(s, s')$ where s blocks the threat as described in T1-1. Let $P_s = P \oplus M_{D,\phi}(s)$. Furthermore, this case is separated into the following two subcases.
   a) $s' \in Z(P_s)$. Let P_D denote $P \oplus M_D(s, s')$. The zone $\Psi(P_D)$ is constructed in operation T1-1c, and is in $RZ(P_D)$ according to the first paragraph of this proof. Since $\Psi(P_D)$ is merged into $\Psi(P)$ in T1-1c, we obtain $\Psi(P_D) \subseteq \Psi(P)$. Thus, $\Psi(P_D)$ is the ΨD.
   b) $s' \in \neg_{P_s}(Z_1(P_s))$. From the above, $\Psi(P_s)$ is in $RZ(P_s)$. Since $s' \in \neg_{P_s}(Z_1(P_s))$, Lemma 3 shows that $\Psi(P_s) \ll 1$ is in $RZ(P_s + \sigma_D(s'))$, meaning $RZ(P \oplus MD(s, s'))$. From operation T1-1b, $(\Psi(P_s) \ll 1) \subseteq \Psi(P)$. Thus, $\Psi(P_s) \ll 1$ is Ψ_D.
2) All defender moves $M_D(s, s')$ in operation T1-2 are relaxed critical defenses. The proof is similar to that in case 1a and therefore omitted.
3) All defender moves $M_D(s, s')$ that do not block the threat. The attacker wins by connecting six on some unblocked threat segments, like strategy S_{3T}. Find Ψ_D by following the proof of Lemma 7. ■

4) No Threats: When the attacker has no threats, it becomes more complicated since the defender has much more freedom to move. In this case, the verifier makes use of the constructed relevance zones to minimize the search space in the following operations.

T0-1) Return value 0 if $V_{C6}(P_\phi, S)$ returns 0, where $P_\phi = P \oplus M_{D,\phi\phi}$.
T0-2) Let $\Psi(P) = \Psi(P_\phi) \ll 2$.
T0-3) For each square s in $Z_2(P_\phi)$, perform the seminull move proof search, as in operations T1-1a to T1-1c.
T0-4) Return 1.

Let us illustrate the above operations by the example in Fig. 2. From the winning moves in Fig. 2(b), operation T0-1 constructs relevance zones $\Psi(P_\phi) = \langle Z_1(P_\phi), Z_2(P_\phi), Z_3(P_\phi)\rangle$, with only the first two zones shown in Fig. 12. Similarly, zone $Z_2(P_\phi)$ is the same as Z in Fig. 2(b). According to operation T0-2, zone $Z_3(P_\phi)$ is shifted and merged into $Z_1(P)$. Then, in operation T0-3, one square s in $Z_2(P_\phi)$ is chosen to perform the seminull move proof search. In the case that 2 in Fig. 2(c) is chosen, the seminull move proof search in T0-3 constructs the relevance zones $\Psi(P_s) = \langle Z_1(P_s), Z_2(P_s), Z_3(P_s)\rangle$, where $P_s = P \oplus M_{D,\phi}(s)$. Zone $Z_1(P_s)$ is actually the same as Z' in Fig. 2(c). After verifying that White wins for all $s \in Z_2(P_\phi)$ and all $s' \in Z_1(P_s)$, the verifier confirms that White wins in P, as shown in Lemma 10. For the position in Fig. 2, the number of the recursive V_{C6} in T0-1–T0-3 is 2656, relatively small when compared with the number of legal moves.

Lemma 10: Assume that the defender is to move and the attacker has no threats in P. From the above, assume that all recursive V_{C6} in both operations T0-1 and T0-3 satisfy Property RZV. Then, the verifier $V_{C6}(P, S)$ also satisfies Property RZV.

Proof: Assume that this verifier $V_{C6}(P, S)$ returns 1. For this lemma, it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. Since $V_{C6}(P, S)$ returns 1, all the recursive V_{C6} in both operations T0-1 and T0-3 must also return 1. Since these recursive V_{C6}, say for position P', satisfy Property RZV from this lemma, the constructed zones $\Psi(P')$ are in $RZ(P')$.

To prove $\Psi(P) \in RZ(P)$, it suffices to prove from Lemma 6 the following: for all defender moves M_D, there exists some Ψ_D such that Ψ_D is in $RZ(P \oplus M_D)$ and $\Psi_D \subseteq \Psi(P)$. All defender moves M_D are classified into the following cases.

1) All defender moves $M_D(s, s')$ where $s \in \neg_{P_\phi}(Z_2(P_\phi))$ and $s' \in \neg_{P_\phi}(Z_2(P_\phi))$. From the first paragraph in this proof, $\Psi(P_\phi)$ is in $RZ(P_\phi)$. Since $s \in \neg_{P_\phi}(Z_2(P_\phi))$ and $s' \in \neg_{P_\phi}(Z_2(P_\phi))$, $\Psi(P_\phi) \ll 2$ is in $RZ(P_\phi + \sigma_D(s) + \sigma_D(s'))$ from Lemma 3. Since $P\phi + \sigma_D(s) + \sigma_D(s') = P \oplus M_D(s, s')$, $\Psi(P_\phi) \ll 2$ is also in $RZ(P \oplus M_D(s, s'))$. In addition, $(\Psi(P_\phi) \ll 2) \subseteq \Psi(P)$ from operation T0-2. Thus, $\Psi(P_\phi) \ll 2$ is Ψ_D.
2) All defender moves $M_D(s, s')$ where $s \in Z_2(P_\phi)$. By following the proof for case 1 (including subcases 1a and 1b) in Lemma 9, we obtain that there exists some Ψ in $P \oplus M_D(s, s')$ for all s' such that $\Psi \subseteq \Psi(P)$. The details are omitted. ■

D. Conclusion

Theorem 2 concludes that the verifier $V_{C6}(P, S)$ in all cases satisfies Property RZV. Therefore, if $V_{C6}(P, S)$ returns value 1, the constructed $\Psi(P)$ is in $\mathrm{RZ}(P)$, and the attacker wins in P from Corollary 2.

Theorem 2: The verifier $V_{C6}(P, S)$ satisfies Property RZV in all cases.

Proof: By induction, the verifier $V_{C6}(P, S)$ satisfies Property RZV in all cases from Lemma 4 to Lemma 10. ■

V. Solving *Connect6* Positions

In Section IV, we present a verifier $V_{C6}(P, S)$ to verify whether the attacker wins in a *Connect6* position P by following strategy S. However, in order to solve positions, we still need to provide the verifier with winning strategies S. Winning strategies can be provided in the following three ways.

1) Let human experts offer the winning strategies manually.
2) Let programs find the winning strategies automatically.
3) Find the winning strategies by mixing the above two.

Traditionally, experts used the first way to claim that some positions are winning, e.g., *Go-Moku* and *Renju* [18]. However, it becomes complicated and tedious for human players to traverse all positions to prove it thoroughly. Hence, it is more feasible to solve these positions by programs using the second way. However, programs may not be smart enough sometimes to find the correct winning moves. Therefore, some researchers chose the third way by following experts' suggestions for some opening moves and then letting programs solve the subsequent moves. For example, Allis [1], [2] solved *Go-Moku* in the free style, and Wágner and Virág [23] solved *Renju*. In Section V-A, we developed some assistant programs to help find the winning strategies for *Connect6*. In Section V-B, we illustrate our new proof search method in Section IV by solving the positions in Figs. 3(a) and 4(a). Finally, we give more results in Section V-C.

A. Assistant Programs

This section describes some assistant programs developed for solvers and verifiers. Given a position P in the attacker's turn, a solver is to return a winning move as well as the relevance zones, if found; and, otherwise, a null move is returned to indicate failure of finding a winning move. A solver of finding a VCDT strategy, denoted by S_{VCDT}, is described as follows.

1) If there exist connect-six moves or triple-threat-or-higher moves, simply choose one of them to win.
2) Evaluate all the double-threat moves and choose some *good* ones for further expansion (according to the evaluations).
3) For each chosen move M, return M if $V_{C6}(P \oplus M, S_{\mathrm{VCDT}})$ returns 1.
4) Return the null move to indicate failure of finding a winning move.

A solver of finding a VCST (VCNT) is similar to the above, except that single-threat (nonthreat) moves are also evaluated and chosen at step 2. Actual solvers are implemented in a more complicated way to reduce the size of a search tree and control the timing. For example, the techniques of iterative deepening and transposition table are normally incorporated.

In this paper, we implemented a solver with VCDT, named VCDT-Solver, and another solver with VCST, named VCST-Solver. More accurately, the VCDT-Solver is to find a Λ^1-strategy, while the VCST-Solver is to find a Λ^2-strategy. Our VCST-Solver also tends to find VCDTs, if any, unless some single-threat moves are evaluated to be much better. Currently, this solver is able to find a Λ^2-strategy up to depth 25 where the size of the longest path with Λ^2-moves is 13. This solver was also incorporated into our *Connect6* program NCTU6, which won the gold at the 11th and 13th Computer Olympiads [26], [33] in 2006 and 2008, respectively; and also won eight games and lost none against top *Connect6* players in Taiwan in 2009 [12]. From our experience, VCST-Solver is able to find Λ^2-strategies, if any, in most cases accurately.

Regarding solvers for Λ^3-strategies or strategies of higher orders, the time complexities become much higher, since the numbers of defensive moves to be verified grow much higher. Therefore, we did not implement it directly.

First, we implemented a verifier, named NCTU6-Verifier, to verify whether the attacker wins for all defender moves. In other words, given a position P in the defender's turn as shown in Fig. 18(a), the verifier uses VCDT-Solver for null moves and VCST-Solver for all seminull moves and nonnull moves. If null and seminull moves are all solved, then move M (from the parent of P to P) in Fig. 18(a) is an attacker Λ^3-move. If some nonnull moves are not solved by VCST-Solver, these moves are reported or generated. Note that the defender Λ^3-moves must be reported. Since our VCST-Solver can find Λ^2-strategies accurately in most cases, most reported moves are the defender Λ^3-moves in our experiments.

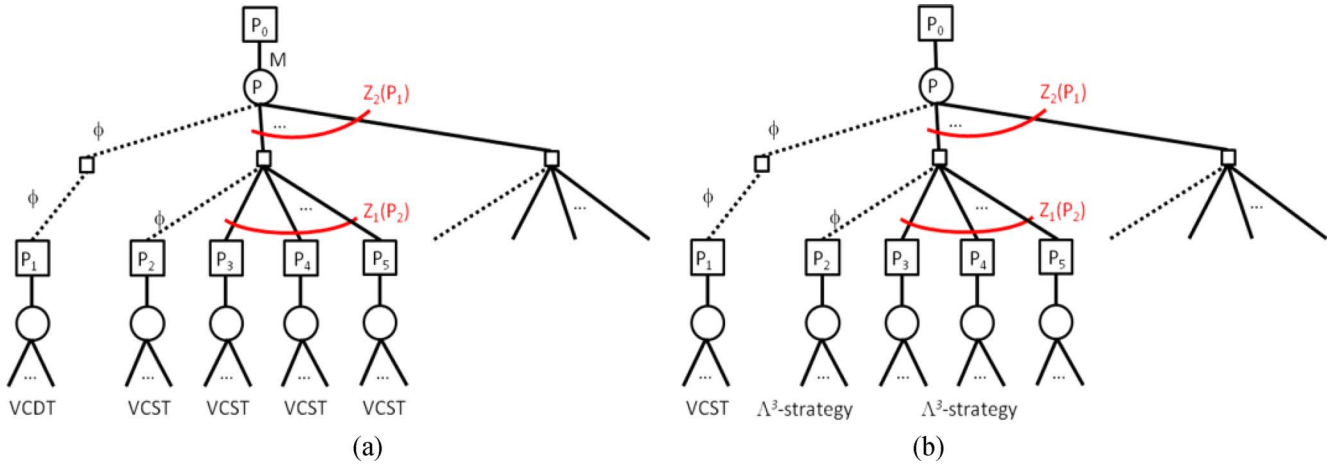

Fig. 18. Proof search tree of (a) NCTU6-Verifier and (b) the verifier of one higher order.

When our *Connect6* program NCTU6 mentioned above cannot find Λ^2-strategies (VCSTs), NCTU6 then chooses some promising moves including nonthreat moves using heuristic evaluations. The details of heuristic evaluations are beyond the scope of this paper and therefore omitted.

Since NCTU6 may not be able to find winning moves all the time, experts are allowed to help find winning moves. (As [1], [2], and [23], expert knowledge was utilized to help solve *Go-Moku* and *Renju*.) Hence, the above programs, such as NCTU6 and NCTU6-Verifier, were integrated into a *Connect6* editor named Connect6Lib [8], modified from Renlib [16], in order to accommodate hints from human experts. In the integrated system [25], [32], the users (human experts) are allowed to suggest some attacker moves directly or let NCTU6 suggest possibly good moves in a designated position. Then, for suggested moves, users invoke NCTU6-Verifier to verify and report all the defensive moves (most are Λ^3-moves). Then, users repeat the above for the subsequent moves, until a Λ^3-strategy is found.

Second, for Λ^4-strategies, the integrated system (on top of the editor Connect6Lib) needs to maintain a global verifier and modify the search by incrementing the order by one as shown in Fig. 18(b).

B. Illustration of Solving Positions

In this section, we illustrate the proof search method in Section IV by solving the two positions in Figs. 3(a) and 4(a). First, consider the one in Fig. 3(a). The position is solved by simply running NCTU6-Verifier. In the proof search tree shown in Fig. 19, P indicates the position at 7 in Fig. 3(a); P_0, the position at 6; P_1, the position after a null move; P_2, the position after the seminull move 8 in Fig. 3(c); and P_{21}, the position after another seminull move at 10 in Fig. 3(c). As can be seen, the attacker wins in a Λ^3-strategy.

Second, consider the position in Fig. 4(a), which is much more complicated than the previous one. This position is solved via the integrated system supporting Λ^4-trees, as described in Section V-A. In the proof search tree shown in Fig. 20, P indicates this position, P_1 does the position after a null move, and

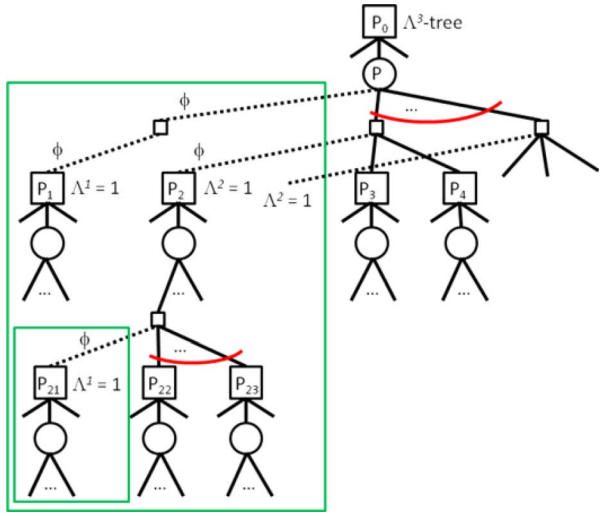

Fig. 19. Proof search tree for the position in Fig. 3(a).

P_2 does the position after a seminull move at 7 in Fig. 4(c). Initially, let NCTU6-Verifier of one higher order run in P. Since VCST-Solver is able to find the winning move for P_1, the defender (Black) should place at least one stone in zone $Z_2(P_1)$. Consider one square s in $Z_2(P_1)$, say square 7 in Fig. 4(c). For the seminull move at 7, choose move 8 and then use NCTU6-Verifier (without raising one order) to derive that the attacker wins at 8. Thus, move 8 is a Λ^3-move. By verifying all null and seminull moves in P, we show that move 6 in Fig. 4(a) is a Λ^4-move (from Definition 1).

Furthermore, the attacker is shown to win at 6 in a Λ^4-strategy as follows. In our experiment, the attacker wins for all defensive (nonnull) moves by finding Λ^3-strategies. For example, for move 7 in Fig. 21, NCTU6-Verifier is recursively employed to find a Λ^3-strategy, where moves 8–12 are shown to be Λ^3-moves.

In the proof search tree shown in Fig. 20, we found three seminull moves that are Λ^3-moves with value 1 [like P_2 which is also 7 in Fig. 4(c)], and 569 defender Λ^3-moves in total. Move 12 in Fig. 21 is the deepest Λ^3-move. In this experiment, experts

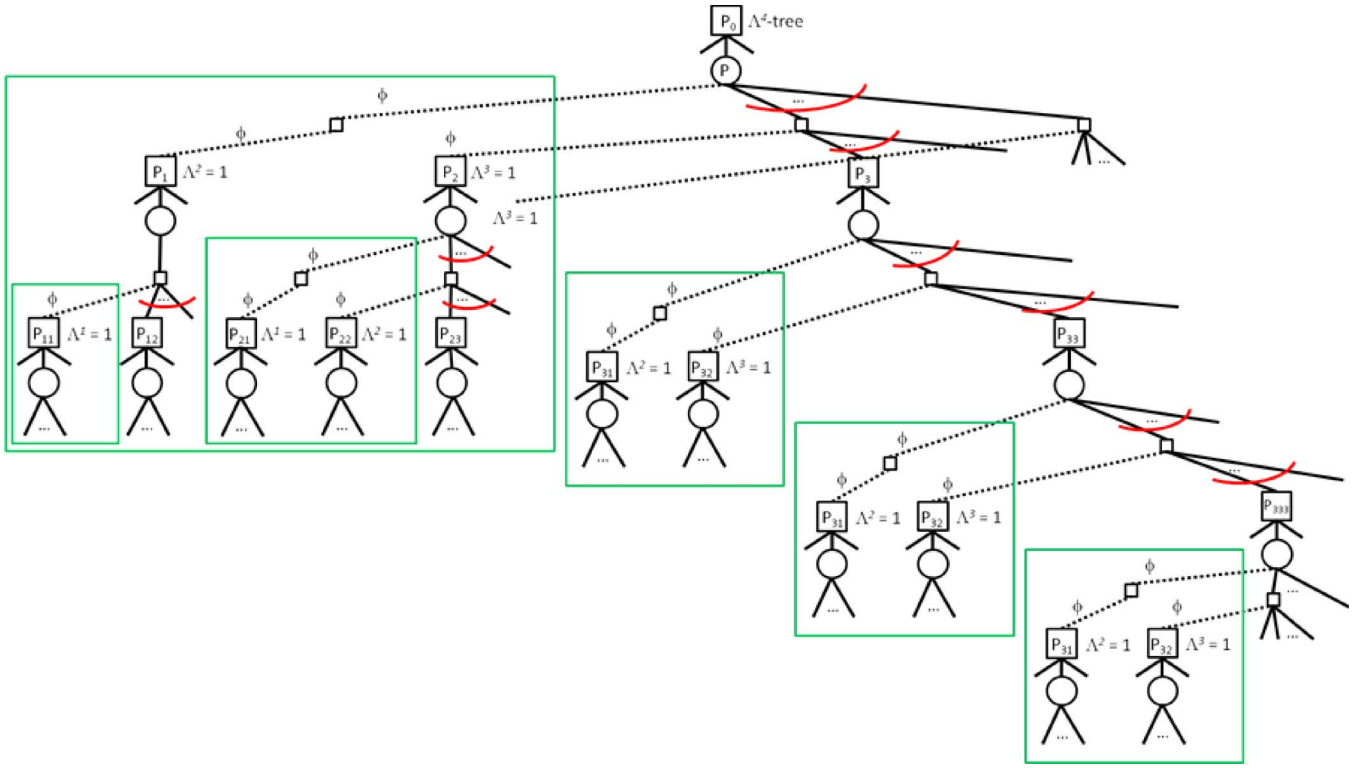

Fig. 20. Proof search tree for the position in Fig. 4(a).

TABLE I
STATISTICS OF SOLVING POSITIONS

Position	Number of nodes	Time (in seconds)
Fig. 2 (a)	65054	18.39
Fig. 3 (a)	498380	104.41
Fig. 4 (a)	210229668	44448.07

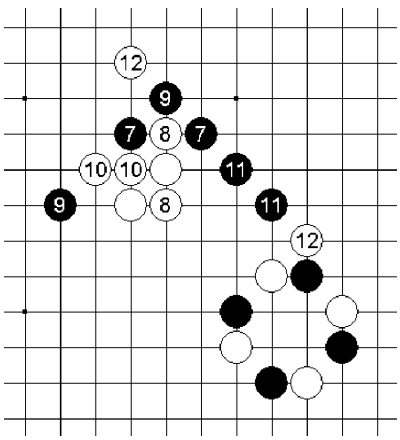

Fig. 21. Sequence of Λ^3-moves starting from 7.

helped find 26 winning nonthreat moves, including move 6 discovered by Huang [11].

Table I shows the number of nodes as well as the computation times used by our system to solve the positions in Figs. 2(a), 3(a), and 4(a) on an Intel Pentium Dual 2.00-GHz machine. The positions in Figs. 2(a) and 3(a) are solved without experts' assistance, while the position in Fig. 4(a) is solved with the help of experts, as above. All the above experiments were performed on 19×19 boards that most current *Connect6* tournaments use. We also used a simple tool to verify that the above example is still winning even in an infinite board. The details are omitted.

C. More Results

In addition to the two positions illustrated in Section V-B, we investigated more positions. Initially, we had experts use the integrated system to help us solve about ten more positions. Wu *et al.* [28] had recently automated with success the proof process by developing a new search algorithm, called job-level proof-number (JL-PN) search. Using the JL-PN search together with our RZOP search, we solved many more positions, up to 65 positions in total, with Λ^3-strategy, within a couple of months. The details of the 65 positions were listed in [27].

Among the 65 positions, 34 were not solved by the scheme, called the VCDT-for-null scheme. The scheme uses VCDTs (not VCSTs) after seminull moves in proof search trees such as the one in Fig. 2(c). If no VCDTs were found for the seminull moves as the one in Fig. 3(c), then the scheme failed to solve positions. In brief, the proof search trees in our RZOP search are as in Fig. 18(a), while those in the scheme are as in Fig. 8.

Many positions were not solved by the VCDT-for-null scheme illustrated below. For the three openings in Fig. 22(c), (d), and (f), the winning moves are live threes at 3. For the seminull moves that use the only stones to block Black's live threes, Black has no more double-threat moves to make. That is, Black cannot win by VCDTs. However, Black actually wins by VCSTs for these seminull moves. Hence, it is important that the proposed RZOP can solve them correctly.

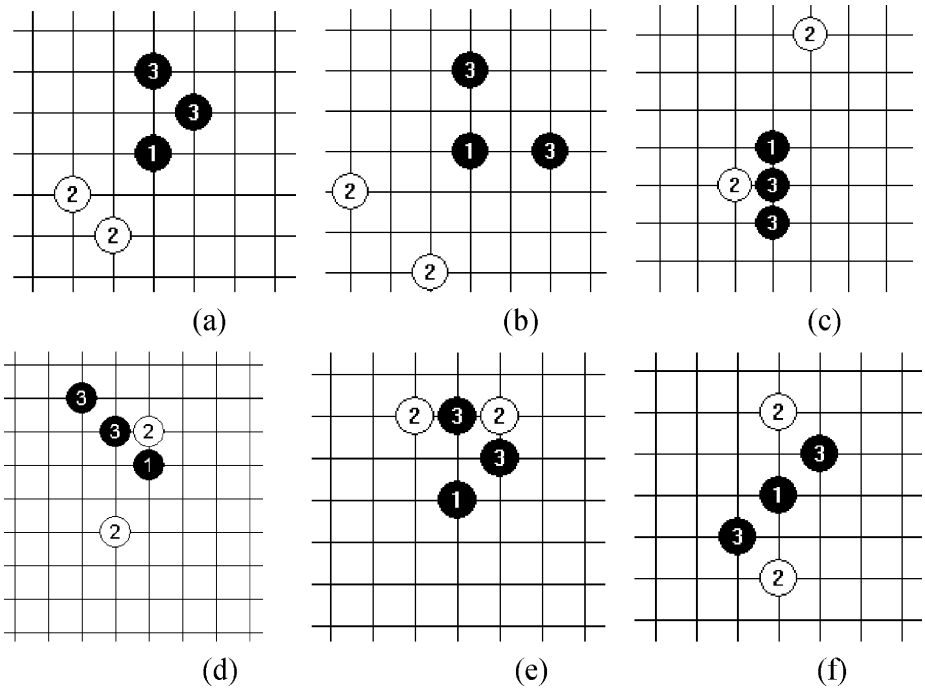

Fig. 22. Six three-move openings in which Black wins at 3.

The 65 positions include 12 three-move openings, among which ten cannot be solved by the VCDT-for-null scheme. Six of the ten openings are shown in Fig. 22. In particular, the fifth one, Mickey Mouse opening, used to be one of the popular openings before we solved it. Mickey Mouse opening was so named in [20], since White 2 and Black 1 together look like the face of Mickey Mouse. The sixth one, also called straight opening, is another difficult one.

Now, the question is whether there exist more cases requiring Λ^4-strategies like the one in Fig. 4. Since the one in Fig. 4 is the only one that we found so far, it is still an open problem to find some more.

VI. Conclusion

This paper investigates a new threat-based proof search for *Connect* games. The contribution of this paper is mainly the new search method, named RZOP search that uses relevance zones to help solve many positions in *Connect6* as well as *Connect* games. In theory, this method can be applied to *Connect* games with infinite boards. Practically, this paper demonstrates the method by solving two typical winning positions in Figs. 3(a) and 4(a) on 19×19 boards, as well as many *Connect6* positions and openings in Section V. In addition, the method can also be easily incorporated into *Connect6* program, such as NCTU6.

This paper also leaves some open problems.

1) Investigate more winning positions in *Connect6* that require Λ^4-strategies, such as the one in Fig. 4(a).
2) Investigate whether there exists a Λ^5-strategy in *Connect6*.
3) Optimize the proof search tree by pruning more branches efficiently [29].
4) Apply the new method (in the Appendix) to solving some real positions in general *Connect* games.
5) Investigate whether dual lambda search [19] is useful for *Connect6* or *Connect* games.

Using the JL-PN search together with our RZOP search, we successfully solved up to 65 positions with Λ^3-strategy. The 65 positions include 12 three-move openings; in particular, Mickey Mouse opening, which used to be one of the popular openings before we solved it. One might ask whether or when *Connect6* on 19×19 boards will be solved. So far, we still could not solve tens of the common openings, many of which experts believed to be well balanced for both players. Hence, the answer to this question is still unknown.

Appendix
Proof Search for Connect Games

In this Appendix, the verifier $V_{C6}(P, S)$ is generalized to general *Connect* games, *Connect*(m, n, k, p, q), while maintaining Property RZV. The generalized verifier is denoted by $V_{CK}(P, S)$. In the case that P is an endgame position or is in the attacker's turn (described in Sections IV-B and IV-C, respectively), the verifier $V_{CK}(P, S)$ is the same as $V_{C6}(P, S)$. So, the rest of this Appendix describes the verifier only in the case that P is in the defender's turn. Furthermore, the position P (in the defender's turn) can be classified into the following two: 1) the number of attacker threats t in P is at least $p + 1$; and 2) the number t is at most p. In the first case, the attacker wins already. Therefore, the verifier returns 1 and constructs relevance zones in the following operation.

Tp1-1) Construct relevance zones by following both operations T3-1 and T3-2, except that the terms "$i+2$" are replaced by "$i + p$."

Similar to Lemma 7, Lemma 11 shows that the verifier also satisfies Property RZV in this case.

Lemma 11: Assume that the defender is to move and the number of the attacker threats is at least $p+1$ in P. The verifier described above satisfies Property RZV.

Proof: The proof is similar to that of Lemma 7 and therefore omitted. ■

In the second case that the number of attacker threats t is at most p, the verifier performs the following operations.

Tp-1) For each of critical defenses M_D (both normal and relaxed), perform the following.
  a) Return 0 if the subverifier $V_{\rm sub}(M_D, P, S)$ returns 0. Note that the subverifier is described below.
  b) Let $\Psi(P) = \Psi(P) \cup \Psi'(P_D)$.

Tp-2) Continue to construct relevance zones in operation Tp1-1, and return 1.

In operation Tp-1a, a subverifier $V_{\rm sub}(M_D, P, S)$ is used to verify whether the attacker wins for all defender moves M'_D dominated by M_D in P, where M'_D has p squares (but M_D may have less than p squares). By dominate, we mean that all squares in M_D must also be in M'_D, but not *vice versa*. For the subverifier $V_{\rm sub}(M_D, P, S)$, the constructed zone is denoted by $\Psi'(P_D) = \langle Z'_1(P_D), Z'_2(P_D), \ldots, Z'_r(P_D)\rangle$, where $P_D = P \oplus M_D$. In addition, the subverifier satisfies the following property (proved in Lemma 12).

Property RZS: If $V_{\rm sub}(M_D, P, S)$ returns 1, the following condition holds. For all defender moves M'_D dominated by M_D, there exists some Ψ'_D such that $\Psi'_D \subseteq \Psi'(P_D)$ and Ψ'_D is in $RZ(P \oplus M'_D)$.

The subverifier $V_{\rm sub}(M_D, P, S)$ performs the following operations.

Par-1) Assume that M_D has exactly $p - u$ defender stones, where u is the number of null stones in M_D and $0 \leq u \leq p$. In the case that $u > 0$, move M_D is a null or a seminull move.

Par-2) Return 0 if $V_{CK}(P_D, S)$ returns 0, where $P_D = P \oplus M_D$.

Par-3) Let $\Psi'(P_D) = \Psi(P_D) \ll u$.

Par-4) Return 1 if $u = 0$, i.e., the move is not a null or a seminull move.

Par-5) For each of unoccupied square $s \in \neg_{P_D}(Z_u(P_D))$, perform the following.
  a) Let the defender move $M_{D,s}$ be $M_D + \sigma_D(s)$.
  b) Return 0 if $V_{\rm sub}(M_{D,s}, P, S)$ returns 0.
  c) Let $\Psi'(P_D) = \Psi'(P_D) \cup \Psi'(P_{D,s})$, where $P_{D,s} = P \oplus M_{D,s}$.

Par-6) Return 1.

Lemma 12 shows that the subverifier satisfies Property RZS, if all the recursive $V_{\rm sub}$ in Par-5b satisfy Property RZS and the verifier V_{CK} in Par-2 satisfies Property RZV.

Lemma 12: For a subverifier $V_{\rm sub}(M_D, P, S)$ as described above, it satisfies Property RZS by assuming that all the recursive $V_{\rm sub}$ in Par-5b satisfy Property RZS and that the verifier V_{CK} in Par-2 satisfies Property RZV.

Proof: Assume that $V_{\rm sub}(M_D, P, S)$ returns 1. Consider all defender moves M'_D (including p stones) that are dominated by M_D. Namely, let $M'_D = M_D + \sigma_D(\varphi)$, where φ has u additional unoccupied squares. For this lemma, it suffices to prove that there exists some Ψ'_D such that $\Psi'_D \subseteq \Psi'(P_D)$ and Ψ'_D is in $RZ(P \oplus M'_D)$. All of these defender moves M'_D are classified into the following cases.

1) All defender moves M'_D in which all additional squares s in φ are in $\neg_{P_D}(Z_u(P_D))$. The proof for this case is similar to that for case 1 in Lemma 10 as follows. Since this subverifier returns 1, the verifier $V_{CK}(P_D, S)$ in Par-2 returns 1. Since the verifier V_{CK} returns 1 and also satisfies Property RZV (from this lemma), $\Psi(P_D)$ is in $RZ(P_D)$. Since all additional $s \in \neg_{P_D}(Z_u(P_D))$, we obtain from Lemma 3 that $\Psi(P_D) \ll u$ is in $(P_D + \sigma_D(\varphi))$. Since

$$\begin{aligned} P_D + \sigma_D(\varphi) &= (P \oplus M_D) + \sigma_D(\varphi) \\ &= P \oplus (M_D + \sigma_D(\varphi)) \\ &= P \oplus M'_D, \Psi(P_D) \ll u \end{aligned}$$

is also in $RZ(P \oplus M'_D)$. In addition, since $\Psi(P_D) \ll u \subseteq \Psi'(P_D)$ from Par-3 in $V_{\rm sub}$, $\Psi(P_D) \ll u$ is the Ψ'_D.

2) All defender moves M'_D where some additional square s in φ is in $Z_u(P_D)$. Since this subverifier returns 1, the recursive $V_{\rm sub}(M_{D,s}, P, S)$ at Par-5b returns 1 as well, and therefore, satisfies Property RZS. From Property RZS, there exists some Ψ such that $\Psi \subseteq \Psi'(P_{D,s})$ and Ψ is in $(P \oplus M'_D)$. Since $\Psi'(P_{D,s}) \subseteq \Psi'(P_D)$ from operation Par-5c, we obtain $\Psi \subseteq \Psi'(P_D)$. Thus, Ψ is the Ψ'_D. ■

From Lemma 12, we derive Lemma 13 as follows.

Lemma 13: Assume that the defender is to move and the number of attacker threats is at most p in P. The verifier described above satisfies Property RZV by assuming that all the recursive subverifiers in operation Tp-1a satisfy Property RZS.

Proof: Assume that this verifier returns 1. For this lemma, it suffices to prove that the constructed $\Psi(P)$ is in $RZ(P)$. Since the verifier returns 1, all the recursive subverifiers in operation Tp-1a returns 1 as well. Assume that these subverifiers satisfy Property RZS. For proving $\Psi(P) \in RZ(P)$, it suffices to prove from Lemma 6 the following: for all defender moves M_D, there exists some Ψ_D such that Ψ_D is in $RZ(P \oplus M_D)$ and $\Psi_D \subseteq \Psi(P)$. All defender moves M_D are classified into the following two cases.

1) All defender moves M_D that block all the threats. There must exist some critical defense M'_D (either normal or relaxed) dominating M_D. Since $V_{\rm sub}(M'_D, P, S)$ returns 1 and satisfies Property RZS from the above, there exists some Ψ_D from the property such that $\Psi_D \subseteq \Psi'(P \oplus M'_D)$ and Ψ_D is in $RZ(P \oplus M'_D)$.
2) All defender moves M_D that leave some threat unblocked. The attacker wins by connecting up to p on some unblocked threat segment, like S_{3T}. From the proof in Lemma 11, we obtain that there exists some Ψ_D such that $\Psi_D \subseteq \Psi'(P)$ and Ψ_D is in $RZ(P_D)$. ■

Theorem 3 concludes that the verifier $V_{CK}(P, S)$ in all cases satisfies Property RZV. Therefore, if $V_{CK}(P, S)$ returns 1, the constructed $\Psi(P)$ is in $RZ(P)$, and the attacker wins in P from Corollary 2. It can also be observed that the operations in Section IV-D are special cases of the operations described in this Appendix.

Theorem 3: The verifier $V_{CK}(P, S)$ satisfies Property RZV in all cases.

Proof: By induction, the verifier $V_{CK}(P, S)$ satisfies Property RZV in all cases from the above lemmas. ■

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments.

REFERENCES

[1] L. V. Allis, "Searching for solutions in games and artificial intelligence," Ph.D. dissertation, Dept. Comput. Sci., Univ. Limburg, Maastricht, The Netherlands, 1994.

[2] L. V. Allis, H. J. van den Herik, and M. P. H. Huntjens, "Go-Moku solved by new search techniques," *Comput. Intell.*, vol. 12, pp. 7–23, 1996.

[3] L. V. Allis, M. van der Meulen, and H. J. van den Herik, "Proof-number search," *Artif. Intell.*, vol. 66, no. 1, pp. 91–124, 1994.

[4] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays*, 2nd ed. Natick, MA: A K Peters. Ltd., 2003, vol. 3.

[5] A. de Bruin, W. Pijls, and A. Plaat, "Solution trees as a basis for game-tree search," *Int. Comput. Chess Assoc. J.*, vol. 17, no. 4, pp. 207–219, Dec. 1994.

[6] T. Cazenave, "Abstract proof search," in *Computers and Games*, ser. Lecture Notes in Computer Science, T. A. Marsland and I. Frank, Eds. Berlin, Germany: Springer-Verlag, 2001, vol. 2063, pp. 39–54.

[7] T. Cazenave, "A generalized threats search algorithm," in *Computers and Games*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2003, vol. 2883, pp. 75–87.

[8] C.-P. Chen, I.-C. Wu, and Y.-C. Chan, "ConnectLib – A Connect6 Editor," 2009 [Online]. Available: http://www.connect6.org/Connect6Lib_Manual.htm

[9] Chinese Association for Artificial Intelligence, "Chinese Computer Games Contest," (in Chinese) [Online]. Available: http://www.caai.cn/

[10] H. J. van den Herik, J. W. H. M. Uiterwijk, and J. V. Rijswijck, "Games solved: Now and in the future," *Artif. Intell.*, vol. 134, no. 1–2, pp. 277–311, 2002.

[11] Y.-C. Huang, *Private Communication.* 2008.

[12] P.-H. Lin and I.-C. Wu, "NCTU6 wins man-machine Connect6 championship 2009," *Int. Comput. Games Assoc. J.*, vol. 32, no. 4, pp. 230–232, 2009.

[13] T. W. Lee, "One of early Tsumegos for Connect6," 2005 [Online]. Available: http://www.connect6.org/web/index.php?option=com_tsumego&task=loadTsumegoHistoryList&class_id=32

[14] Littlegolem, "Online Connect6 Games," 2006 [Online]. Available: http://www.littlegolem.net/

[15] Renju International Federation, "The International Rules of Renju," 1998 [Online]. Available: http://www.renju.net/study/rifrules.php

[16] Renlib, Renju—A Ranju Editor [Online]. Available: http://www.renju.se/renlib/

[17] W. Pijls and A. de Bruin, "Game tree algorithms and solution trees," in *Computers and Games*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 1999, vol. 1558, pp. 195–204.

[18] G. Sakata and W. Ikawa, *Five-in-a-Row*. Tokyo, Japan: The Ishi Press, 1981.

[19] S. Soeda, T. Kaneko, and T. Tanaka, "Dual lambda search and its application to Shogi endgames," in *Advances in Computer Games*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2006, vol. 4250, pp. 126–139.

[20] Taiwan Connect6 Association, Connect6 Homepage, [Online]. Available: http://www.connect6.org/

[21] ThinkNewIdea Inc., CYC Game, (in Chinese) 2005 [Online]. Available: http://cycgame.com/

[22] T. Thomsen, "Lambda-search in game trees – with application to Go," *Int. Comput. Games Assoc. J.*, vol. 23, no. 4, pp. 203–217, 2000.

[23] J. Wagner and I. Virag, "Solving Renju," *Int. Comput. Games Assoc. J.*, vol. 24, no. 1, pp. 30–34, 2001.

[24] I.-C. Wu, "Proposal for a New Computer Olympiad Game—Connect6," 2005 [Online]. Available: http://ticc.uvt.nl/icga/news/Olympiad/Olympiad2006/connect6.pdf, or http://www.connect6.org/articles/RZOP/connect6.pdf

[25] I.-C. Wu, C.-P. Chen, P.-H. Lin, K.-C. Huang, L.-P. Chen, D.-J. Sun, Y.-C. Chan, and H.-Y. Tsou, "A Volunteer-computing-based grid environment for Connect6 applications," in *IEEE Int. Conf. Comput. Sci. Eng.*, Vancouver, BC, Canada, Aug. 29–31, 2009, pp. 110–117.

[26] I.-C. Wu and P.-H. Lin, "NCTU6-lite wins Connect6 tournament," *Int. Comput. Games Assoc. J.*, vol. 31, no. 4, pp. 240–243, 2008.

[27] I.-C. Wu and P.-H. Lin, "Benchmark for RZOP search," [Online]. Available: http://www.connect6.org/articles/RZOP/

[28] I.-C. Wu, H.-H. Lin, P.-H. Lin, D.-J. Sun, Y.-C. Chan, and B.-T. Chen, "Job-level proof-number search for Connect6," presented at the Int. Conf. Comput. Games Kanazawa, Japan, 2010.

[29] I.-C. Wu, H.-H. Lin, and P.-H. Lin, "A more efficient proof search for Connect6," 2010, in preparation.

[30] I.-C. Wu, D.-Y. Huang, and H.-C. Chang, "Connect6," *Int. Comput. Games Assoc. J.*, vol. 28, no. 4, pp. 234–242, 2006.

[31] I.-C. Wu and D.-Y. Huang, "A new family of k-in-a-row games," in *Advances in Computer Games*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2006, vol. 4250, pp. 180–194.

[32] I.-C. Wu, D.-J. Sun, H.-H. Lin, P.-H. Lin, C.-P. Chen, L.-P. Chen, and H.-Y. Tsou, "A volunteer computing system for Connect6 Applications", National Chiao Tung Univ., Hsinchu, Taiwan, Tech. Rep., 2010.

[33] I.-C. Wu and S.-J. Yen, "NCTU6 wins Connect6 tournament," *Int. Comput. Games Assoc. J.*, vol. 29, no. 3, pp. 157–158, Sep. 2006.

I-Chen Wu (M'10) received the B.S. degree in electronic engineering and the M.S. degree in computer science from the National Taiwan University (NTU), Taipei, Taiwan, in 1982 and 1984, respectively, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 1993.

Currently, he is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan. His research interests include artificial intelligence, Internet gaming, volunteer computing, and cloud computing.

Dr. Wu introduced the new game *Connect6*, a kind of six-in-a-row game, and presented this game at the 2005 11th Advances in Computer Games Conference (ACG'11). Since then, *Connect6* has become a tournament item at the Computer Olympiad. He led a team developing a *Connect6* program, named NCTU6. The program won the gold twice at the Computer Olympiad in both 2006 and 2008.

Ping-Hung Lin is currently working towards the Ph.D. degree at the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.

He is the Current Chief Designer of the *Connect6* program NCTU6 that won the gold twice at the Computer Olympiad in both 2006 and 2008. His research interests include artificial intelligence and grid and cloud computing.